



HAL
open science

Preserving Individual Privacy with Personal Data Management Systems

Iulian Sandu Popa

► **To cite this version:**

Iulian Sandu Popa. Preserving Individual Privacy with Personal Data Management Systems. Computer science. Université de Versailles Saint-Quentin-en-Yvelines; Université Paris-Saclay, 2021. tel-03531619

HAL Id: tel-03531619

<https://inria.hal.science/tel-03531619>

Submitted on 18 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



université PARIS-SACLAY

UNIVERSITE DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

Laboratoire Données et Algorithmes pour une Ville
Intelligente et Durable David –UR 7431

HABILITATION A DIRIGER DES RECHERCHES

DISCIPLINE / SPECIALITE : INFORMATIQUE / BASES DE DONNEES

Présentée par :

Iulian SANDU POPA

Preserving Individual Privacy with Personal Data Management Systems

Soutenue le : 10 décembre 2021

JURY

Mr Amr EL ABBADI, Professeur, University of California Santa Barbara (Rapporteur)

Mme Indrakshi RAY, Professeur, Colorado State University (Rapporteur)

Mr Vincent ROCA, Chargé de Recherche, HDR, Inria Rhône-Alpes (Rapporteur)

Mr Sébastien GAMBS, Professeur, Université du Québec à Montréal (Examineur)

Mme Esther PACITTI, Professeur, Université Montpellier 2 (Examineur)

Mme Karine ZEITOUNI, Professeur, UVSQ (Examineur)

Mr Luc BOUGANIM, Directeur de Recherche, Inria Saclay-Ile-de-France (Tuteur)

Acknowledgments

As my bibliography attests, all of my research is the result of teamwork and fruitful collaborations I have had over the years with the wonderful people I have been fortunate enough to meet and work with in my field. I am deeply grateful to all my co-authors and to all the (many) unnamed people who have supported my work.

Contents

1	Introduction	2
1.1	Historical Overview of My Research Background	4
1.1.1	Ph.D. and Post-doc (2006-2011)	4
1.1.2	Associate Professor (since 2011)	5
1.1.3	Sketch of My Research Contributions	6
1.2	Overview of My Research on Spatio-temporal Data and Related Topics	8
1.3	Overview of Research Projects and Software Development	10
1.3.1	Research Projects and Industry Collaboration	10
1.3.2	Software Development	12
1.4	Document Outline	13
2	Towards an Extensive and Secure PDMS Architecture	16
2.1	Context and Motivation	18
2.2	Existing Personal Cloud Solutions	19
2.2.1	Online Personal Cloud Solutions	20
2.2.2	Zero-knowledge-based Personal Clouds	21
2.2.3	Home Cloud Software	21
2.2.4	Home Cloud Plugs	22
2.2.5	Tamper-resistant Home Cloud	22
2.2.6	Synthesis of the Existing Approaches	23
2.3	Approach and Scientific Results	24
2.3.1	Specificities of Data Management in the PDMS Context	25
2.3.2	Security Properties of a PDMS	26
2.3.3	Extensive and Secure PDMS Architecture	28
2.3.4	Concrete PDMS Instances	31
2.4	Related Contributions and Future Work	34
2.4.1	Scientific Contributions	34
2.4.2	More Related Challenges and Contributions	35
3	Scalable PDMS Search Engine with Secure Hardware	38
3.1	Context and Motivation	40
3.2	Related Work	41
3.2.1	Search Engine Requirements	41
3.2.2	Hardware Constraints of Secure Tokens	42
3.2.3	State-of-the-Art Solutions	43
3.2.4	Problem Formulation	45

3.3	Approach and Scientific Results	45
3.3.1	Design Principles	46
3.3.2	Write-Once Partitioning and Linear Pipelining	47
3.3.3	Background Linear Merging	48
3.3.4	Document Deletions	49
3.3.5	Conditional Top-k Queries	51
3.3.6	Experimental Validation	53
3.4	Related Contributions and Future Work	54
3.4.1	Scientific Contributions	54
3.4.2	Future Work	55
4	Preserving Data Confidentiality in Fully-Decentralized PDMS Systems	57
4.1	Context and Motivation	59
4.2	Related Work	60
4.3	Approach and Scientific Results	62
4.3.1	SEP2P Architectural Design and Threat Models	62
4.3.2	SEP2P Security Requirements	64
4.3.3	Verifiable Random Actor Selection	66
4.3.4	Experimental Validation	73
4.4	Related Contributions and Future Work	74
4.4.1	Scientific Contributions	74
4.4.2	Future Work	74
5	Conclusion and Future Research Perspectives	77
Appendix A	Ph.D. Students	85
A.1	Saliha Lallali	85
A.2	Dai-Hai Ton-That	85
A.3	Julien Loudet	85
A.4	Robin Carpentier	86
A.5	Julien Mirval	86
Appendix B	Curriculum Vitae	88
Appendix C	Personal Bibliography	95
Appendix D	Personal Data Management Systems: The security and functionality standpoint	101
Appendix E	Supporting secure keyword search in the personal cloud	126
Appendix F	SEP2P: Secure and Efficient P2P Personal Data Processing	154
Appendix G	PAMPAS: Privacy-Aware Mobile Participatory Sensing Using Secure Probes	168
Appendix H	DIVERT: A Distributed Vehicular Traffic Re-Routing System for Congestion Avoidance	182

Acronyms	199
Bibliography	200

List of Figures

1.1	Sketch of my research contributions	7
2.1	Global (logical) architecture	29
2.2	Data collection	30
2.3	Physical ES-PDMS instances: (a) home box; (b) mobile device; (c) cloud based	34
3.1	Example of a basic secure PDMS platform	41
3.2	Typical inverted index structure	42
3.3	Linear Pipeline computation of Q over terms t_i and t_j	48
3.4	The Scalable and Sequential Flash structure	49
3.5	Linear pipeline computation of Q in the presence of deletions	51
4.1	Sketch of verifiable selection	69
4.2	Verifiable random	70

List of Tables

2.1	Main functionalities of the state-of-the-art personal cloud solutions	23
2.2	Trust considerations in the state-of-the-art of personal cloud solutions	24
2.3	Claimed properties of different execution environments (left) and required properties of the different modules of our ES-PDMS architecture (right)	32
4.1	Main notations in SEP2P protocols	66
4.2	Complementary notations in SEP2P protocols	69

Chapter 1

Introduction

Summary. *This chapter draws an overview of my research work starting from the Ph.D. up to the current time. It then presents a short overview of my work on spatio-temporal databases and mobility data, which is not detailed further in this document. It also presents briefly the main research projects in which I participated as well as the main software developments and related teachings.*

Contents

1.1	Historical Overview of My Research Background	4
1.1.1	Ph.D. and Post-doc (2006-2011)	4
1.1.2	Associate Professor (since 2011)	5
1.1.3	Sketch of My Research Contributions	6
1.2	Overview of My Research on Spatio-temporal Data and Related Topics	8
1.3	Overview of Research Projects and Software Development	10
1.3.1	Research Projects and Industry Collaboration	10
1.3.2	Software Development	12
1.4	Document Outline	13

1.1 Historical Overview of My Research Background

All of my research work can be roughly situated in the data management area. Nevertheless, I have had the chance to collaborate with researchers from other or related domains such as graph algorithmic, mobile computing, vehicular networks, distributed systems and security. My first solid contact with the academic research world has occurred in 2006 during a research internship in the context of a research-oriented Master's degree at the University of Versailles Saint-Quentin-en-Yvelines (UVSQ). This enriching experience fully confirmed my intention to pursue with a Ph.D. and, later on, to embark on an academic career. Since then I have been studying and contributing to, for shorter or longer periods, different topics such as: text mining, data and query models for mobile objects and sensors, spatio-temporal indexing and indexing techniques for Flash memory, compression of trajectory data, management of spatio-temporal data constrained by a transport network, and dynamic allocation of road traffic. However, since 2012 the focus of my research has been the secure management of personal data and more specifically: privacy-by-design architectures, storage and indexing of personal data, management of personal data embedded in secure devices, enforcement of personal data sharing policies, and distributed privacy-preserving querying.

In the rest of this section, I present a brief overview of my research background, the collaborations it involved and the thematic evolution from spatio-temporal databases to secure management of personal data. Section 1.2 presents the highlights of my research on spatio-temporal data management and mobile systems. Despite its importance but for the sake of brevity, this part of my work is not included in this document which is focused on the privacy-preserving personal data management. Sections 1.3.1 and 1.3.2 list the main research projects and software developments respectively, while Section 1.4 outlines the content and structure of the rest of the chapters in this document.

1.1.1 Ph.D. and Post-doc (2006-2011)

I received my Ph.D. from UVSQ in 2009 under the supervision of Karine Zeitouni and Georges Gardarin as a member of the Data Integration and Management (DIM) team at PRiSM lab. During my Ph.D. period I have explored mainly two topics: (i) modeling and querying of mobile location sensor data [100, 112] and (ii) indexing trajectory data [104]. In collaboration with Ahmed Kharrat, another Ph.D. student in the DIM team, I have also addressed some issues related to mining trajectory data [60].

The work on mobile location sensor data involved a collaboration with Jacques Erhlich and his team at the Livic laboratory at INRETS¹ (National Research Institute on Transports and their Security). The work on indexing trajectory data involved a collaboration with Dominique Barth and Sandrine Vial, two researchers from the graph algorithmic team at PRiSM lab. It also offered me the opportunity to collaborate with Vincent Oria from the New Jersey Institute of Technology (NJIT), a collaboration that was extended in the following years and it is still active as detailed below.

The Ph.D. period was followed by a post-doc until mid-2011 accomplished mostly in the DIM team at PRiSM lab. This post-doc allowed me to deepen my work on trajectory indexing [103], continue the software development of a framework for managing mobile sensor data (detailed in Section 1.3.2), and open to new research topics such as spatio-temporal data

¹Meanwhile, INRETS became IFSTTAR: www.livic.ifsttar.fr/linstitut/cosys/laboratoires/livic-ifsttar/

compression [105]. Also, Vincent Oria visited the PRiSM laboratory as an invited professor in 2010, which allowed us to strengthen our collaboration.

Invited researcher at NJIT (2011). Following the visit of Vincent Oria, I visited NJIT for two months in February-March 2011. During my stay I continued to work with Vincent Oria on trajectory data management related topics. Also, I have started a new collaboration with Cristian Borcea (Professor at NJIT) on topics at the border between mobile data management, mobile and distributed systems, dynamic traffic assignment, and vehicular networks. Finally, I had the occasion to closely work again with Cristian Borcea during his visit of the PRiSM laboratory in June-August 2011. Our collaboration has been fruitful over the years and is still active today.

1.1.2 Associate Professor (since 2011)

At the end of 2011, I was recruited as an Associate Professor at UVSQ in the Secure and Mobile Information Systems (SMIS) team. SMIS was a joint team between Inria Rocquencourt and UVSQ and led by Philippe Pucheral. In 2016, SMIS ended (a natural process for Inria teams which are limited in time to a maximum duration of 12 years) and its permanent members created in 2017 a new Inria/UVSQ joint team called PETRUS (PERsonal and TRUSTed cloud) under the lead of Nicolas Anceaix.

SMIS period (2011-2016). Before joining SMIS my knowledge on privacy-preserving data management was very limited. However, I was intrigued by this research domain which was already gaining significant momentum at that time. I should also underline the fact that SMIS was internationally recognized as the first research group to embed a relational database engine inside a smart card, a fascinating thing for any (young) researcher.

Joining SMIS had a major impact on my research work from several viewpoints and I will shortly evoke three of them. Thematically, although situated in the wide area of core database technology, SMIS focused on privacy protection related to personal data management, which was relatively distant from my early research topics. In particular, SMIS developed three research axes: (i) secure data management using highly secure but highly constrained hardware; (ii) secure data sharing and distributed computations; and (iii) privacy-by-design data management architectures.

Second, SMIS had a well-established software development strategy to see their academic contributions reach a real societal impact. Besides isolated prototypes that were regularly demonstrated at major database conferences, most of the software development efforts were organized around a unified platform with a triple objective: (i) federate the research results of the the team; (ii) employ it as a learning tool in advanced academic courses; and (iii) use it as a transfer vector in industry-oriented projects. This flagship platform was PlugDB [99] an operational tamper-resistant Personal Data Server dedicated to a secure and ubiquitous management of personal data.

Third, by searching to combine high level academic contributions and societal impact, SMIS understood the importance of addressing within a multidisciplinary approach the reciprocal entanglements between economic, legal, societal and technological aspects of its research objectives. Hence, the team members were involved in research projects and collaborations bringing together computer scientists, economists, jurists and sociologists.

All this, has appeared to me as a novel and highly engaging research environment and allowed me to embrace new research topics without hesitation. Besides, over the years, it had become clear to me that privacy-centric data management was a particularly challenging and

exciting research area.

PETRUS period (since 2017). After SMIS reached its maximum life duration in 2016, we created a new Inria team called PETRUS (PErsonal and TRUSTed cloud) integrating all the permanent members of SMIS. In PETRUS we continue to search for solutions in the privacy-centric data management area but the focus of the activity has shifted on building a secure and extensive Personal Cloud platform.

The Personal Cloud paradigm holds the promise of a privacy-by-design storage and computing platform, where each individual can gather her complete digital environment in one place and share it with applications and users, while preserving her control. However, this paradigm leaves the privacy and security issues in the user's hands, which leads to a paradox if we consider the lack of individuals' autonomy in terms of computer security, and ability and willingness to administer sharing policies. The challenge is however paramount in a society where emerging economic models are all based - directly or indirectly - on exploiting personal data. While many research works tackle the organization of the user's workspace, the semantic unification of personal information, the personal data analytics problems, the objective of the PETRUS project-team is to tackle the privacy and security challenges from an architectural point of view.

The research program of PETRUS is structured around three axes: (i) *Personal cloud server architectures*. Based on the intuition that user control, security and privacy are key properties in the definition of trusted personal cloud solutions, the objective is to propose new architectures (encompassing both software and hardware aspects) for secure personal cloud data management and formally prove important bricks of the architecture. (ii) *Global query evaluation*. The goal of this line of research is to provide capabilities for crossing data belonging to multiple individuals (e.g., performing statistical queries over personal data, computing queries on social graphs or organizing participatory data collection) in a fully decentralized setting while providing strong and personalized privacy guarantees. This means proposing new secure distributed database indexing models and query processing strategies. In addition, we concentrate on locally ensuring to each participant the good behaviour of the processing, such that no collective results can be produced if privacy conditions are not respected by other participants. (iii) *Economic, legal and societal issues*. This research axis is more transverse and entails multidisciplinary research, addressing the links between economic, legal, societal and technological aspects.

Invited researcher at NJIT (2018). Ever since my employment as an Associate Professor, I have continued to collaborate with Cristian Borcea and Vincent Oria at NJIT on topics related to mobile and distributed systems, participatory sensing and spatio-temporal data management. To enforce this collaboration, I visited a second time NJIT in March-April 2018. Some details about this work are presented in Section 1.2.

1.1.3 Sketch of My Research Contributions

Figure 1.1 draws a sketch of my main research axes and related contributions. At the top level, there is the *Personal Cloud architectures* topic. In this context, I study different research problems revolving around decentralized architectures for privacy-preserving data management. Such architectures leverage secure hardware [11] (e.g., smartcards or secure MCUs, or different Trusted Execution Environments [110] such as Intel SGX) at the user-side to enforce data confidentiality and integrity. This leads to different architectures, e.g., asymmetric architectures [8, 14, 15] (i.e., combining a large number of low-end but highly secure hardware

at the user-side with a high-end but untrusted supporting server) or fully decentralized architectures [79, 70, 78] (i.e., peer-to-peer). More recently, we structured the study of this topic by providing a more formal definition of the general set of functionality and security requirements that any *Personal Data Management System* (PDMS) should consider [9]. We then identified the challenges of implementing such a PDMS and propose a preliminary design for an extensive and secure PDMS reference architecture satisfying the considered requirements.

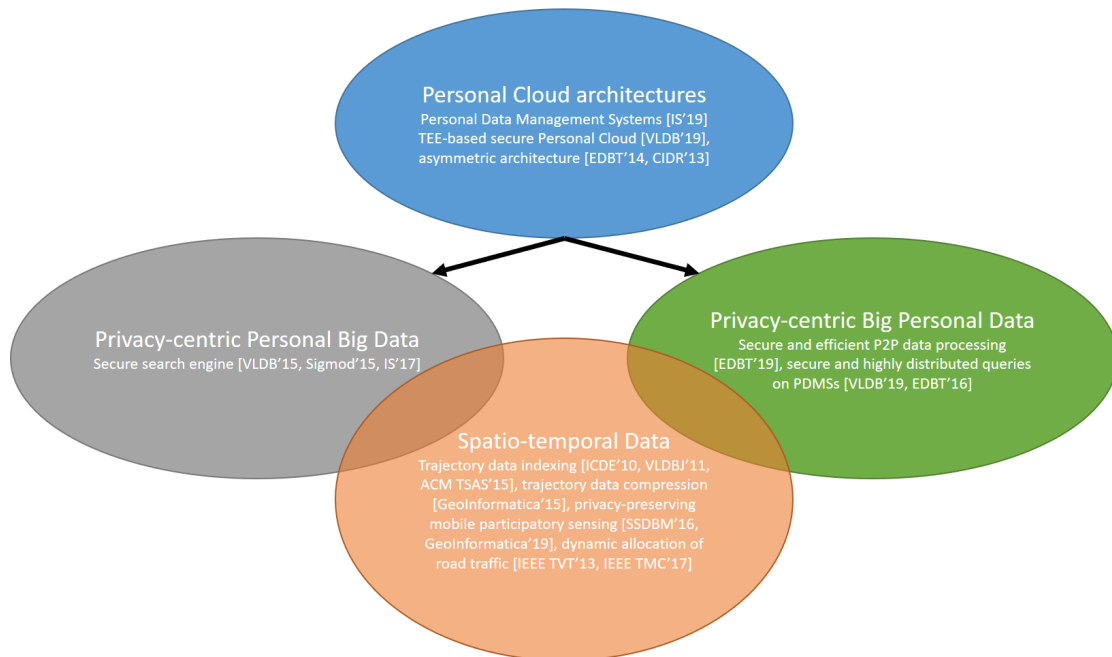


Figure 1.1: Sketch of my research contributions

Personal Cloud solutions allow individuals integrating in a single place all their personal data which are generally scattered across distinct and closed data silos. A first consequence is that this opens the way for novel applications able to cross-exploit individual's data (sometimes referred to as *Personal Big Data*, i.e., advanced computations over a person's data). However, integrating the entire digital life of an individual into a single system raises major issues for a privacy-preserving platform, i.e., related to securing the data collection, the storage and recovery, and the interactions of the user's app with her data. To address these issues, we need to conceive novel privacy-preserving methods adapted to the peculiar PDMS context. An example here can be a full-text search engine embedded in a secure token allowing the user to securely query her file collection [12] and securely apply the user-defined access control policies for all the apps that are permitted to access her data [69, 68].

A second important consequence of the Personal Cloud paradigm is personal data become massively distributed. In this context, an important issue needed to be addressed is: how can users/applications execute queries and computations over this massively distributed data in a secure and efficient way? Distributed computations over the personal data of (very) large sets of individuals unquestionably pave the way for *Big Personal Data* computations with many applications in a Personal Cloud context, like computing recommendations, launching participative studies, learning information using the data of users belonging to a community

(e.g., training a neural network in a patient community) or making collective decisions. However, this also requires privacy preserving implementations. A primary condition under which large sets of individuals would contribute with their own private data to collective uses is the guarantee that neither the other participants nor the infrastructure can access individual data. Hence, providing strong guarantees for the confidentiality [79, 78, 70] and the integrity of the distributed computations is paramount in this context.

In addition to the above mentioned three interconnected research axes, I also have an interest in topics related to spatio-temporal data management. I have developed this research axis during my Ph.D. and post-doc and continued to contribute to this domain afterwards. This work was mainly done in collaboration with Karine Zeitouni from UVSQ and Cristian Borcea and Vincent Oria from NJIT. I should note that over the years, the focus of my contributions in the spatio-temporal data field has naturally shifted to take into account the aspects related to privacy and data decentralization, which explains the intersection with my other research axes. Since the rest of this document is dedicated to my (most recent) contributions in the first three research axes, I briefly detail my major contributions in the spatio-temporal data field in the following section.

1.2 Overview of My Research on Spatio-temporal Data and Related Topics

Spatio-temporal data management was already a hot topic at the end of 2006 when I began to take an interest in it. Looking back in time, spatio-temporal data is to some extent the precursor of the Big Data era due to its specific features such as generating very large volumes, requiring fast and continuous updating, or being privacy sensitive. In this context, my first works focused on efficient data management of trajectory data from constrained moving objects (i.e., objects that move in a transport network such as vehicles) and led to contributions related to data indexing and data compression. Later on, my approach was to tackle mobility related issues from a more (distributed) system-oriented point of view. This led to contributions in the fields of dynamic road traffic allocation for congestion avoidance and mobile participatory sensing with strong privacy guarantees.

Trajectory data indexing. A first research topic was to efficiently retrieve the trajectories of objects moving in road networks. In this context, we proposed first an index structured called PARINET [104], which is based on a combination of graph partitioning and a set of composite B⁺-tree local indexes. PARINET is designed for historical data and relies on the distribution of the data over the network as for historical data, the data distribution is known in advance. Because the network can be modeled using graphs, the partitioning of the trajectory data is based on graph partitioning theory and can be tuned for a given query load. The data in each partition is indexed on the time component using B⁺-trees. PARINET can easily be integrated into any RDBMS, which is an essential asset particularly for industrial or commercial applications.

Since the trajectory data acquisition can be made in real time, we extend PARINET to solve the more general problem, i.e., indexing trajectory data flows. The Temporal PARINET (T-PARINET) [103] provides an optimized handling of trajectory data flows. T-PARINET is configurable in a dynamic environment and to fulfill its goal, T-PARINET uses an online tuning process that creates periodically a new PARINET to index the trajectory data from the current moment to a future moment in time. The online tuning process is based on

monitoring a set of parameters indicating the quality of the last built index in the structure of the T-PARINET. Hence, our approach is to propose a smooth between static indexes for continuous indexing of trajectory flows.

A second research topic was to consider the trajectory indexing problem in the context of novel storage devices. Due to several important features, such as high performance, low power consumption and shock resistance, NAND flash has become a very popular stable storage medium for embedded mobile devices, personal computers, and enterprise servers. However, the peculiar characteristics of flash memory require redesigning the existing data storage and indexing techniques that were devised for magnetic hard-disks. In this context, we proposed TRIFL [129], an efficient and generic TRajjectory Index for FLash. TRIFL is designed around the key requirements of trajectory indexing and flash storage. TRIFL is generic in the sense that it is efficient for both simple flash storage devices such as the SD cards and more powerful devices such as the solid state drives. In addition, TRIFL is supplied with an online self-tuning algorithm that allows adapting the index structure to the workload and the technical specifications of the flash storage device to maximize the index performance. Moreover, TRIFL achieves good performance with relatively low memory requirements, which makes the index appropriate for many application scenarios.

Trajectory data compression. With the continuous increase in volume of trajectory data, the problems concerning the transmission and the storage of such data have become prominent. A few works in the field of moving object databases deal with spatio-temporal compression. However, these works only consider the case of objects moving freely in the space. In this context, we tackled the problem of compressing trajectory data in road networks with deterministic error bounds [105]. We analyze the limitations of the existing methods and data models for road network trajectory compression. Then, we propose an extended data model and a network partitioning algorithm into long paths to increase the compression rates for the same error bound. We integrate these proposals with the state-of-the-art Douglas-Peucker compression algorithm to obtain a new technique to compress road network trajectory data with deterministic error bounds. The extensive experimental results confirm the appropriateness of the proposed approach that exhibits compression rates close to the ideal ones with respect to the employed Douglas-Peucker compression algorithm.

Dynamic allocation of road traffic. Traffic congestion causes driver frustration and enormous costs in lost time and fuel consumption. In this line of work, we proposed in [96, 98] five traffic re-routing strategies designed to be incorporated in a cost-effective and easily deployable vehicular traffic guidance system that reduces travel time. The proposed strategies proactively compute individually tailored re-routing guidance to be pushed to vehicles when signs of congestion are observed on their route. Extensive simulation results show the proposed strategies are capable of reducing the travel time as much as a state-of-the-art Dynamic Traffic Assignment (DTA) algorithm, while avoiding the issues that make DTA impractical such as lack of scalability and robustness, and high computation time. Furthermore, the variety of proposed strategies allows tuning the system to different levels of trade-off between rerouting effectiveness and computational efficiency. Also, the proposed traffic guidance system can significantly improve the traffic even if many drivers ignore the guidance or if the system adoption rate is relatively low.

However, centralized solutions for vehicular traffic re-routing to alleviate congestion suffer from two intrinsic problems: scalability, as the central server has to perform intensive computation and communication with the vehicles in real-time; and privacy, as the drivers have to share their location as well as the origins and destinations of their trips with the server.

In a subsequent work [97], we proposed DIVERT, a distributed vehicular re-routing system for congestion avoidance. DIVERT offloads a large part of the re-routing computation at the vehicles, and thus, the re-routing process becomes practical in real-time. To take collaborative re-routing decisions, the vehicles exchange messages over vehicular ad hoc networks. DIVERT is a hybrid system because it still uses a server and Internet communication to determine an accurate global view of the traffic. In addition, DIVERT balances the user privacy with the re-routing effectiveness. The simulation results demonstrate that, compared with a centralized system, the proposed hybrid system increases drastically the user privacy. In terms of average travel time, DIVERT's performance is slightly less than that of the centralized system, but it still achieves substantial gains compared to the no re-routing case. In addition, DIVERT reduces greatly the CPU and network load on the server.

Privacy-preserving mobile participatory sensing. Mobile participatory sensing (MPS) could be used in many applications such as vehicular traffic monitoring, pollution tracking, or even health surveying. However, MPS's success depends on finding a solution for querying large numbers of smart phones or vehicular systems, which protects user location privacy and works in real-time. To this end, we proposed first in [128, 130] PAMPAS, a privacy aware mobile distributed system for efficient data aggregation in MPS. In PAMPAS, mobile devices enhanced with secure hardware, called secure probes (SPs), perform distributed query processing, while preventing users from accessing other users' data. A supporting server infrastructure (SSI) coordinates the inter-SP communication and the computation tasks executed on SPs. PAMPAS ensures that SSI cannot link the location reported by SPs to the user identities even if SSI has additional background information. Moreover, in [101] we proposed an enhanced version of the protocol, named PAMPAS⁺, which makes the system robust even against advanced hardware attacks on the SPs. Hence, the risk of user location privacy leakage remains very low even for an attacker controlling the SSI and a few corrupted SPs. Our experimental results demonstrate that these protocols work efficiently on resource constrained SPs being able to collect the data, aggregate them, and share statistics or derive models in real-time.

1.3 Overview of Research Projects and Software Development

In this section, I present the most important research projects and industry collaborations in which I am or was involved in. Then, I briefly discuss some of my activities related to software development.

1.3.1 Research Projects and Industry Collaboration

My research projects and industry collaborations follow closely the research topics developed above in the spatio-temporal data and privacy protection areas. As it is often the case with collaborative projects, these projects involve researchers from different computer science areas (e.g., cryptographers and security experts), from different fields (e.g., humanities and social sciences) and industrial partners (e.g., Gemalto or Cozy Cloud).

MobiScope (2009). MobiScope is a DIGITEO-OMTE valorization project of a research prototype.

The objective of this project is to promote and enhance the development of a research prototype, called CALM [112], on mobile sensor data management that I implemented during my Ph.D. thesis. CALM has been selected for a market research phase. In this project, I was involved as the main designer of the platform and principal programmer of the software stack.

Although limited in scope, this project allowed me to have a first solid contact with industry and to understand the importance of going from a proof-of-concept prototype to a full-fledged system in technology transfer. The relative maturity of the prototype allowed me to employ it in advanced database courses and projects for Master students. More technical details about CALM are presented below (see Section 1.3.2).

ANR KISS (2011-2015). Partners: Yvelines Departmental Council, CryptoExperts, Gemalto, Inria (SMIS & SECRET team projects), LIRIS (INSA Lyon), PRiSM (UVSQ).

The global objective pursued by KISS (Keeping your Information Safe and Secure) was to provide a credible alternative to a systematic centralization of personal data on third-party servers and to pave the way for new privacy-by-design solutions dedicated to the management of personal data. The idea promoted in KISS is to embed, in trusted devices, software components capable of acquiring, storing and managing securely various forms of personal data (e.g., salary forms, invoices, banking statements, geolocation data, depending on the applications). These software components form a full-fledged Personal Data Server which can remain under holder's control. The scientific challenges include: embedded data management issues tackling regular, streaming and spatio-temporal data (e.g., geolocation data), data provenance based privacy models, crypto-protected distributed protocols to implement private communications and secure global computations. My contribution in KISS was threefold: (i) design and implement adapted data and query models for embedding spatio-temporal data into secure tokens [129, 128]; (ii) design and implement indexing methods to efficiently manage large document collections stored in such secure devices [12, 68]; and (iii) study fully-distributed privacy-by-design architectures allowing for large-scale distributed data management with trusted devices [8].

ANR PerSoCloud (2017-2021). Partners: Orange Labs, PETRUS (Inria-UVSQ), Cozy Cloud, UVSQ.

The objective of PerSoCloud (Personal and Social Trusted Cloud) is to design, implement and validate a full-fledged privacy-by-design Personal Cloud sharing platform. One of the major difficulties linked to the concept of personal cloud lies in organizing and enforcing the security of the data sharing while the data is no longer under the control of a central server. We identify three dimensions to this problem. Devices-sharing: assuming that the primary copy of user U1's personal data is hosted in a secure place, how to share and synchronize it with U1's multiple (mobile) devices without compromising security? Peers-sharing: how user U1 could exchange a subset of her data with an identified user U2 while providing to U1 tangible guarantees about the usage made by U2 of this data? Community-sharing: how user U1 could exchange a subset of his-her data with a large community of users and contribute to personal big data analytics while providing to U1 tangible guarantees about the preservation of her anonymity? In addition to tackling these three scientific and technical issues, a legal analysis will guarantee compliance of this platform with the security and privacy French and EU regulation, which firmly promotes the privacy-by-design principle, including the current reforms of personal data regulation. My contribution in PerSoCloud is twofold: (i) define an extensive and secure reference architecture for the personal cloud [9, 11]; and (ii) design and implement secure and efficient protocols allowing for data sharing and querying in (very) large communities of personal cloud users [79, 78].

MASTER (2018-2022). The consortium has 10 partners: 5 are European academic, 1 European non academic and 4 are International academic. <http://www.master-project-h2020.eu>

MASTER (Multiple ASpects Trajectory management and analysis) is a project funded

under the call H2020-MSCA-RISE-2017 with the objective of forming an international and inter-sectorial network of organisations working on a joint research program to define new methods to build, manage and analyse multiple aspects semantic trajectories. MASTER conveys the idea that pure movement data can be enriched with multiple heterogeneous contextual aspects. These aspects are intimately interconnected and should be referenced as a whole as holistic trajectories. Hence, the scientific objective is to propose methods to analyze and infer knowledge from holistic trajectories, considering as vital issues the privacy and big data dimensions. Currently, I have two secondments planned as part of the project: to Federal University of Santa Catarina, Brazil and to University of Pireus Research Center, Greece.

Industrial collaboration with Cozy Cloud (since 2014). Since 2012 Cozy Cloud has been developing Cozy (<https://cozy.io>), a free and privacy-friendly platform that allows users to integrate their personal data in a single place, under their control. A range of applications (Drive, Photos, Banks, Contacts, etc.) allow them to view, organize or share their data. At the same time, a system of connectors, developed in part by the open-source community, allows users to connect their accounts to third party services (energy, mobile/internet operators, banks, insurance companies, etc.) to automatically retrieve associated documents, such as invoices or health records, directly into their personal cloud. It is therefore natural that a strong collaboration has been established between Cozy Cloud and SMIS/PETRUS since 2013. Over time, this collaboration has grown through research projects (e.g., ANR PerSoCloud) or industrial projects (e.g., PIA Secsi 2016-2017) and CIFRE theses (i.e., two CIFRE Ph.D. theses defended in 2018 and 2019). In particular, the latest CIFRE thesis "Distributed and Privacy-Preserving Personal Queries on Personal Clouds", which was co-advised by Luc Bouganim and myself, has tackled scientific and industrial issues on the possibility of performing distributed and privacy-preserving queries on a large set of personal cloud nodes. This thesis has also opened the way for doing distributed machine learning over the shared personal data in large communities of personal clouds. A new CIFRE thesis with Cozy on distributed and secure machine learning, which I co-advise with Luc Bouganim, started in autumn of 2020.

1.3.2 Software Development

Given my research activity at the intersection between core database technology and personal data security and privacy, I believe software development is very important (and sometimes mandatory) to fully validate the research works. Thus, I have implemented and participated in the implementation and design of many prototypes, several of which have been demonstrated at international [128, 68, 70, 78] and national conferences [135, 80], or in scientific days (ANR days, during laboratory or research team evaluations, public open-doors days, etc.). However, as indicated above, my research group has a well-established software development strategy, which I fully share. That is, most of the software development efforts are organized around a unified platform to increase the odds that academic contributions reach a real societal impact (e.g., for technology transfer with industry partners or for academic teachings). Therefore, I present hereafter only the most important of my software related contributions.

CALM prototype. CALM: data management system of "CApteurs à Localisation Mobile". The CALM prototype is a spatiotemporal extension of a relational-object database for the management of mobile sensor data. It is implemented as a cartridge in Oracle 11g Server. I developed this prototype as part of my Ph.D. thesis and post-doc work. It concerns the management of mobile trajectories coupled with measurements collected along the trajectory by

embedded sensors. It aims to cover the whole process from data collection to map-matching (fusion with road data), modeling, storage, queries, optimization, exploratory analysis and finally visualization. CALM is used as a basis for teaching project development for Master students. CALM allows to go beyond the concepts related to managing, querying and indexing spatio-temporal data. It is a generic tool allowing users to comprehend the entire process of extending an RDBMS to integrate new data types and related operations and optimizations.

PlugDB related SW developments. PlugDB is a secure personal server which allows the individual to exercise control over their personal data, while preserving durability, availability and sharing (<https://project.inria.fr/plugdb/en>). It was the flagship development platform of SMIS. The main idea of PlugDB is to embed in secure hardware (e.g., smart cards with large storage capacity) software components capable of acquiring, storing and managing various forms of personal data (e.g., payment slips, bills, bank statements, medical data, geolocation traces, etc.) depending on the target applications. My interaction with PlugDB mainly concerned two extensions to allow integrating new data models into the database engine. First, we developed appropriate data types and operators to allow storing and querying spatio-temporal data (e.g., user trajectory traces) as well as an access method for Flash storage [129]. Moreover, we also considered the case of dealing with stream spatio-temporal data in PlugDB having mobile participatory sensing as application [128, 130]. These extensions were developed in the context of Dai-Hai Ton-That's Ph.D. thesis [127]. Second, we considered the case of querying document collections and developed appropriate indexing methods for PlugDB [68, 12, 70]. This extension was developed in the context of Saliha Lallali's Ph.D. thesis [67].

ES-PDMS platform on Intel SGX. With the shift of the research focus in PETRUS team on building an Extensive and Secure Personal Data Management System (ES-PDMS) (detailed in Chapter 2), we started developing a new flagship platform since the spring of 2020. The current platform is developed based on Intel SGX enclaves [38] which offer (some of) the security properties required by an ES-PDMS [9]. I am involved with several members of PETRUS in the design and development of the ES-PDMS platform, which will eventually integrate the majority of the team research results (see the research perspectives developed in Chapter 5).

1.4 Document Outline

This document focuses on my most recent contributions in the area of Personal Data Management Systems (PDMSs). An individual chapter is dedicated for each specific research branch I pursue in this field, i.e., Personal Cloud architectures, privacy-centric personal (single user) computations and privacy-centric collective computations (see the three upper parts of Figure 1.1). More specifically, I will detail the following topics:

- Chapter 2 introduces a reference architecture for extensive and secure PDMSs. Following the observation that existing Personal Cloud solutions do not cover all the major functionalities and specific threats in this context, we propose a definition of what an extensive (combining all functionalities) and secure (circumventing all the threats) PDMS should be. Then, we propose an abstract design for an ES-PDMS reference architecture satisfying the properties we have defined and briefly discuss some related challenges (e.g., linked to personal and collective computations).
- Chapter 3 addresses a challenge related to personal computations for a specific hardware

PDMS instance (i.e., secure token with mass storage of NAND Flash). In this context, we propose a scalable embedded full-text search engine to index large document collections and manage tag-based access control policies.

- Chapter 4 addresses a challenge related to collective computations in a fully-distributed architecture of PDMSs. It discusses the system and security requirements and proposes secure protocols to enable distributed query processing with strong security guarantees for a wide range of attacks (i.e., including the worst case of an attacker mastering many colluding corrupted nodes).

Without exception, the content of this document is based on published works, which are indicated at the beginning of each chapter. Thus, the main objective of this document is to give a general view of the addressed problems and proposed solutions and, therefore, many technical details, implementation issues and experimental results are left out. To ease the reading, Chapters 2 to 4 follow the same structure in four parts: (i) context and motivation, (ii) related work, (iii) approach and scientific results, and (iv) related contributions and future work. Chapter 5 concludes the document and presents future research perspectives.

The document is also accompanied by eight appendices (Appendices A to H). The first three appendices contain respectively: the list of my Ph.D. students (Appendix A), my *curriculum vitae* (Appendix B) and my bibliography (Appendix C). The following appendices contain a list of five selected articles being most representative for my research (Appendices D to H). I have selected these articles based on the following criteria: (i) the papers in Appendices D, E and F are most representative for the research work described in this document corresponding to Chapters 2, 3 and 4 respectively; (ii) Appendices E, F and G are most representative for the research work done by the three PhD graduate students that I co-advised; (iii) finally, Appendix H is most representative for the research work that I accomplished in the context of the international collaboration that I have with NJIT.

Chapter 2

Towards an Extensive and Secure PDMS Architecture

Summary. *Riding the wave of smart disclosure initiatives and new privacy-protection regulations, the Personal Cloud paradigm is emerging through a myriad of solutions offered to users to let them gather and manage their whole digital life. On the bright side, this opens the way to novel value-added services when crossing multiple sources of data of a given person or crossing the data of multiple people. Yet this paradigm shift towards user empowerment raises fundamental questions with regards to the appropriateness of the functionalities and the data management and protection techniques which are offered by existing solutions to laymen users. These questions must be answered in order to limit the risk of seeing such solutions adopted only by a handful of users and thus leaving the Personal Cloud paradigm to become no more than one of the latest missed attempts to achieve a better regulation of the management of personal data. In this chapter, we review, compare and analyze personal cloud alternatives in terms of the functionalities they provide and the threat models they target. From this analysis, we derive a general set of functionality and security requirements that any Personal Data Management System (PDMS) should consider. We then identify the challenges of implementing such a PDMS and propose a preliminary design for an extensive and secure PDMS reference architecture satisfying the considered requirements.*

Contents

2.1	Context and Motivation	18
2.2	Existing Personal Cloud Solutions	19
2.2.1	Online Personal Cloud Solutions	20
2.2.2	Zero-knowledge-based Personal Clouds	21
2.2.3	Home Cloud Software	21
2.2.4	Home Cloud Plugs	22
2.2.5	Tamper-resistant Home Cloud	22
2.2.6	Synthesis of the Existing Approaches	23
2.3	Approach and Scientific Results	24
2.3.1	Specificities of Data Management in the PDMS Context	25
2.3.2	Security Properties of a PDMS	26
2.3.3	Extensive and Secure PDMS Architecture	28
2.3.4	Concrete PDMS Instances	31
2.4	Related Contributions and Future Work	34
2.4.1	Scientific Contributions	34
2.4.2	More Related Challenges and Contributions	35

Chapter content and scientific publications:

- Sections 2.1 to 2.3 of this chapter overview the scientific results presented in [9] and as such are based on the content of that paper. The full version of the paper is included in Appendix D.
- Section 2.4 of this chapter browses through the other related publications.

This work was led in collaboration with Philippe Bonnet from Technical University of Copenhagen (Denemark) who visited the SMIS team during 2013/2014 with a funded Marie Curie grant. It also involved Benjamin Nguyen, former member of SMIS and currently at INSA Centre-Val-de-Loire. Part of this work was founded by the ANR PerSoCloud project (2017-2021).

2.1 Context and Motivation

Behaviors, movements, social relationships and interests of individuals are now constantly recorded, evaluated and analyzed in real-time by a small number of data aggregator companies [33]. This concentration negatively impacts privacy preservation and self-determination of individuals as well as innovation and fair competition between companies. Smart disclosure initiatives (e.g., Blue and GreenButton in the US¹, Midata in the UK², MesInfos in France³) are rising worldwide, aiming to restore individuals' control over their data and improve fairness in personal data management practices. The smart disclosure principle allows individuals to freely retrieve their personal data through a simple click, in a computer readable format, from the companies and administrations hosting them. According to the US government, this is a means to “help consumers make more informed choices; give them access to useful personal data; power new kinds of digital tools, products, and services for consumers; and promote efficiency, innovation, and economic growth” [109]. This fundamental principle has been recently translated into law with the *right to data portability* of the European General Data Protection Regulation (GDPR) [47].

Not only does smart disclosure allow individuals to be aware of the information collected about them, it also holds the promise of new services of high social and societal interest [109]. Indeed, individuals can now gather their complete digital environment in a so-called Personal Cloud or Personal Information Management Systems [1], Personal Data Server [4] or Personal Data Store [42]. A Personal Cloud is not only composed of data from many (previously) isolated information silos (e.g. secondary copies of data issued by their bank, employer, supermarket, hospital) but also of primary data (e.g. produced by quantified-self devices and smart meters, photos taken with their smartphone or documents stored on their PC). This unprecedented concentration of personal data opens the way for new value-added services when crossing multiple data of a given person (e.g., crossing medical data with eating patterns or bank statements with shopping history) or crossing the data of multiple people (e.g., conducting an epidemiological study), all this under the concerned individual's control. While this will certainly not stop data aggregator companies' current practices, the Personal Cloud introduces an alternative way to develop *fairer* personal data management services using richer personal data. Hence, smart disclosure and the related Personal Cloud concept have become the cornerstone of what is called today *user empowerment*.

However, we should be cautious of a potential boomerang effect of user empowerment: returning individual's their data without providing them with the appropriate environment to exercise their control over it. Several companies are now riding the Personal Cloud wave and the spectrum of proposals in the internet sphere is highly diverse. *Online personal cloud* solutions (e.g., CozyCloud, Digi.me, BitsAbout.Me to only cite a few) propose a centralized web hosting of personal data combined with a rich set of services to collect personal data from various sources, store them and cross-exploit them. This approach assumes, by construction, that individuals do not question the honesty of the hosting company (including the honesty of the employees) nor its capacity to defeat severe attacks, since centralization creates by essence a massive honeypot. *Zero-knowledge personal cloud* solutions (e.g., SpiderOak [121], Sync) mitigate this strong trust assumption by offering a fully encrypted (yet still centralized) data store, but impose a new responsibility (managing the encryption keys) on the individuals,

¹<https://www.healthit.gov/topic/health-it-initiatives/blue-button>

²<https://www.gov.uk/government/news/the-midata-vision-of-consumer-empowerment>

³<http://mesinfos.fng.org/>

thus trading user friendliness and rich services for improved security. *Home cloud software* solutions (e.g., OpenPDS [42], DataBox [53]) build upon the paradigm of local/edge computing by considering that the raw personal data should remain stored physically close to the user. Hence, they advocate for a decentralized approach where individuals install local personal servers on their own equipment (e.g., PC). *Home cloud plugs* (e.g., CloudLocker, Helixee) go further in this direction by offering a dedicated box that can store TBs of data, run a server and simply be plugged on an individual's home internet gateway, alleviating the burden of installing and administering a server. Finally *tamper-resistant home clouds* are home cloud plugs integrating secure chips on their hardware board to improve their resistance to confidentiality attacks, viruses and ransoms. All these alternatives belong to the large, fuzzy, personal cloud system family but neither provide the same set of functionalities nor consider the same threat model.

This diversity of solutions raises important questions. Which functionalities are really mandatory in the personal cloud context? Which threat model better captures the various uses and architectural environments of the personal cloud? Do solutions exist combining the required set of functionalities and appropriate threat model? If not, where does the difficulty stem from? Can solutions be devised by adapting existing corporate cloud-based techniques or is the personal cloud problem fundamentally different, thus imposing a deep rethink of these techniques? These questions are important for the data management research community and for the individual as well. By leaving these questions without answers, the risk is high to see the Personal Cloud paradigm be nothing but a missed attempt to reach a better regulation of the management of personal data, and maybe one of the last.

This chapter searches to answer these questions as follows: In Section 2.2, we review, compare and categorize the various personal cloud alternatives sketched above in terms of provided functionalities and targeted threat model. Beyond identifying the main expected features and privacy threats that need to be addressed, we also show that existing alternatives do not cover all of these features and threats, and cannot be combined for this purpose. In Section 2.3, we propose a definition of what an extensive (combining all functionalities) and secure (circumventing all the threats) Personal Data Management System should be. Therefore, we analyze the specificities of each functionality in the light of the individual context considered in this paper, and we deduce the corresponding security properties to achieve them. Then, we propose an abstract design for an ES-PDMS reference architecture satisfying the defined properties and illustrate how this abstract architecture can be instantiated in different concrete settings. Finally, Section 2.4 presents the related contributions and future research directions based on this work.

2.2 Existing Personal Cloud Solutions

The Personal Cloud concept originally appeared under different names such as Personal Information Management Systems [1], Personal Data Server [4] or Personal Data Store [42]. It attracts today significant attention from both the research and industrial communities. This section provides a short review of existing personal cloud solutions, representative of current approaches, to help understand the fundamental aspects in terms of *functional requirements* and *security/privacy threats*.

We distinguish cloud data management solutions designed for the corporate/enterprise context from those targeting individuals, i.e., tailored for personal use and referred to as

Personal Clouds in this section. Corporate cloud solutions offer digital tools to employees including, e.g., file storage space, email/agenda applications, file sharing with other employees based on permissions, user management and authentication based on LDAP repositories. They come as enterprise cloud tools either implemented as-a-service in the cloud or hosted on a server owned by the company and managed by internal administrators, such as SeaFile, Pydio, ownCloud/NextCloud, Sandstorm or Tonido/FileCloud. In terms of data management, the primary foci are multi-user features, workspace and user management, authentication, access control, privilege settings, administration and analysis tools. Such solutions do not apply to the personal cloud case and hence are not further detailed in this state of the art.

In contrast, solutions tailored for personal (and generally private) use are mono-user and seek to help users manage their entire digital life, i.e., by providing connectors to external data sources (e.g., bank, hospital, employer, social network, etc.), by allowing cross-data computation usages (e.g., linking the bank records of the individual with corresponding bills and email confirmations) and community uses based on groups of users sharing data for a social benefit (e.g., epidemiological study in a community of patients), by permitting their installation and configuration by laymen, and by helping individuals (rather than IT experts or administrators) understand and control data dissemination.

2.2.1 Online Personal Cloud Solutions

Many online personal cloud solutions flourish today such as CozyCloud [34], Digi.me [44], Meeco [88], BitsAbout.Me [25], Nextcloud [93] or Camilistore/Perkeep to name a few. Governmental programs like MyData [92] in Finland, MesInfos [90] in France or MyDex.org in the UK, target the same objective. These initiatives provide online personal cloud solutions to help users gather and store all their personal data in the same place and in a usable format, with the possibility to cross-exploit it through various applications. In terms of privacy and security, a common claim of these solutions is to proscribe any secondary usage (and in particular monetization of personal data) by the personal cloud provider and to guarantee to their users that their personal data is never disclosed to third parties except on their explicit request.

Overall, these solutions focus on a very similar set of functionalities, which cover the *collection of personal data*, *storage in an individual personal data store*, and the *integration of the data* such that transversal information processing (calles *cross-computations* hereafter) is made possible. However, while most initiatives claim to guarantee users' privacy, these approaches mainly rely on legal and economic frameworks with an unclear impact on the technical means to enforce security and privacy guarantees. Moreover, these approaches implicitly rely on very *strong hypotheses in terms of security*: (i) the personal cloud provider, employees and administrators are assumed to be fully-honest, and (ii) the overall personal cloud code as well as the whole set of personal applications and services running on top of it are considered trusted. Common security and privacy threats remain thus insufficiently addressed. Typically, data leakage resulting from attacks conducted against the personal cloud provider or the applications (which could be granted access to large subsets of raw personal data), or resulting from human errors, negligence or corruption of personal cloud employees and application developers, cannot be avoided in practice. This is critical because such solutions rely on a centralized cloud infrastructure settings which exacerbate the risk of exposing a large number of personal cloud owners, and hence may be subject to many sophisticated attacks.

2.2.2 Zero-knowledge-based Personal Clouds

Zero-knowledge personal clouds such as SpiderOak [121] or Sync [123] and to a certain extent MyDex or Digi.me mentioned above, propose architectural variations of the online Personal Cloud solutions in particular to mitigate some of the internal privacy issues raised by the strong assumption that the service provider is trusted. These solutions focus in particular on *secure storage* and *backup*. Regarding the secure storage, in most of the personal cloud solutions offering zero-knowledge storage, data is stored encrypted in the cloud and the user inherits the responsibility to store and manage the encryption keys elsewhere (and never transmit it to the outside nor to the personal cloud provider). Also, a common asset advertised by many zero-knowledge personal cloud services is secure backup, as a means to recover personal data when faced with a ransomware attack, personal device failure or unexpected data deletion. Thus, most zero-knowledge solutions, like SpiderOak, propose point-in-time recovery, such that users can recover any previous version of their personal files at a given date in the past.

Compared to the basic online personal cloud (see Section 2.2.1), the zero-knowledge personal cloud solutions offer a higher level of security since the personal cloud provider cannot access personal data in clear. However, the price to pay is a *minimalist functionality*, i.e., the difficulty to develop advanced services on top of zero-knowledge personal clouds, which reduces the uses of a zero-knowledge personal cloud to those of a robust personal data safe. Moreover, since data processing (beyond basic storage) cannot be delegated to the server, data-oriented treatments are embedded into the client applications, shifting the security and privacy issues to the client's side. On the other hand, the assumption of an honest client application (which has access to the decryption keys and to the raw data in clear) on which such cloud solutions rely, may be too strong to hold in practice (due to, e.g., the ubiquity of viruses), thus creating a vicious circle.

2.2.3 Home Cloud Software

Other personal cloud initiatives, called 'home cloud' hereafter, build upon the paradigm of local/edge computing. These initiatives consider that raw personal data should remain stored at the extremities of the network (e.g., within the user's equipment or close to the IoT device which produced it) as a means to circumvent the intrinsic security risks of data centralization (i.e., corruption of the server resulting in massive data leakage and illicit data usages). Some remarkable representatives of home cloud software solutions are OpenPDS [42] and DataBox [53]. Both focus mainly on new privacy models allowing users to reduce the amount of personal data exposed to remote parties (i.e., data services or other users) and audit data exchanges. The core of these proposals is based on a *trusted storage* hosted locally on the user's device or at the edge of the network, combined with *cross-computations* and *data sharing* such that users may consent revealing only query results to third parties instead of disclosing large amounts of sensitive raw data.

Compared with zero-knowledge solutions, formal security guarantees (in particular on the backup service) are lost. But interestingly, the impact in terms of 'minimalist functionality' is alleviated since advanced data services could potentially be provided as part of the personal cloud platform (which is obviously not compatible with the zero-knowledge guarantee). In addition, such solutions target user privacy protection against over-privileged third-parties and applications under the strong security assumption of having a fully-secured personal cloud software user side.

2.2.4 Home Cloud Plugs

Home cloud plug solutions such as Lima, Helixee [94], CloudLocker [35] and MyCloud, distinguish themselves from the previous approaches by providing solutions helping the users to self-host their personal data at home on a dedicated hardware platform. These solutions take the form of hardware plugs that can store TBs of data, which are synchronized with all the devices of the owner and can be accessed online.

The focus in terms of personal data uses is again mainly on *trusted storage and backup*, and to a certain extent *basic data sharing* to support data synchronization between all the devices of the user. However, collaborative uses involving personal data from multiple individuals are, to the best of our knowledge, outside of the scope of these approaches. In terms of privacy and security, the gain compared to home cloud software solutions is, to some extent, a better controlled client execution environment for the personal cloud software, which nevertheless does not provide strong security guarantees. These two complementary approaches pose a problem in terms of safely extending the data related functionalities. Indeed, implementing a new advanced data service would require either to extend the trusted code hosted on the hardware plug, which should be considered as a closed platform for security reasons, or to add it as an external app, which in this case would run on vulnerable client devices.

2.2.5 Tamper-resistant Home Cloud

To improve the security of home cloud plugs, research proposals like Personal Data Server (PDS) [4] and Trusted Cells [8] introduce secure (i.e., tamper-resistant) hardware at the network edges to manage the user's personal data. These approaches propose to embed a minimal Trusted Computing Base (TCB) dedicated to data management in the secure element of smart phones, set-top boxes or portable USB tokens to form a global decentralized secured data platform. The focus is on providing the users with *secure storage*, *secure cross-computations* and *secure distributed computations*. Let us take a closer look at the last functionality.

The possibilities of crossing data belonging to multiple individuals (e.g., performing statistical queries over personal data, computing queries on social graphs or organizing participatory data collection) while providing strong privacy guarantees have been explored in the context of a network of PDSs so that each user can keep control over her data. The personal data is stored locally in each user's PDS and the execution takes place on a hybrid infrastructure called an asymmetric architecture: on the one hand the PDSs of the participants are secure (i.e., behave honestly) but have low computation power, on the other hand, they are supported by an untrusted cloud infrastructure (e.g., honest-but-curious) implementing an IaaS or PaaS with significant storage and computing power. Different algorithms and computing paradigms have been studied on this architecture, from SQL aggregates [133] to special aggregation in a mobile participatory sensing context [130, 101]. In all cases, the challenge is to trade privacy for performances depending on the equilibrium between the secure computations executed by the secure PDSs and the ones delegated to the untrusted cloud infrastructure.

In conclusion, a high security level can be achieved since tamper resistant hardware is used. However, only rather simple queries can be addressed, and the underlying query engine being part of the TCB (running inside the secure microcontroller) must be proven secure. In addition, this approach leads to a non-extensible data system for two main reasons. First, supporting any new data and query model would require redesigning the underlying data storage, indexing and query processing techniques to comply with the strong constraints of tamper re-

sistant hardware. Second, any advanced and potentially extensible database processing (e.g., large pieces of code implementing user defined database functions, stored procedures, database workflows or involving existing libraries supporting data intensive processes) is proscribed as it cannot be integrated as part of the TCB. These two main drawbacks drastically limit the practicality and the genericity of the PDS approach. The security is hence achieved at the price of extensibility. Hence, such solutions mainly target ad-hoc applications managing highly sensitive data, e.g., personal Electronic Health Records.

2.2.6 Synthesis of the Existing Approaches

The solutions presented above address different functionalities of the personal cloud and consider different trust models, which are both summarized in Tables 1 and 2.

		Representative Personal Cloud approaches				
		<i>Online personal cloud</i>	<i>Zero-knowledge personal cloud</i>	<i>Home cloud software</i>	<i>Home cloud plug</i>	<i>Tamper resistant home cloud</i>
Functionality	<i>Storage</i>	Regular DBMS technology	Zero-knowledge cloud storage	Data stored on a generic user-side device, separated stores for different data sources	Data stored on a dedicated user-side device	Data stored in a tamper resistant device at the user-side
	<i>Backup</i>	Regular DBMS technology	Encrypted archive, point-in-time recovery	Replication / offline storage	Replication / offline storage	Replication / offline storage
	<i>Data collection</i>	Web scrapping	Files pushed or synchronized by the user to the cloud provider	Personal data inserted by the user or automatically to the home cloud	Personal files inserted by the user to the home cloud	Personal data pushed by the user or third parties to the home cloud
	<i>Cross-computations</i>	Transversal DB queries	At application level	Question answering, local data aggregation	At application level	Simple transversal DB queries
	<i>Distributed computations</i>					Simple distributed SQL statistics at large scale
	<i>Data dissemination</i>	[synchronization]	At application level	Minimum privileges for third parties and applications	[synchronization]	Minimum privileges for third parties and applications, secure access control

Table 2.1: Main functionalities of the state-of-the-art personal cloud solutions

In terms of functionalities (see Table 2.1), two important conclusions can be drawn. First, **the whole personal cloud data life-cycle must be covered**. We observe that the different solutions tackle different stages of the life cycle of the personal data in a personal cloud. In particular, all solutions discussed above address *data collection*, *storage*, *backup*, *cross-computations* and *data dissemination*. An *extensive* personal cloud solution should hence include all these functionalities to cover the whole personal data life cycle.

Second, **distributed computations should be part of the covered functionalities**. We also note that the *distributed computations* step is currently poorly covered. Is this because this functionality is less useful or because it is too difficult to be covered in practice in the personal cloud context? Regarding the utility of this functionality, we argue the opposite. Distributed computations over the personal data of (very) large sets of individuals unquestionably pave the way for Big –personal– Data computations with many applications in a personal cloud context, like computing recommendations, launching participative studies, learning information using the data of users belonging to a community (e.g., training a neural network in a patient community) or making collective decisions. However, this also

requires privacy preserving implementations. A primary condition under which large sets of individuals would contribute with their own private data to collective uses is the guarantee that neither the other participants nor the infrastructure can access individual data. This probably explains why the only line of work addressing this step is focusing on security and proposes solutions based on tamper resistant hardware.

Trust is another essential concern of the personal cloud (see Table 2.2). Two conclusions can be drawn for the state-of-the-art analysis.

		Representative Personal Cloud approaches				
		<i>Online personal cloud</i>	<i>Zero-knowledge personal cloud</i>	<i>Home cloud software</i>	<i>Home cloud Plug</i>	<i>Tamper resistant personal server</i>
Trust	<i>Considered threats</i>	secondary usages (monetization) performed by the cloud provider	Data snooping or leakage, secondary usages performed by the cloud provider, client device failure and ransomware attacks	Massive data snooping or leakage, secondary usages, over-privileged third parties and applications	Massive data snooping or leakage, pecuniary usages, home plug device failure	Massive data snooping or leakage, secondary usages, over-privileged third parties and applications
	<i>Trust model</i>	Fully-honest personal cloud provider, trusted personal cloud code, trusted applications	Honest-but-curious to Malicious cloud provider, trusted applications, trusted client device (before time of failure or ransomware attack)	Trusted personal cloud code, trusted client device, untrusted applications	Trusted personal cloud code, trusted home plug, trusted client applications	Trusted personal cloud code, honest-but-curious central supporting infrastructure, untrusted applications
	<i>Privacy and security measures</i>	Security standards, virtuous legal and economic framework, transparency and auditability of the code (apps/PDMS)	Client-side encryption, ‘no-knowledge’ cloud store	<u>SafeAnswers</u> , logical separation of the personal data stores, audit	Closed platform (dedicated device), physical ownership	Secure hardware, physical ownership small TCB, secure distributed protocols

Table 2.2: Trust considerations in the state-of-the-art of personal cloud solutions

First, **all the privacy threats considered in the state-of-the-art solutions must be circumvented** to protect user’s privacy and security in a meaningful way. Indeed, several threats are addressed by the different proposals, such as data snooping and secondary data uses performed by cloud providers (e.g., data monetization), corrupted applications or client devices (e.g., ransomware), or personal device failure. They all makes sense from a personal user point of view, since her whole digital life is managed and controlled using the platform.

However, a second (negative) conclusion is that **unifying these different solutions does not lead to a secure personal cloud architecture**. This is the case because building the union of the proposals would undeniably face irreconcilable architectural choices. Indeed, we observe that the existing personal cloud solutions cover a rather wide spectrum of architectural choices, but this leads to different – and sometimes contradictory – trust models and security measures. More precisely, each class of solutions addresses a specific subset of functionalities while considering a specific threat model. Our goal in the next section is to progress towards a clearer definition of what an *extensive* (covering all the functionalities) and *secure* (addressing all the threats) PDMS should be.

2.3 Approach and Scientific Results

Currently, the principles underlying most existing solutions seem to be directly inherited from those considered in the context of the corporate cloud and have not been rethought with personal use in mind. For example, many solutions examined in the previous section rely

on data encryption but nothing is said about restoring the master key in case of damage, except resorting on trivial unsecure protocols or on so-called-trusted third parties. Similarly, how to securely collect personal data from web sites through a myriad of unsecure wrappers without leaking both the user credentials needed to connect to the remote site and the collected personal data ? We argue that the way the PDMS functionalities are implemented and secured is determined by the intrinsic personal use of the PDMS, and must be deeply redesigned with this statement in mind.

In what follows, we first review in Section 2.3.1 some of the PDMS functionalities identified in the state of the art as needed to cover the whole data life-cycle and we analyze their intrinsic specificities. Then, in Section 2.3.2, we derive the fundamental security property attached to those functionalities. This analysis leads to the definition of an Extensive (i.e., providing the needed functionalities to cover the whole data life-cycle in a personal cloud) and Secure (i.e., achieving all the expected security goals) Personal Data Management System (ES-PDMS), which is provided in Section 2.3.3. Finally, Section 2.3.4 presents a few examples of concrete PDMS instances based on existing software and hardware security solutions.

2.3.1 Specificities of Data Management in the PDMS Context

As identified in Section 2.2.6, the PDMS functionalities are expected to cover the main stages of the personal data life-cycle and should thus integrate *data collection*, *storage and recovery*, *personal computations*, *distributed computations* and *data dissemination management*. For illustrative purpose, we select three functionalities, discuss their main specificities, and highlight to which extent they differ from their corporate data management system counterpart. The interested reader can refer to [9] for the complete discussion.

Data collection. The data collection functionality concerns both primary copies of user data (e.g., quantified-self data, smart home data, photos, videos, documents generated by the user, etc.) and secondary copies (e.g., banking data, health, employment, insurance, etc.). While the primary copies can be directly fed to the PDMS from data sources under the user's control, secondary copies have to be scrapped from the online services holding them. Collecting data from external sources is a basic operation of any corporate data management system. This task is usually handled thanks to a well-known and predefined set of carefully audited, patched and supported wrappers, under the control of data and security administrators who guarantee the quality and integrity of the integrated data. In the PDMS context, the situation is totally different. The PDMS owner is confronted with a large variety of scrappers (e.g., Web Outside of Browsers⁴) capable of capturing various types of data from a myriad of online services, the code of which cannot be trusted due to code complexity, diversity of contributors and sometimes closed source. However, by construction, such scrappers have access to highly sensitive data, from the credentials required to connect to the online service to the scrapped data itself (e.g., bank records, pay slips, invoices, medical records). Moreover, the user environment (e.g., operating system, network, other apps) in which the wrappers run is by far less trusted than an enterprise administered environment.

Personal computations. Personal computations in a PDMS usually refer to apps crossing various data of a single individual: the PDMS owner. We can distinguish between two types of such apps reflecting two specific uses of a PDMS: (i) apps used directly by the PDMS owner (e.g., for quantified-self, health and wellbeing, statistics and analyses related to smart

⁴weboob.org

home data or user’s mobility, etc.); and (ii) apps representing external services to which the owner willingly subscribes (e.g., billing apps allowing a car insurance company to compute the premium based on the owner’s car trajectory data – as in pay-as-you-drive, or an electric company to compute the bill based on the user’s electric smart meter traces). Therefore, an important specificity in the PDMS context is that apps “move” towards the data as opposed to personal data migrating towards remote services as it happens with most existing cloud services.

This has two main implications. First, the apps manipulate sensitive raw data, but neither the apps nor the environment in which they run can be trusted in general, leading to similar security problems as the ones discussed for data collection. *By-default auditing mechanisms* are thus required to detect malicious apps deviating from their manifest. These mechanisms must be easily understandable by non-expert users, disqualifying advanced audit tools based on complex models and formal languages usually employed by audit experts and security administrators in an enterprise context. Second, some external service apps need strong guarantees regarding the results produced by a PDMS (e.g., the billing apps discussed above). That is, an attestation process is required to ensure that the result was indeed produced by a certain computation code using all the required input data, and that the PDMS owner cannot tamper with the computation process nor the inputs.

Collective computations. Collective computations relate to various types of big personal data processes computed over a large set of PDMSs (e.g., contributing to participative studies, training a neural network, building an anonymous dataset). In addition to the security issues already discussed regarding the intrinsic untrusted nature of apps and computing environment, collective computations introduce a new difficulty. Gathering all the participants’ data in a single place to perform the computation introduces a single point of vulnerability and maximizes the incentive to attacks. Conversely, decentralizing the processing implies to temporarily transfer personal data among participants, transforming each into a potential attacker. In this latter case, two guarantees must be provided: (i) data confidentiality, i.e., any PDMS owner cannot access the data in transit of other participants, and (ii) distributed computation integrity, i.e., any participant PDMS can attest that any result it supplies corresponds to the assigned computation. Classical distributed computation techniques used in enterprise systems cannot apply here due to the unusual scale of the distribution (i.e., the computation may target a fraction of the population of a country). Generic secure multiparty computation protocols based on cryptographic techniques (MPC) are disqualified for the same reason (performance does not scale with the number of participants). Conversely, the participants cannot trust each other since participants are unknown a priori (and probably wish remaining anonymous).

2.3.2 Security Properties of a PDMS

As a conclusion of the preceding analysis, the PDMS context sketches an open and rich ecosystem of new untrusted data processing apps in interaction with an unsecure execution environment and a layman PDMS owner. This significantly contrasts with corporate data management systems where the applications and computing environment are significantly more static and carefully controlled by data and security administrators. Additionally, typical distributed computation infrastructures (cluster/cloud) strongly differ from a fully decentralized infrastructure of PDMSs (e.g., in terms of scale, ownership, legal agreements, deployed hardware and software, etc.). As a consequence, the PDMS must integrate by default novel

security measures to overcome the inherent weaknesses of the PDMS owner and tackle the specific threats to this open and untrusted ecosystem. We detail below the security properties expected from a PDMS, and linked to each functionality described in the previous section (one security property is thus associated with each step of the data life-cycle). We make no assumption about the technical means to enforce these properties, delaying this discussion to the next sections.

Piped data collection. Under the hypothesis of untrusted collection code and untrusted user computing environment, a PDMS is said to enforce *piped data collection* iff:

1. the only PDMS data accessible by the collection code are the credentials allowing access to the related data providers;
2. the credentials and the collected data related to a given data provider cannot be leaked outside the PDMS and the data provider.

This property guarantees that the only channel to the outside world provided to the data collector is a specified data provider and that the code is suitably isolated so as not to be able to leak data to a potentially corrupted user environment.

Bilaterally trusted personal computation. Under the hypothesis of untrusted external code and untrusted owner computing environment, a PDMS is said to enforce *bilaterally trusted personal computation* iff:

1. a personal computation may access only the owner's raw data specifically required for the computation;
2. only the final result of the computation - not the raw data - may ever be exposed to a third party;
3. the execution of the computation produces trustworthy audit trails accessible to the owner;
4. the PDMS can provide a proof that the result of the computation was produced by the expected code.

This property provides bilateral guarantees to the PDMS owner and the third party willing to execute code on the owner's data. It guarantees to the former that the minimal collection principle enacted in laws protecting personal data (e.g., GDPR) is fulfilled, that the computation cannot leak unexpected data and finally that she will have the ability - not the obligation - to audit the compliance to this property. Conversely, it guarantees to the latter (e.g., an energy provider willing to compute the owner's bill) that the code remotely sent to the PDMS has been accurately computed. If this computation combines several tasks, the proof produced by the PDMS must guarantee that the orchestration of these tasks cannot be tampered with without the caller being able to detect it. However, the PDMS cannot attest by itself that the data targeted by this code is genuine. Such attestation remains under the responsibility of the computation code, assuming that the data has been properly signed by their producer.

Mutually trusted collective computation. Under the hypothesis of untrusted external code and untrusted user computing environment, a PDMS is said to enforce mutually trusted collective computation iff:

1. a collective computation may access only the participants' raw data specifically required for the computation;
2. only the result of the computation - not the raw data - may ever be exposed to a third party or to any participant;
3. the execution of the computation on a participant generates trustworthy audit trails accessible to that participant;

4. a proof can be provided that the result of the computation was produced by the expected code over the expected set of participants.

This property targets the same objective as its bilaterally trusted personal computation counterpart. It must integrate the fact that participants can contribute to the collection phase and/or the processing phase of the computation with no assumption on the cardinality of these two sets of participants and on their intersection.

Definition 1. EXTENSIVE AND SECURE PDMS. *An Extensive and Secure Personal Data Management System provides the expected set of functionalities to cover the complete data life cycle in a personal cloud, namely data collection, storage and recovery, personal cross-computations, collective computations and data dissemination management, and is compliant with their respective security properties counterparts, namely piped data collection, mutual data at rest protection [9], bilaterally trusted personal computation, mutually trusted collective computation and controlled data dissemination [9].*

2.3.3 Extensive and Secure PDMS Architecture

Designing an extensive and secure PDMS architecture providing the functionalities and the five security properties indicated above is highly challenging, given the fundamental tension between the security expectations of a layman PDMS owner and the need for supporting an open ecosystem of applications running on an untrusted environment. In this section, we introduce the fundamentals of a PDMS reference architecture tackling this tension and detail its building blocks. We then show that physical instances of this reference architecture can be already envisioned today and discuss how existing and forthcoming software and hardware mechanisms may impact the satisfaction of our security properties.

Logical Architecture and Building Blocks

Ideally, the support of potentially complex manipulations of personal data while preventing unexpected data leaks could be achieved by securely collecting, storing and manipulating the data in a secure subsystem, managed under the control of the holder, while never letting the (untrusted) applications or third parties directly access the raw data. Such a clean separation can only be based on the assumption that there exist a few generic functions that can be used to manipulate personal data without violating privacy. Such an assumption is obviously a fantasy, because the most interesting manipulations of personal data are application-specific and consequently, privacy violations are also application-specific.

To solve this problem, we propose a three-layer logical architecture where a minimal Secure Core (Core) implementing basic operations on personal data is extended with Isolated Data Tasks (Data tasks) themselves accessed by Applications (Apps) on which no security assumption is made (see Figure 2.1). The objective is to control the flow of raw personal data from the Core to the outside, such that only expected results are declassified to untrusted applications or third parties.

Core. The Core is a secure subsystem that is a Trusted Computing Base (TCB) ideally minimal, inextensible, proven correct through formal methods and isolated from the rest of the system. The Core must provide all basic operations required to enforce the confidentiality, integrity and resiliency of the personal data hosted by the PDMS. It must be the unique entry point to manipulate this data. The Core must thus implement a data storage module. A policy enforcement module must be integrated in the Core to regulate the data access performed by

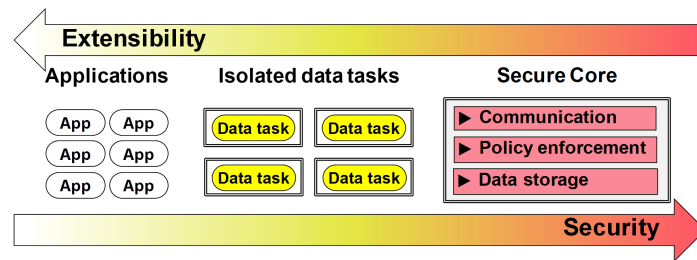


Figure 2.1: Global (logical) architecture

the other layers of the architecture. A communication manager is also needed to securely communicate with other users, applications and third parties.

Data Tasks. Data tasks are introduced as a means to deal with application-specific personal data management. The idea is to control complex data-oriented tasks by (1) splitting their execution into data tasks evaluated in a sufficiently isolated environment to maintain control on the data accessed by the Core and delivered to the Apps in order to avoid any side effect in terms of data leaks, and (2) scheduling and verifying the execution of data tasks by the Core such that security and privacy can be globally enforced.

Apps. Any developer should be able to develop an application to ensure a wide and diverse application panel. However, the complexity of these applications (large code base, extensible and not proven) and their execution environment (web browser, smartphone, etc.) make them vulnerable. Therefore, no security assumption is made on applications, which manipulate only authorized data resulting from data tasks but have no privileges on the raw data.

Given this global logical architecture, our approach is to identify the required elementary building blocks to satisfy the defined security properties (see Section 2.3.2) and explain how they should be combined to reach the expected goal without introducing security breaches.

Each building block in turn relies on a set of common security primitives provided by the Operating System (OS) and/or the hardware platform hosting the PDMS. Hence, these primitives are the foundations of our empirical minimal definition of the Core. Since they are commonly used by various building blocks, we present them first. As their implementations differ across platforms, we concentrate below of the security primitives they provide.

Common security primitives:

- **Isolation.** A component of the architecture is said to be isolated if (i) the internal execution state of the component cannot be accessed nor influenced from the outside of the component except with the collaboration of the system administrator (e.g., the PDMS owner) and (ii) the component may not observe nor influence the behavior of any external system except through its own inputs/outputs behavior. See for example [48] for a survey of ways to implement code isolation in a partly untrusted context.
- **Attestation.** A component is said to be attestable if a trustworthy certificate can be produced to demonstrate that the component output was indeed produced by the specific code of this component. This common security primitive is usually considered for a whole complex system (e.g., [29]). It was formalized in the case of a single task running on a complex system in [21].
- **Confidentiality.** A component is said to ensure data confidentiality if neither the internal execution state nor the input and output data of the component can be leaked to any system other than the party initiating the component (in our case the PDMS Core) even with the collaboration of the system administrator. This property is described in [57] and

formalized together with isolation and attestation as ‘secure outsourced computations’ in [21].

- *Peripherals isolation.* A component is said to satisfy peripherals isolation if the data exchanged between that component and the peripherals cannot be leaked outside of the component except with the intervention of the PDMS owner. For illustration purpose, [117] and [73] show how text messages can be securely displayed even in the presence of an untrusted OS using ARM TrustZone.

For the sake of brevity, we only present in the following, as illustrative example, the building blocks required for implementing the piped data collection security property. The complete discussion can be found in [9]. Let us note that the following two chapters of this document are dedicated to contributions related to other two security properties, i.e., bilaterally trusted personal computation and mutually trusted collective computation.

Piped data collection. The piped data collection property requires the ability to execute arbitrary data collection code (e.g. a scrapper) in a secure manner. The objective of this property is actually twofold.

First, it should guarantee that the collection code will not access any data stored in the PDMS other than the credentials required to connect to the related service provider (e.g., the web site to be scrapped). This specific privilege must be part of the manifest declared at the time this collection code is registered in the PDMS.

Second, it should guarantee that the collected data and the credentials related to a given data provider cannot leak to any third party (including another data provider targeted by the same collection code). According to the reference architecture sketched in Figure 2.1, this guarantee can be provided by considering the collection code as an isolated data task and granting a write access to this data task only to the destination PDMS. This requires data task authentication to be implemented in the Core. In other words, this means restricting the write capacity of the data task to the insertion of the collected data into the Core. The enforcement of this restriction at execution time relies itself on the code isolation property.

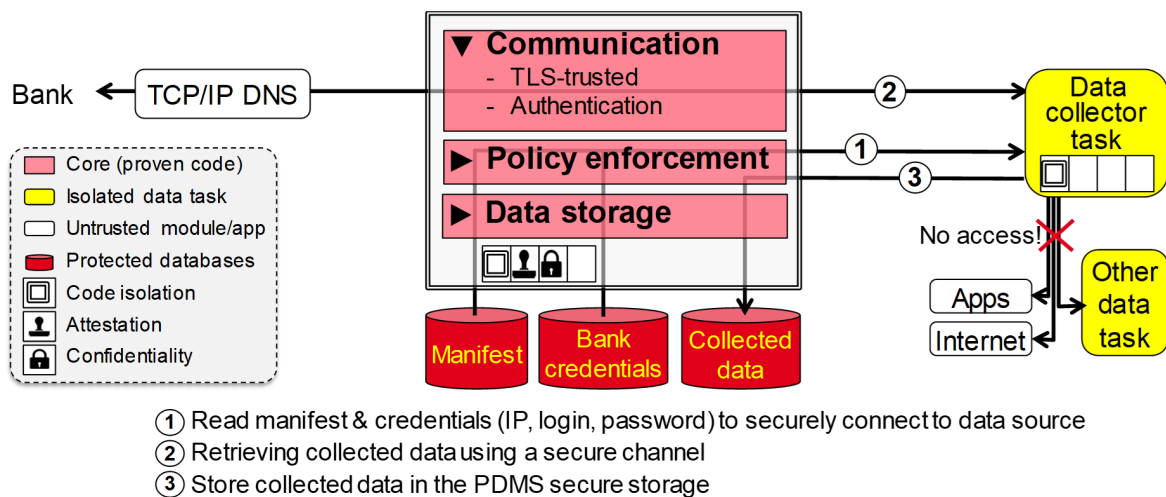


Figure 2.2: Data collection

While executing the collection code as an isolated data task inside the PDMS ensures that the effects on the Core are controlled, further measures are required in order to ensure the absence of leakage of both credentials and collected data. Indeed, the collection code needs

to communicate with the outside world in order to reach the data provider. To preclude the collection code to leak data to any other party than the related data provider, the Core must be able to establish a (TLS for example) secure channel between that specific data provider and the data task. Regarding the minimality objective, the complete network stack (e.g., TCP/IP, DNS) does not have to be in the Core, only the critical security operations of the creation of the secure channel, named TLS trusted in Figure 2.2, need to be.

Remark that each data collector task should be dedicated to a single remote site (e.g., my bank), to avoid a malicious data task from leaking credentials or personal data (e.g., the bank credentials/data to another site) through an authorized communication channel. Note that in addition a malicious data collector task may use the owner's credentials on the remote site to perform unexpected actions (e.g., the data collector retrieving the bank related data could trigger a money transfer using the bank credentials). However, such issues are mostly related to the definition of a weak security policy at the data provider side, rather than a problem to be addressed at the PDMS architectural level. We thus assume here that the credentials delivered by the data provider for data collection purposes will grant only read access to the reduced dataset of interest (e.g., bank account history).

Summing up the architecture presented in Figure 2.2, when performing data collection for a specific source, the Core launches an isolated data task executing the collection code for said source, provides it with the credentials and a secure channel to the source, which requires the implementation of data task authentication and secure channel set up (TLS-trusted) in the Core. Once collection is finished the collector returns the data to the Core which stores it appropriately. This ensures the absence of leakage (through isolation and secure channel), and proper behavior of the data collector in terms of input and output data (through access control). Another statement is that the safety properties are not independent from each other. Piped data collection indeed relies on controlled data dissemination and mutual data at rest protection to make sense when considered in a complete scenario.

Equally important, the Core must run in an execution environment that satisfies isolation, attestation and confidentiality (see Figure 2.2). The isolation property is required for the Core to protect it from all the other software components running on the same personal cloud platform (in particular the Apps and the data tasks). The attestation property is required for the collective computations which are orchestrated and/or executed by the Core. Finally, to provide mutual guarantees of security between PDMS users and third parties (see Section 2.3.2), the environment of the Core also has to provide confidentiality since it coordinates the distributed data tasks with potential access to private personal data supplied by other nodes, which remain hidden from the PDMS owner.

2.3.4 Concrete PDMS Instances

The goal of this subsection is to show that the logical architecture presented above can be instantiated in practice, using existing software and hardware solutions, and that its modular aspect helps defining physical PDMS instances easily. We first discuss here existing software and hardware security solutions, offering the required security primitives (namely isolation, attestation, confidentiality and peripheral isolation). Second, we show how to combine them into physical ES-PDMS architectures which instantiate three different configurations: (1) PDMS on a home box, (2) on a mobile device and (3) in the cloud.

Software-based security solutions. Let us first consider the security properties which could be provided by pure software-based solutions. The first of these properties is isolation,

which is intensively investigated as it constitutes a foundation of secure software architectures. Isolation is usually provided by an operating system (e.g., Linux, seL4, etc.), a virtual machine monitor (VMM or hypervisor, e.g., XEN, KVM) or a container manager (e.g., Docker, Kubernetes). Such solutions have the advantage to provide a high level of extensibility (in the sense that potentially complex/external code can be run). But enforcing the isolation security primitive means considering the underlying software components (the OS, VMM or container manager) as part of the trusted computing base (TCB), i.e., critical part of the software that, when compromised, can jeopardize the security of the entire system. The TCB must therefore be made completely free of vulnerabilities (e.g. bugs, buffer overflows, etc.) and ideally must be formally proven. This is a difficult task with large and complex code. Existing solutions rely on reducing the attack surface by minimizing the code which is part of the TCB (e.g., microkernels like seL4 and unikernels like MirageOS) or by hardening it (e.g., by adding access control, an IDS or a firewall [55]). Today, software solutions offer some form of isolation and peripheral isolation. However, assuring strong isolation guarantees in software is still an open issue (see [48] for a recent survey) and side channel attacks remain prominent [22] (thus, the mention ‘satisfied with limitations’ in Table 2.3). Moreover, since using software as a root of trust is still an unresolved problem [87], confidentiality against the PDMS owner and attestation of the code execution cannot be achieved purely through software. As a conclusion, the advantage of using VMMs in terms of extensibility is obvious, but such solutions should be considered in combination with other solutions (typically, secure hardware) to achieve the desired confidentiality and attestation properties of a PDMS (as summarized in Table 2.3).

Architecture	Code isolation	Attestation	Confidentiality	Peripherals isolation	Extensibility
Secure Element	✓	✓	✓		
TrustZone	✓	✓	✓	✓	≈
Intel SGX	✓	✓	✓	✓	✓
Software	≈			≈	✓

Modules	Code isolation	Attestation	Confidentiality	Peripherals isolation	Extensibility
Core	✓	✓	✓		
Data collector	✓				✓
Personall computation	✓	✓			✓
Collective computation	✓	✓	✓		✓
Data dissemination	✓			✓	✓
Applications					✓

The properties can be satisfied (✓) or satisfied with limitations (≈)

Table 2.3: Claimed properties of different execution environments (left) and required properties of the different modules of our ES-PDMS architecture (right)

Hardware-based trusted execution environments (TEE). Hardware TEEs target many different kinds of devices, from personal computers and IoT devices to cloud servers. The most prominent TEEs include secure elements as SIM cards (e.g., in smartphones), ARM TrustZone [126] for SoCs and CPUs integrated into smartphones, tablets and smart appliances, and Intel SGX [40] embedded in all recent Intel CPUs present in personal computers and cloud servers. Although there is still no unique security definition of TEE and their capabilities vary depending on proposals [110], most TEEs offer capabilities to run code in isolation and remote attestation, which allows it to prove required properties of the code running to third parties. Table 2.3 summarizes the security tools offered by three technologies: SGX, TrustZone and secure elements (smartcard). These three solutions mainly vary in the way confidentiality and

peripheral isolation are achieved and in terms of the possible level of extensibility of the code they can run:

- **Intel SGX** [40] integrates cryptographic primitives in hardware to isolate applications within enclaves, while providing confidentiality and attestation of the code executed in the enclave. Additionally, it provides mechanisms for managing the information flow from code running in the TEE to the outside world, both secure external storage primitives thanks to encryption and management of the communications between processes running in TEEs. It can be used both for securing Cloud apps and in the context of personal computers, leading to an advanced form of extensibility.
- **ARM TrustZone** technology isolates sensitive code executed in the secure area of the CPU, from application code executed in the rich area. It also aims to isolate the device's peripherals (typically, the screen of a smartphone) once code inside the secure area is executed. The code executed inside the secure area has some limitations in terms of resources (e.g., TrustZone ARM1176JZ based on Cortex A series is clocked at 772 MHz and can access several tens of MBs of RAM) leading to a certain form of extensibility.
- **Secure elements**, on the other hand, offer much less resources (e.g., advanced secure elements like ST33 based on ARM SecureCore SC300 Cortex M series is clocked at 60 MHz and has only 50KB RAM), thus having a negative impact on extensibility, if used to run a data task. In terms of security, on the contrary, such components provide –in addition to isolation and attestation– confidentiality (with strong guarantees due to tamper-resistance) for running code and data.

The common security primitives needed by the different modules of our ES-PDMS architecture and the primitives provided by these TEEs are reported in Table 2.3. Several conclusions can be drawn regarding the implementation of the logical architecture we envision: (i) it cannot be implemented using a single technology since none currently ensures all the required properties, and therefore has to combine several elements; (ii) the Core relying on isolation, confidentiality and attestation, can be run on a secure element or on SGX, or be implemented by a combination of execution environments (e.g., pioneer works like TrustVisor [87] propose a secure hypervisor based on combining a secure element with an hypervisor to provide confidentiality and attestation); and (iii) the data tasks needing peripherals isolation (i.e., decision-making) can only be run on TrustZone or a hypervisor/VMM.

Physical architectures of an ES-PDMS. In Figure 2.3 we propose three illustrative physical instances of our logical architecture adapted to three different ES-PDMS configurations based on a home box, a personal device and the cloud:

- **Home box** (Figure 2.3.a). The data is stored in the box and is cryptographically protected. The Core runs on a secure element as well as the collective computation tasks⁵ to benefit from its security and achieve the attestation and confidentiality requirements. Personal computation and data collector tasks are executed on a TrustZone CPU equipping the box, and thus benefit from extensibility. If no peripheral is available on the box to interact with the user (e.g., no touch screen), the apps and the decision-making tasks run on the smartphone of the PDMS owner, in the rich area and the trusted area respectively, to safely capture the user's I/O.

⁵The collective computation data tasks being operated on a secure element are highly secure (tamper resistant) at the price of extensibility. However, advanced distributed data processing tasks must be done with the support of a more powerful CPU integrated in the box and connected to the secure element, which is an open issue.

- **Mobile device** (Figure 2.3.b). Similarly to the home box, the Core and collective computation tasks are operated in the secure element of the smartphone (i.e., an additional SIM card slot is needed), the other tasks (personal computation, data collector and decision-making) run in the trusted area of the TrustZone CPU of the smartphone, while the apps run in the rich area of the CPU.
- **Cloud** (Figure 2.3.c). The SGX-based instance pictured on the right part of Figure 2.3 is running both the Core and the data tasks in distinct SGX enclaves. The enclave running the Core takes advantage of the attestation capabilities of SGX to control the other enclaves, and collective computations can be performed seamlessly (as well as other personal computation and data collector tasks) given the confidentiality property offered by SGX, with the Core relaying the remote attestation guarantees to the other participants of the protocol. To protect the user's I/O with the decision-making console, this architecture also needs a smartphone with a TEE (e.g., ARM TrustZone).

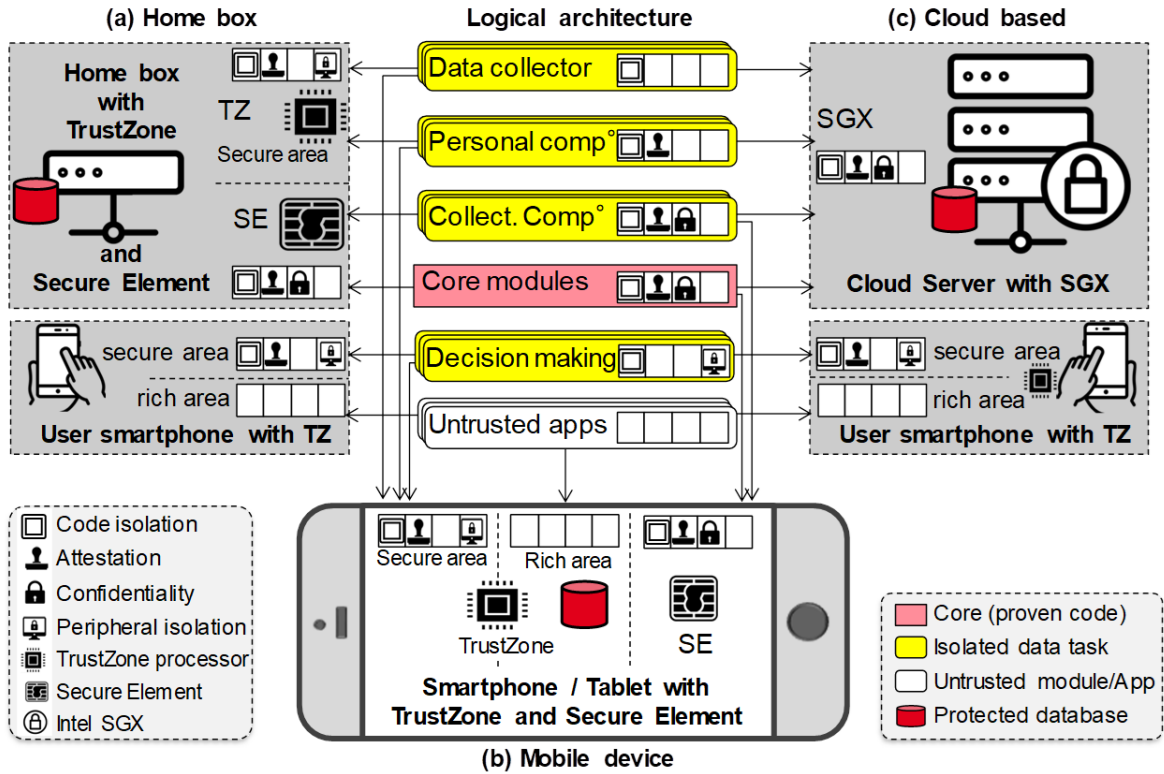


Figure 2.3: Physical ES-PDMS instances: (a) home box; (b) mobile device; (c) cloud based

2.4 Related Contributions and Future Work

In this section, we overview our contributions related to the PDMS architecture and the defined security properties. We then present some important challenges that we intend to address on the short term.

2.4.1 Scientific Contributions

Contributions related to Personal Cloud architectures. As indicated at the beginning of this chapter, its content is based on [9] (see Appendix D).

- In [9] we first review and compare the various Personal Data Management Systems (PDMS) in terms of functionality provided and targeted threats. From this analysis, we draw the definition of an extensive (combining all features) and secure (bypassing all threats) PDMS. We then propose a reference architecture for extensive and secure PDMSs meeting the given definition and security properties and discuss the major research challenges related to PDMS architectures.
- In [11] we presented a tutorial offering a global perspective of the current state of work at the confluence of two rapidly growing areas: (i) new PDMS solutions are provided to individuals to preserve their entire digital life and (ii) the emergence of Trusted Execution Environments (TEEs) changes the game in privacy-preserving data management with novel security models.

It is important to note that ES-PDMS is the result of a long reflection on privacy-centric data management architectures and as such, builds upon our previous contributions. These contributions already promoted the idea of employing data decentralization and secure hardware at the user-side as the cornerstone for a privacy-by-design architecture. Although not included in this chapter, we mention the following important related contributions:

- [8] which presents an earlier vision of implementing privacy-preserving personal data services within a decentralized and asymmetric architecture, i.e., personal data servers running on secure devices in users' hands at the edges of the internet and communicating over an untrusted cloud infrastructure.
- The tutorials in [14, 15, 16] overview techniques for secure management of personal data in this asymmetric architecture.

2.4.2 More Related Challenges and Contributions

Addressing the security properties of a PDMS raises specific challenges that go beyond the architectural considerations. In the following, we introduce two contributions which are related to implementing personal and collective computations in the PDMS context and which are further developed in the following chapters of this document. Other important challenges and future work are discussed in the next section.

Contributions related to *trusted personal computation*. The minimal trusted code base defined for the Core aims at being implemented on specific hardware used as TEE. There is an intense research agenda on data management for new and/or specialized hardware [72] and systems like Oracle M7 even start to provide hardware implementations of rich sets of database operations (SQL in silicon). The challenge here is to optimize and secure the execution of DB primitives according to the TEEs constraints.

A first challenge that we addressed was to embed Core primitives in highly constrained but highly secure hardware. As indicated in Section 2.3.4, secure elements such as tamper-resistant micro-controllers (MCUs) are very good candidates for securing a PDMS due to the strong security guarantees they provide (e.g., code isolation and confidentiality) and also due to their low cost. However, secure elements also exhibit drastic limitations in terms of computation resources (e.g., the amount of available RAM is in the order of tens of KB). This

requires revisiting the traditional data storage, indexing and query processing techniques to comply with the strong tamper-resistant hardware constraints. Thus, we present in Chapter 3 the design and implementation of a scalable full-text search engine designed for secure MCUs.

Contributions related to *trusted collective computation*. Personal cloud holders could willingly share personal data for the benefit of the community, e.g., to obtain recommendations about films, music albums or trip destinations, based on the ratings of other users that have a specific profile. A major issue in this context rises from the necessity of efficiently and securely identifying the pertinent nodes for queries. The classical solutions for indexing content addressable networks (i.e., Distributed Hash Tables [108, 122]) are efficient and scalable but have not been designed with privacy protection in mind. However, the shared data and metadata required to index a network of personal clouds is very sensitive.

In the personal cloud context, not only the shared content of a node is relevant for the search but also the profile (e.g., age, occupation, preferences, etc.) of the user owning the node. Hence, the personal metadata indexing challenge in a fully distributed system of PDMSs is threefold: (1) enable pertinent searches through distributed user profile indexing, (2) design the index and the query processing in such a way that the privacy of each contributor is protected and (3) make the decentralized query processing scalable even for very large PDMS networks (e.g., at the nationwide level). We present in Chapter 4 the design and implementation of distributed protocols which address these challenges.

Beside the two above mentioned challenges that we discuss in detail in the following two chapters, there are many other important challenges linked to the proposed PDMS architecture. We mention a few of them in Chapter 5. More architectural related challenges are also discussed in [9, 11].

Chapter 3

Scalable PDMS Search Engine with Secure Hardware

Summary. *The Personal Cloud paradigm has emerged as a solution that allows individuals to manage under their control the collection, usage and sharing of their data. However, by regaining the full control over their data, the users also inherit the burden of protecting it against all forms of attacks and abusive usages. The extensive and secure PDMS architecture aims to relieve the individual from this security task. A simplified way to implement the secure PDMS architecture is by employing a secure token (i.e., a tamper-resistant hardware device) to control all the sensitive information (e.g., encryption keys, metadata, indexes) and operations (e.g., authentication, data encryption/decryption, access control, and query processing). However, secure tokens are usually equipped with extremely low RAM but have significant Flash storage capacity (Gigabytes), which raises important barriers for embedded data management. This chapter presents an embedded search engine specifically designed for secure tokens, which applies to the important use-case of managing and securing documents in the Personal Cloud context. Conventional search engines privilege either insertion or query scalability but cannot meet both requirements at the same time. Moreover, very few solutions support data deletions and updates in this context. In this chapter, we introduce three design principles, namely Write-Once Partitioning, Linear Pipelining and Background Linear Merging, and show how they can be combined to produce an embedded search engine matching the hardware constraints of secure tokens and reconciling high insert/delete/update rate and query scalability. Our experimental results, obtained with a prototype running on a representative hardware platform, demonstrate the scalability of the approach on large datasets and its superiority compared to state-of-the-art methods.*

Contents

3.1	Context and Motivation	40
3.2	Related Work	41
3.2.1	Search Engine Requirements	41
3.2.2	Hardware Constraints of Secure Tokens	42
3.2.3	State-of-the-Art Solutions	43
3.2.4	Problem Formulation	45
3.3	Approach and Scientific Results	45
3.3.1	Design Principles	46
3.3.2	Write-Once Partitioning and Linear Pipelining	47
3.3.3	Background Linear Merging	48
3.3.4	Document Deletions	49
3.3.5	Conditional Top-k Queries	51
3.3.6	Experimental Validation	53
3.4	Related Contributions and Future Work	54
3.4.1	Scientific Contributions	54
3.4.2	Future Work	55

Chapter content and scientific publications:

- Sections 3.1 to 3.3 of this chapter overview the scientific results presented in [69] and as such are based on the content of that paper. The full version of the paper is included in Appendix E.
- Section 3.4 of this chapter browses through the other related publications.

Most of the scientific contributions presented in this chapter were obtained during the Ph.D. thesis of Saliha Lallali [67] that I co-supervised with Nicolas AnCIAUX and Philippe Pucheral. Also, some of the contributions cited in Section 3.4 were obtained during the Ph.D. thesis of Dai-Hai Ton-That [127] that I co-supervised with Karine Zeitouni. Part of this work was funded by the ANR KISS project (2011-2015).

3.1 Context and Motivation

The Personal Cloud paradigm has emerged as a way to allow individuals to manage under their control the collection, usage and sharing of their data (see Chapter 2). This user-centric vision illustrates the gravity shift of information management from organizations to individuals. However, at the time individuals recover their sovereignty of their data, they also inherit the burden of organizing this personal data space and more importantly of protecting it against all forms of attacks and abusive usages, a responsibility that they cannot endorse.

The extensive and secure PDMS architecture introduced in the previous chapter aims to relieve the individual from this security task by leveraging hardware-based trusted execution environments (see Section 2.3.4 in Chapter 2). Figure 3.1 presents a simplified version of a secure PDMS architecture¹, which combines a traditional personal home cloud server (e.g., running on a plug computer or an internet gateway at home) and a secure token (i.e., a tamper-resistant hardware device). Heterogeneous data issued by external sources and by personal appliances are all transformed into documents. Document metadata (keywords extracted from the file content, date, type, authors, tags set by the user herself, etc.) is extracted at insertion time, stored in the secure token and indexed so that the secure token can act as a privacy preserving Google Desktop or Spotlight for the user's dataspace. Documents themselves are encrypted by the secure token before being stored in the Personal Cloud, locally or remotely. Thus, the secure token plays the role of a gatekeeper for the whole Personal Cloud by managing all the sensitive information (e.g., encryption keys, metadata, indexes) and operations (e.g., authentication, data encryption/decryption, access control, and query processing) [68, 70].

In the context of the Personal Cloud, embedding a full-text search engine in a secure token will allow a user to securely search through her file collection in a simple way (i.e., using keywords), without exposing any metadata to the outside world. A file can be any form of document, mail, picture, music or video file, etc., that is associated with a set of terms. A query can be any form of keyword search using a ranking function (e.g., *tf-idf*) identifying the top-k most relevant files. Designing such embedded search engine is however very challenging. Indeed, data storage in secure tokens is usually provided by large capacity removable SD or μ SD cards or by soldered raw Flash chips while computing power is provided by microcontrollers (MCU) equipped with tiny RAM (tens of KB). This conjunction of hardware constraints raises critical issues as discussed in detail in [10]. Typically, NAND Flash badly adapts to random fine-grain updates while state-of-the-art indexing techniques either consume a lot of RAM or produce a large quantity of random fine-grain updates.

We present in this chapter an efficient and scalable search engine adapted to the highly constrained architecture of secure tokens. Section 3.2 details the search engine requirements, the secure tokens' hardware constraints, analyses the state-of-the-art solutions and derives from this analysis a precise problem statement. Section 3.3 introduces our three design principles and a summary of the proposed inverted index structure derived from these principles as well as the need for conditional top-k queries and the extension of our search engine to match such a requirement. Section 3.4 presents the related contributions and future research directions based on this work.

¹Compared with the ES-PDMS architecture introduced in the previous chapter, there are no Data Tasks in this simplified PDMS version and all the Core modules run inside the secure token.

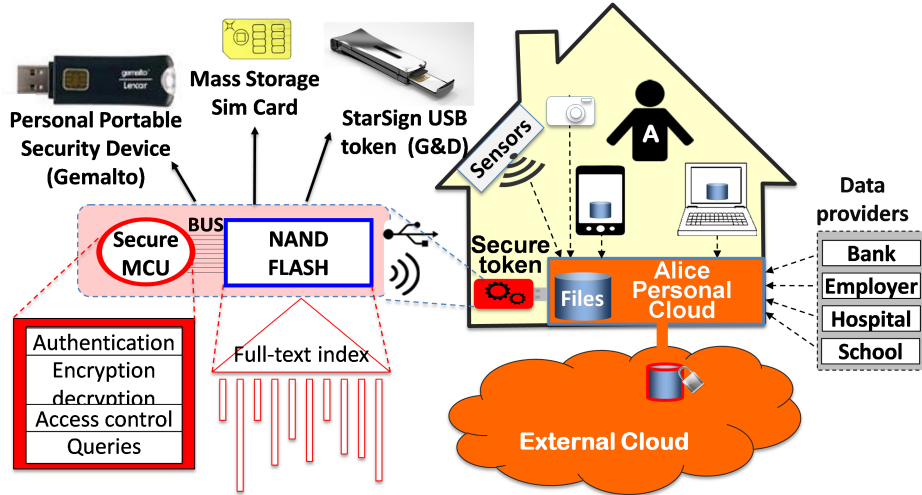


Figure 3.1: Example of a basic secure PDMS platform

3.2 Related Work

This section describes the main requirements of a full-text search engine, the hardware constraints of secure tokens, and reviews the literature addressing the problem of implementing a search engine under these constraints. Then, in the light of the existing works and their shortcomings, we precisely state the problem addressed in this chapter.

3.2.1 Search Engine Requirements

As in [124], we consider that the search engine of interest here has similar functionality as a Google Desktop embedded in secure tokens. Hence, we use the terminology introduced in the Information Retrieval literature for full-text search. Then, a document refers to any form of data files, terms refers to any forms of metadata elements, term frequencies refer to metadata element weights and a query is equivalent to a full-text search.

Full-text search has been widely studied by the information retrieval community since decades (see [149] for a recent survey). The core problem is, given a collection of documents and a user query expressed as a set of terms $\{t_i\}$, to retrieve the k most relevant documents according to a ranking function. In the wide majority of the related works, the *tf-idf* score, i.e., term frequency-inverse document frequency, is used to rank the query results. A document can be of many types (e.g., text file, image, etc.) and is associated with a set of terms (or keywords) describing its content and weights indicating their respective importance in the document. For text documents, the terms are words composing the document and their weight is their frequency in the document. For images, the terms can be tags, metadata or visterms describing image subparts [145]. For a query $Q = \{t\}$, the *tf-idf* score of each indexed document d containing at least a query term can be computed as follows: $tf-idf = \sum_{t \in Q} \log(f_{d,t} + 1) \times \log \frac{N}{F_t}$, where $f_{d,t}$ is the frequency of term t in document d , N is the total number of indexed documents, and F_t is the number of documents that contain t . This formula is given for illustrative purpose, the weight between $f_{d,t}$ and $\frac{N}{F_t}$ varying depending on the proposals.

Classically, full-text search queries are evaluated efficiently using an inverted index, named

I hereafter (see Figure 3.2). Given $D = \{d_i\}$ a set of documents, the inverted index I over D consists of two main components [149]: (i) a search structure $I.S$ (also called dictionary) which stores for each term t appearing in the documents the number F_t of documents containing t and a pointer to the inverted list of t ; (ii) a set of inverted lists $\{I.L_t\}$ where each list stores for a term t the list of $(d, f_{d,t})$ pairs where d is a document identifier in D that contains t and $f_{d,t}$ is the weight of the term t in the document d (typically the frequency of t in d). The dictionary is constituted by all the distinct terms t of the documents in D , and is large in practice, which requires organizing it into a search-efficient structure such as a B-tree.

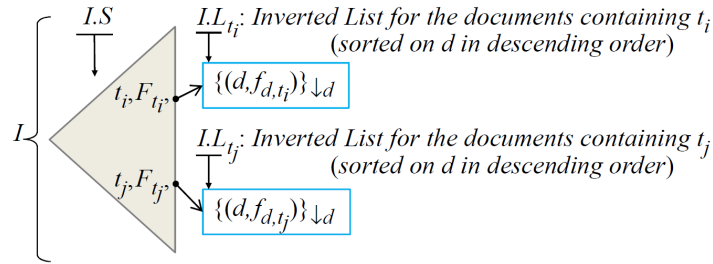


Figure 3.2: Typical inverted index structure

A query $Q = \{t\}$ is traditionally evaluated by: (i) accessing $I.S$ to retrieve for each query term t the inverted lists elements $\{I.L_t\}_{t \in Q}$; (ii) allocating in RAM one container for each unique document identifier in these lists; (iii) computing the score of each of these documents using a weight function, e.g., $tf-idf$; (iv) ranking the documents according to their score and producing the k documents with the highest scores.

3.2.2 Hardware Constraints of Secure Tokens

Whatever their form factor and usage, secure tokens share strong commonalities in terms of data management architecture. Indeed, a large NAND Flash storage is used to persistently store the data and the indexes, and a microcontroller (MCU) executes the embedded code, both being connected by a bus. Hence, the architecture inherits hardware constraints from both the MCU and the Flash memory.

The MCUs embedded in secure tokens usually have a low power CPU, a tiny RAM (few KB), and a few MB of persistent memory (ROM, NOR or EEPROM) used to store the embedded code. The NAND Flash component (either raw NAND Flash chip or SD/ μ SD card) also exhibits strong limitations. In NAND Flash, the base unit for a read and a write operation is the sector (usually 512 bytes) with raw NAND Flash chips or the page (usually 2 Kbytes or four sectors) with SD/ μ SD cards in which the access to the Flash memory is managed by a Flash Translation Layer (FTL). The sectors/pages must be erased before being rewritten but the erase operation must be performed at a block granularity (e.g., 256 pages). Erases are then costly and a block wears out after about 10^4 repeated write/erase cycles. In addition, the sectors/pages have to be written sequentially in a block. Therefore, NAND Flash badly supports random writes. We observed this same bad behavior both with raw NAND Flash chips and SD/ μ SD cards. Our own measurements [69] corroborate the ones published in [26] indicating that random writes are (much) more costly than sequential writes on SD cards. While high-end SSDs use large on-device RAM (e.g., 512MBytes) to reorder random writes and optimize their performance, secure tokens equipped with very scarce RAM and

basic NAND Flash storage cannot hide NAND Flash constraints and as such are exposed to large performance degradation. Hence, in the embedded context, random writes in Flash storage must be proscribed.

Finally, it is worth observing that given the strong similarity between the hardware architecture of secure tokens and of smart objects, we can consider that the secure tokens represent a specific instance of smart objects, i.e., smart objects having a tamper-resistant MCU. Hence, we use the terms *secure token* and *smart object* interchangeably according to this observation.

3.2.3 State-of-the-Art Solutions

Data management embedded in secure tokens [8, 132] or more generally in smart objects is no longer a new topic. Many proposals from the database community tackle this problem in the context of the Internet of Things [24], strengthening the idea that smart objects must now be considered as first-class data sources. For instance, simple query evaluation facilities have been recently proposed for sensor nodes equipped with large Flash memory [43] to enable filtering operations. Relational database operations like selection, projection and join for new generations of SIM cards with large Flash storage capacities have been proposed in [10, 137]. However, several works [10, 74, 137] consider a traditional database context and do not address the full-text search problem, leading to different query processing techniques and indexing structures. Therefore, we focus below on works specifically addressing embedded search engines and then extend the review to a few works related to Flash-based indexing when the way they tackle the MCU and Flash constraints can enlighten the discussion.

Embedded search engines. A few pioneer works demonstrate the interest of embedding search engine techniques into smart objects equipped with extended Flash storage to manage collections of files stored locally [125, 124, 140, 141, 145]. These works rely on a similar design of the embedded inverted index as proposed in [124]. Instead of maintaining one inverted list per term in the dictionary, each term is hashed to a bucket and a single inverted list is built for each bucket. The inverted lists are stored sequentially in Flash memory, within chained pages, and only a small hash table referencing the first Flash page of each bucket is kept in RAM. The number of buckets is kept small, such that (i) the large dictionary of terms (usually tens of MB) is replaced by a small hash table stored in RAM, and (ii) the main part of the RAM can be used as an insertion buffer for the inverted lists elements, i.e., $(t, d, f_{d,t})$ triples. This approach complies with a small RAM and suits well the Flash constraints by precluding fine grain random (re)writes in Flash. However, each inverted lists corresponds to a large number of different terms, which unavoidably leads to a high query evaluation cost that increases proportionally with the size of the data collection. The less RAM available, the smaller the number of hash buckets and the more severe the problem is. In addition, these techniques do not support document deletions, but only data aging mechanisms, where old index entries automatically expire when overwritten by new ones. A similar design is proposed in [145] that builds a distributed search engine to retrieve images captured by camera sensors. A local inverted index is embedded in each sensor node to retrieve the relevant images locally, before conducting the distributed search. However, this work considers powerful sensors nodes (with tens of MB of local RAM) equipped with custom SD card boards (with specific performance). At the same time, the underlying index structure is based on inverted lists organized in a similar way as in [124]. All these methods are highly efficient for document insertions, but fail to provide scalable query processing for large collections of documents. Therefore, their

usage is limited to applications that require storing only a small number (few hundreds) of documents.

B-tree indexing in NAND Flash. In the database context, adapting the B-tree to NAND Flash has received a great attention. Indeed, the B-tree is a very popular index and its standard implementation performs poorly in Flash [144]. Many proposals (e.g., [2, 75, 144]) tackle this problem. The key idea in these approaches is to buffer the updates in log structures that are written sequentially and to leverage the fast (random) read performance of Flash memory to compensate the loss of optimality of the lookups. When the log is large enough, the updates are committed into the B-tree in a batch mode, to amortize the Flash write cost. The log must be indexed in RAM to ensure performance. The different proposals vary in the way the log and the in-memory index are managed, and in the impact it has on the commit frequency. To amortize the write cost by a significant factor, the log must be seldom committed, which requires more RAM. Conversely, limiting the RAM size leads to increasing the commit frequency, thus generating more random writes. The RAM consumption and the random write cost are thus conflicting parameters. Under severe RAM limitations, the gain on random writes definitely vanishes.

Partitioned indexes. In another line of work, partitioned indexes have been extensively employed especially to improve the storage performance in environments with insert-intensive workloads and concurrent queries on magnetic disks. A prominent example is the LSM-tree (i.e., the Log-Structured Merge-tree) [95] [30] and its many variants (e.g., the Partitioned Exponential file [56] and the bLSM-tree [95] to name but a few). The LSM-tree consists in one in-memory B-tree component to buffer the updates and one on-disk B⁺-tree component that indexes the disk resident data. Periodically, the two components are merged to integrate the in-memory data and free the memory. The benefit of such an approach is twofold. First the updates are integrated in batch, which amortizes the write cost per update. Second, the merge operation uses sequential I/Os, which reduces the disk arm movements and thus, highly increases the throughput. If the indexed dataset becomes too large, the index disk component can be divided into several disk components of exponentially increasing size to reduce the write amplification of merges. Many works have proposed optimized versions of the LSM-tree. For instance, bLSM [119] fixes several limitations of the LSM-tree. Among the improvements, the main contribution is an advanced merge scheduler that bounds the index write latency without impacting its throughput. Also, the FD-tree [75] proposes a similar structure with the LSM-tree to optimize the data indexing on SSDs. Furthermore, the storage systems of the major web service provider, e.g., Google's Bigtable and Facebook's Cassandra, employ a similar partitioning approach to implement key-value stores. The idea is to buffer large amounts of updates in RAM and then flush them in block on disk as a new partition. Periodically, the small partitions are merged into a large partition.

The proposed search engine shares the general idea of index partitioning and merging with the above mentioned works. However, the similarity stops at the general level since the specific hardware constrains and type of queries in our context cannot be satisfied by the existing solutions. In particular, the small amount of RAM requires frequent flushes of the buffered updates. This leads to a specific organization of the partitions and merge scheduling in our structure. The type of query in our context (i.e., top-k keyword search) represents another major difference with the existing partitioning methods that only consider the classical key-value search. To be able to evaluate full text search queries in the presence of deletions and limited amount of RAM, our search engine proposes a novel index organization with a specific

query processing.

In general, designing access methods is often a matter of tradeoff between minimizing read times, update cost and memory/storage overhead as observed in [18]. Given the specific architecture of secure tokens, this tradeoff translates to a tension between memory and Flash storage in the embedded context [3]. Specifically, tiny RAM and NAND Flash persistent storage introduce conflicting constraints and lead to split state of the art solutions in two families. The *insert-optimized family* reaches insertion scalability thanks to a small indexed structure buffered in RAM and sequentially flushed in Flash, thereby precluding costly random writes in Flash. This good insertion behavior is however obtained to the detriment of query scalability, the performance of searches being roughly linear with the index size in Flash. Conversely, the *query-optimized family* reaches query scalability by adapting traditional indexing structures to Flash storage, to the detriment of insertion scalability, the number of random (re)writes in Flash (linked to the log commit frequency) being roughly inversely proportional to the RAM capacity. In addition, we are not aware of works addressing the crucial problem of random document deletions in the context of an embedded search engine.

3.2.4 Problem Formulation

In the light of the preceding sections, the problem addressed in this paper can be formulated as *designing an embedded full-text search engine that has the following two properties*:

- *Bounded RAM agreement*: the proposed engine must be able to respect a predefined RAM consumption bound (RAM_Bound), precluding any solution where this consumption depends on the size of the document set.
- *Full scalability*: the proposed engine must be scalable for queries and updates (insertion, deletion of documents) without distinction.

The Bounded RAM agreement is required to comply with the widest population of secure tokens. The consequence is that the full-text search engine must remain functional even when very little RAM (a few KB) is made available to it. Note that the RAM_Bound size is a subpart of the total physical RAM capacity of a secure token considering that the RAM resource is shared by all software components running in parallel on the platform, including the operating system. The RAM_Bound property is also mandatory in a co-design perspective where the hardware resources of a given platform must be precisely calibrated to match the requirements of a particular application domain.

The Full scalability property guarantees the generality of the approach. By avoiding to privilege a particular workload, the index can comply with most applications and data sets. To achieve update scalability, the index maintenance needs to be processed without generating random writes, which are badly supported by the Flash memory. At the same time, achieving query scalability means obtaining query execution costs in the same order of magnitude with the ideal query costs provided by a classical inverted index I .

3.3 Approach and Scientific Results

Satisfying the Bounded RAM agreement and Full scalability properties simultaneously is challenging, considering the conflicting MCU and Flash constraints mentioned above. To tackle this challenge, we propose an indexing method that relies on the following three design principles.

3.3.1 Design Principles

P1. Write-Once Partitioning: *Split the inverted index structure I in successive partitions such that a partition is flushed only once in Flash and is never updated.*

By precluding random writes in Flash, Write-Once Partitioning aims at satisfying update scalability. Considering the Bounded RAM agreement, the consequence of this principle is to parse documents and maintain I in a streaming way. Conceptually, each partition can be seen as the result of indexing a window of the document input flow, the size of which is limited by the `RAM_Bound`. Therefore, I is split in an infinite sequence of partitions $\langle I_0, I_1, \dots, I_p \rangle$, each partition I_i having the same internal structure as I . When the size of the current I_i partition stored in RAM reaches `RAM_Bound`, I_i is flushed in Flash and a new partition I_{i+1} is initialized in RAM for the next window.

A second consequence of this design principle is that document deletions have to be processed similar to document insertions since the partitions cannot be modified once they are written. This means adding compensating information in each partition that will be considered by the query process to produce correct results.

P2. Linear Pipelining: *Compute each query Q with respect to the Bounded RAM agreement in such a way that the execution cost of Q over $\langle I_0, I_1, \dots, I_p \rangle$ is in the same order of magnitude as the execution cost of Q over I .*

Linear Pipelining aims at satisfying query scalability under the Bounded RAM agreement. A unique structure I as the one pictured in Figure 3.2 is assumed to satisfy query scalability by nature and is considered hereafter as providing a lower bound in terms of query execution time. Hence, the objective of Linear pipelining is to keep the performance gap between Q over $\langle I_0, I_1, \dots, I_p \rangle$ and Q over I , both small and predictable (bounded by a given tuning parameter). Computing Q as a set-oriented composition of a set of Q_i over I_i , (with $i = 0, \dots, p$) would unavoidably violate the Bounded RAM agreement as p increases, since it will require to store all Q_i 's intermediate results in RAM. Hence the necessity to organize the processing in pipeline such that the RAM consumption remains independent of p , and therefore of the number of indexed documents. Also, the term linear pipelining conveys the idea that the query processing must preclude any iteration (i.e., repeated accesses) over the same data structure to reach the expected level of performance. This disqualifies brute-force pipeline solutions where the *tf-idf* scores of documents are computed one after the other, at the price of reading the same inverted lists as many times as the number of documents they contain.

However, Linear Pipelining alone cannot prevent the performance gap between Q over $\langle I_0, I_1, \dots, I_p \rangle$ and Q over I to increase with the increase of p as (i) multiple searches in several small $I_i.S$ are more costly than a single search in a large $I.S$ and (ii) the inverted lists in $\langle I_0, I_1, \dots, I_p \rangle$ are likely to occupy only fractions of Flash pages, multiplying the number of Flash I/Os to access the same amount of data. A third design principle is then required.

P3. Background Linear Merging: *To limit the total number of partitions, periodically merge partitions in a way compliant with the Bounded RAM agreement and without hurting update scalability.*

The objective of partition merging is therefore to obtain a lower number of larger partitions to avoid the drawbacks mentioned above. Partition merging must meet three requirements. First the merge must be performed in pipeline to comply with the Bounded RAM agreement. Second, since its cost can be significant (i.e., proportional to the total size of the merged partitions), the merge must be processed in background to avoid locking the index structure for

unbounded periods of time. Since multi-threading is not supported by the targeted platforms, background processing can simply be understood as the capacity to interrupt and recover the merging process at any time. Third, update scalability requires that the total cost of a merge run be always smaller than the time to fill out the next bunch of partitions to be merged.

Taken together, principles P1 to P3 reconcile the Bounded RAM agreement and Full scalability index properties. An overview of the technical solutions to implement these three principles is presented in the next sections.

3.3.2 Write-Once Partitioning and Linear Pipelining

These two design principles are discussed together because the complexity comes from their combination. Indeed, Write-Once Partitioning is straightforward on its own. It simply consists in splitting I in a sequence $\langle I_0, I_1, \dots, I_p \rangle$ of small indexes called partitions, each one having a size bounded by `RAM_Bound`. The difficulty is to implement a linear pipeline execution of any query Q on this sequence of partial indexes.

Executing Q over I would lead to evaluate: $Top_k \left[\sum_{t \in Q} W \left(f_{d,t}, \frac{N}{F_t} \right) \right]$, with $d \in D$, where Top_k selects the k documents $d \in D$ having the largest *tf-idf* scores, each score being computed as the sum, for all terms $t \in Q$, of a given weight function W taking as parameter the frequency $f_{d,t}$ of t in d and the inverse document frequency N/F_t . Our objective is to remain agnostic regarding W and then let the precise form of this function open. Let us now consider how each term of this expression can be evaluated by a linear pipelining process on a sequence $\langle I_0, I_1, \dots, I_p \rangle$.

Computing N , F_t and $f_{d,t}$. We assume that the number of documents is a global metadata maintained at insertion/deletion time and needs not be recomputed for each Q . F_t should be computed only once for each term t since F_t is constant for Q . This is why F_t is usually materialized in the dictionary part of the index ($\{t, F_t\} \subset I.S$), as shown in Figure 3.2. When I is split in $\langle I_0, I_1, \dots, I_p \rangle$, the global value of F_t should be computed as the sum of the local F_t of all partitions. To this end, during the F_t computation phase, the dictionary of each partition is read only once and the RAM consumption sums up to one buffer to read each dictionary, page by page, and one RAM variable to store the current value of each F_t . Also, the pointers referencing the inverted lists for each t are actually stored in the dictionary which has already been read while computing F_t . According to the Linear pipelining principle, we avoid reading again the dictionary by storing these pointers in RAM during the F_t computation. The extra RAM consumption is minimal and bounded by the fact that the number of partitions is itself bounded thanks to the merging process (see next section). Finally, each $f_{d,t}$ value is stored in the inverted lists together with the respective document id.

Computing Top_k . Traditionally, a RAM variable is allocated to each document d to compute its *tf-idf* score by summing the results of $W(f_{d,t}, N/F_t)$ for all terms $t \in Q$ [149]. Then, the k best scores are selected. Unfortunately, this approach conflicts with the Bounded RAM agreement since the size of the document set is likely to be much larger than the available RAM. Hence, we organize the query processing in a pure pipeline way, allocating a RAM variable only to the k documents having currently the best scores. This forces the complete computation of *tf-idf*(d) to be done for each d , one after the other. To meet this requirement while precluding any iteration on the inverted lists, these lists are maintained sorted on the document id. Note that if document ids reflect the insertion ordering, the inverted lists are naturally sorted. Hence, the *tf-idf* computation sums up to a simple linear pipeline merging

process of the inverted lists for all terms $t \in Q$ in each partition (see Figure 3.3). The RAM consumption for this phase is therefore restricted to one variable for each of the current k best $tf-idf$ scores and to one buffer (i.e., a RAM page) per query term t to read the corresponding inverted lists $I_i.L_t$ (i.e., $I_i.L_t$ are read in parallel for all t , the inverted lists for the same t being read in sequence). Figure 3.3 summarizes the data structures maintained in RAM and in Flash to handle this computation.

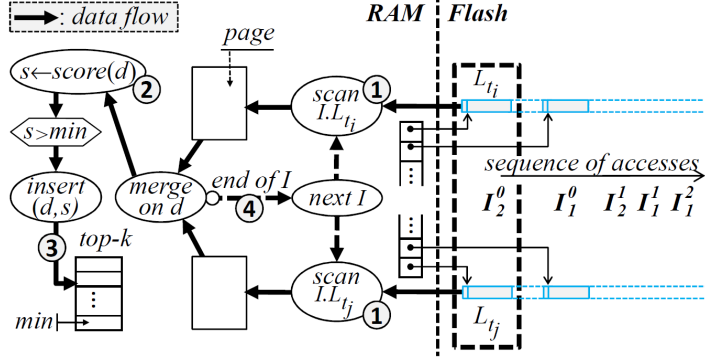


Figure 3.3: Linear Pipeline computation of Q over terms t_i and t_j

Note that another complexity comes from the fact that the same document d may cross several partitions, which requires specific metadata to be stored in the partitions to allow for correct computation of F_t and $f_{d,t}$ while still remaining compliant to the Bounded RAM agreement (please refer to [69] for the details).

3.3.3 Background Linear Merging

The background merging process aims at achieving scalable query costs by timely merging several small indexes into a larger index structure. As mentioned in Section 3.3.1, the merge must be a pipeline process in order to comply with the Bounded RAM agreement while keeping a cost compatible with the update rate. Moreover, the query processing should continue to be executed in Linear Pipelining (see Section 3.3.2) on the structure resulting from the successive merges. Therefore, the merges have to preserve the global ordering of the document ids within the index structures.

To meet these requirements, we introduce a Sequential and Scalable Flash structure, called *SSF*, pictured in Figure 3.4. *SSF* consists in a hierarchy of partitions of exponentially increasing size. Specifically, each new index partition is flushed from RAM into the first level of *SSF*, i.e., L_0 . The merge operation is triggered automatically when the number of partitions in a level becomes b , the branching factor of *SSF*, which is a predefined index parameter. The merge combines the b partitions at level L_i of *SSF*, denoted by I_1^i, \dots, I_b^i , into a new partition at level L_{i+1} , denoted by I_j^{i+1} and then reclaims all partitions at level L_i .

The merge is directly processed in pipeline as a multi-way merge of all partitions at the same level. This is possible since the dictionaries of all the partitions are already sorted on terms, while the inverted lists in each partition are also sorted on document ids. So are the dictionary and the inverted lists of the resulting partition at the upper level. More precisely, the algorithm works in two steps. In the first step, the $I.L$ part of the output partition is produced. Given b partitions in the index level L_i , $b + 1$ RAM pages are necessary to process the merge in linear pipeline: b pages to merge the inverted lists in $I.L$ of all b partitions and one

page to produce the output. The indexed terms are treated one after the other in alphabetic order. For each term t , the head of its inverted lists in each partition is loaded in RAM. These lists are then consumed in pipeline by a multi-way merge. Document ids are encountered in descending order in each list and the output list resulting from the merge is produced in the same order. The $I.S$ tree is built from the leaves to the root. This step requires one RAM page to scan $I.L$, plus one RAM page per $I.S$ tree level. This Background Merging process generates only sequential writes in Flash and previous partitions are reclaimed in large blocks after the merge. This pipeline process sequentially scans each partition only once and produces the resulting partition also sequentially. Hence, assuming $b + 1$ is strictly lower than RAM_bound , one RAM buffer (of one page) can be allocated to read each partition and the merge is I/O optimal. If b is larger than RAM_bound , the algorithm remains unchanged but its I/O cost increases since each partition will be read by page fragments rather than by full pages.

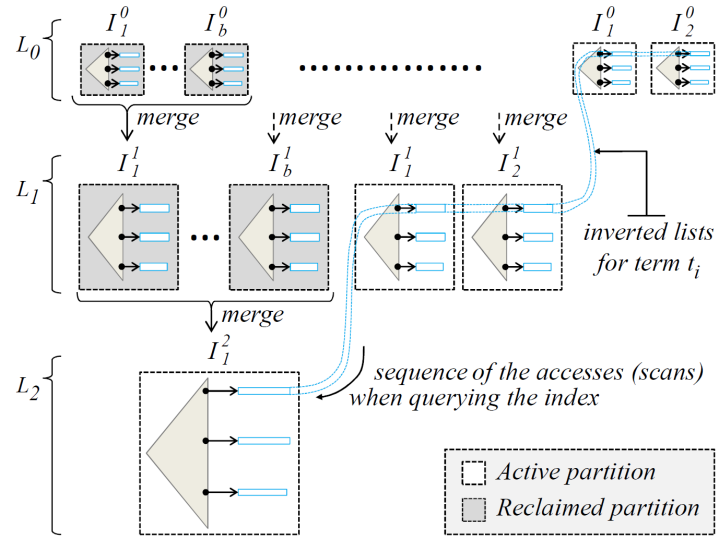


Figure 3.4: The Scalable and Sequential Flash structure

SSF provides scalable query costs since the amount of indexed documents grows exponentially with the number of levels, while the number of partitions increases only linearly with the number of levels. Note that merges in the upper levels are exponentially rare (one merge in level L_i for b^i merges in L_0) but also exponentially costly. To mitigate this problem, we perform the merge operations in background (i.e., in a non-blocking manner) (see [69] for the details).

3.3.4 Document Deletions

Implementing the delete operation is challenging, mainly because of the Flash memory constraints which proscribe the straightforward approach of updating in-place the inverted index. The alternative to updating in-place is compensation, i.e., the deleted documents' identifiers (*DDIs*) are stored in an appropriate way and used as a filter to eliminate the ghost documents retrieved by the query evaluation process.

A basic solution could be to organize the *DDIs* as a sorted list in Flash and to intersect this list at query execution time with the inverted lists in the *SSF* corresponding to the query

terms. However, this solution raises several problems. First, the documents are deleted in random order, according to users' and application decisions. Hence, maintaining a sorted list of *DDIs* in Flash would violate the Write-Once Partitioning principle since the list has to be rewritten each time a set (e.g., a page) of new *DDIs* is flushed from RAM. Second, the computation of the F_t for each query term t during the first step of the query processing cannot longer be achieved without an additional merge operation to subtract the sorted list of *DDIs* from the inverted lists of the *SSF*. Third, the full *DDI* list has to be scanned for each query regardless of the query selectivity. These two last elements make the query cost dependent of the total number of deleted documents and then conflict with the Linear pipelining principle.

Therefore, instead of compensating the query evaluation process, we propose a solution based on compensating the indexing structure itself. In particular, a document deletion is treated similarly to a document insertion, i.e., by re-inserting the metadata (terms and frequencies) of all deleted documents in the *SSF*. The objective is threefold: (i) to be able to compute, as presented in Section 3.3.2, the F_t for each term t of a query based on the metadata only (of both existing and deleted documents), (ii) to have a query performance that depends on the query selectivity (i.e., number of inserted and deleted documents relevant to the query) and not on the total number of deleted documents and (iii) to effectively purge the indexing structure from the largest part of the deleted documents at Background Merging time, while remaining compliant with the Linear Pipelining principle.

Computing Top_k in the presence of deletions. A document deletion is treated similarly to a document insertion. Assuming a document d is deleted in the time window corresponding to a partition I_i , a pair $(d, -f_{d,t})$ is inserted in each list $I_i.L_t$ for the terms t present in d and the F_t value associated to t is decremented by 1 to compensate the prior insertion of that document. To distinguish between an insertion and a deletion, the frequency value $f_{d,t}$ for the deleted document id is simply stored as a negative value, i.e., $-f_{d,t}$.

Integrating deleted documents makes the computation of Top_k more subtle. Following the Linear Pipelining principle, the *tf-idf* scores of all documents are computed one after the other, in descending order of the document ids, thanks to a linear pipeline merging of the insert lists associated to the queried terms. To this end, the algorithm introduced in Section 3.3.2 uses k RAM variables to maintain the current k best *tf-idf* scores and one buffer (i.e., a RAM page) per query term t to read the corresponding inverted lists. Some elements present in the inverted lists correspond actually to deleted documents and must be filtered out. The problem comes from the fact that documents are deleted in random order. Hence, while inverted lists are sorted with respect to the insertion order of documents, a pair of the form $(d, -f_{d,t})$ may appear anywhere in the lists. In case a document d has been deleted, the unique guarantee is to encounter the pair $(d, -f_{d,t})$ before the pair $(d, f_{d,t})$ if the traversal of the lists follows a descending order of the document ids. However, maintaining in RAM the list of all encountered deleted documents in order to filter them out during the follow-up of the query processing would violate the Bounded RAM agreement.

The proposed solution works as follows. The *tf-idf* score of each document d is computed by considering the modulus of the frequencies values $|\pm f_{d,t}|$ in the *tf-idf* score computation, regardless of whether d is a deleted document or not. Two lists are maintained in RAM: $Top_k = \{(d, score(d))\}$ contains the current k best *tf-idf* scores of documents which exist with certainty (no deletion has been encountered for these documents); $Ghost = \{(d, score(d))\}$ contains the list of documents which have been deleted (a pair $(d, -f_{d,t})$ has been encountered while scanning the inverted lists) and have a score better than the smallest score in Top_k . Top_k and $Ghost$ lists are managed as follows. If the score of the current document d is worse

than the smallest score in Top_k , it is simply discarded and the next document is considered (step 2 in Figure 3.5). Otherwise, two cases must be distinguished. If d is a deleted document (a pair $(d, -f_{d,t})$ is encountered), then it enters the *Ghost* list (step 3); else it enters the Top_k list unless its id is already present in the *Ghost* list (step 4). Note that this latter case may occur only if the id of d is smaller than the largest id in *Ghost*, making the search in *Ghost* useless in many cases. An important remark is that the *Ghost* list has to register only the deleted documents which may compete with the k best documents, to filter them out when these documents are later encountered, which makes this list very small in practice, i.e., approximately $\delta \times k \times (\ln n + \epsilon)$ [69], where δ is the percentage of deleted documents, n is the total number of indexed documents and ϵ is a small value.

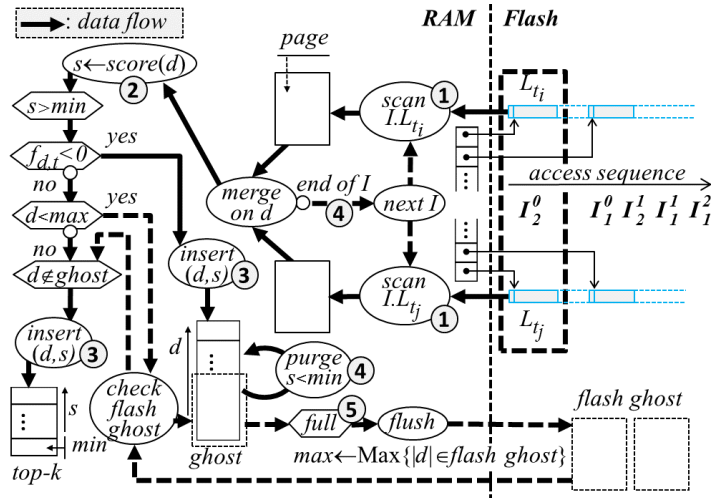


Figure 3.5: Linear pipeline computation of Q in the presence of deletions

Note that the introduction of deletions has actually a marginal impact on the merge operation, which continues to be efficiently processed in linear pipeline as before. Besides, when inverted lists are merged during the Background Merging, a new particular case may occur, that is when two pairs $(d, f_{d,t})$ and $(d, -f_{d,t})$ are encountered in separate lists for the same d . This means that document d has actually been deleted. d is then purged (the document deletion is absorbed) and will not appear in the output partition. Hence, the merge plays a supplementary role of absorbing the data deletions.

3.3.5 Conditional Top-k Queries

In this section, we discuss the extension of the proposed search engine to support *conditional top-k queries*, i.e., top-k queries combined with conditions expressed over documents' metadata. Specifically, we present two major use-cases in the context of the Personal Cloud, i.e., keyword search combined with file metadata and tag-based access control, and explain how the SSF data structure and algorithms can be extended to support it in a natural way.

Combining keyword search and file metadata search. Similar to the classical inverted index, the base implementation of our search engine does not make any distinction between the terms extracted from the file content and the file metadata terms. Hence, the score of a document is computed using the classical *tf-idf* formula (see Section 3.2.1) regardless if the query terms are content keywords or metadata terms in the document (e.g., creation

date, filename, file type and extension, tags set by the user herself, etc.).

Nonetheless, from a semantic point of view, it makes sense to separate the content terms from the metadata terms in the query evaluation like in the existing file search engine implementations (e.g., Google desktop or Spotlight). The idea is that the query terms are matched only with the content terms of the indexed documents, while users can specify additional constraints regarding the document metadata. For instance, a user can combine the query terms “research meeting Bordeaux” with the constraints “file type = pptx or (file type = mail and sender = Bob)”. In this case, the query results will only consist of documents having the extension “pptx” or documents of type “mail” sent by “Bob”, which contain at least one term among “research meeting Bordeaux” and ranked based on the weight of these three words in the documents. In general, the metadata search consists in one or several metadata terms that are combined by AND/OR to form a logical expression, i.e., disjunctions of conjunctions of metadata terms. Formally, a metadata search rule is a logical expression: $R_{metadata} = \bigvee_j (\bigwedge_i t_{i,j}^{pred})$, where each $t_{i,j}^{pred}$ is a metadata term predicate which for a given document d is evaluated true only if the metadata term $t_{i,j}$ is associated with d . For each document matching the query content terms, the logical expression on the metadata terms is evaluated and the document is considered in the query results only if the metadata expression is evaluated true.

Tag-based access control. The owner of a personal cloud might want to share some of her documents with other users or applications. To this end, she needs to customize the sharing of her documents by adding a personalized access control (AC) policy for each user/application accessing her personal cloud. She also wants to be sure that the personal cloud is able to securely enforce the defined policies. The latter is achieved by integrating the access control engine within the secure token together with the search engine as depicted in Figure 1. The definition of the AC policies depends on the employed AC model. Access control is a well-established topic in the databases field. However, traditional AC models like MAC (Mandatory Access Control), DAC (Discretionary Access Control) or R-BAC (Role-Based Access Control) are less suitable for the Personal Cloud paradigm than Tag-Based Access Control (TBAC) models. Several works [146, 64, 84, 85, 86] consider TBAC models, due to their simplicity, for non-technical computer users that require to define access control policies over their personal data.

TBAC can be easily integrated with our inverted index if we consider that (1) an appropriate set of access control terms can be inserted to tag the documents at insertion, and (2) these access terms can be used at query processing time to evaluate logical expressions leading to discard or not each document from the query results. Let us consider a simple example. Bob is a friend of Alice with whom she shares the passion for jazz music. Bob asks Alice to share with him her jazz music collection. A collection rule has been settled such that any music file inserted in the personal cloud of Alice is associated with the access terms “music” and “<music_style>” where the value of music style is derived from the domain of music styles (e.g., “country”, “variety”, “jazz”, ...). Alice may define a permission rule for user Bob in her Personal Cloud: “Bob: music \vee jazz”. Thus, Bob can query all the documents in the Alice’s server that contain both the access term “music” and “jazz”. Alice herself needs to access her email archive over the last two years from her smartphone, but does not want that the rest of her personal data space to be accessible from her smartphone. Therefore, assuming a collection rule associates any email file with the access terms “email” and “<date>”, Alice may define the rule “Alice_smartphone: (email \vee 2019) \wedge (email \vee 2020)”.

Note that the logical expression used to define the AC policies has the same format (i.e.,

disjunctions of conjunctions) as the document metadata logical expression that can be used in the scenario that combines keyword search with file metadata search describe above. Hence, these two scenarios require the same modifications to be integrated in the proposed search engine.

Modification of the top-k scoring function. Conditional top-k queries require to evaluate a logical expression for all the candidate documents for the query results, i.e., documents having a *tf-idf* score high enough to enter the top-k results. Executing a query Q in this context is equivalent to evaluate the following function: $Top_k \left[Filter(le, T_d) \times \sum_{t \in Q} W \left(f_{d,t}, \frac{N}{F_t} \right) \right]$, with $d \in D$.

Compared with the initial top-k scoring function defined in Section 3.3.2, the only difference is the appearance of the $Filter()$ element. $Filter$ is a function that takes as input a logical expression le (e.g., a TBAC policy or a file metadata expression) and the list of metadata terms T_d of a document. The function returns 1 if le is true on T_d and 0 otherwise. In other words, the *tf-idf* score of the documents whose metadata verify the logical expression remains unchanged, whereas the *tf-idf* score is set to 0 for the rest of the documents, which are thus eliminated from the query results. The reader interested can refer to [69] for the detailed implementation of the linear pipeline computation of conditional top-k queries.

3.3.6 Experimental Validation

We validated our design in two complementary ways. First, we did a precise analysis of the RAM consumption of each algorithm and demonstrate that each satisfies the Bounded RAM agreement. Second, we conducted a comprehensive set of experiments on a real representative secure token platform, using three real and synthetic datasets, and showed that query and insertion/deletion/update performance can be met together demonstrating Full scalability compliance. The reader interested can find in [69] the complete set of experimental results. Nonetheless, we can list the following highlights and draw the following conclusions from the implementation and its performance analysis.

All the experiments have been conducted on a development board ST3221G-EVAL equipped with the MCU STM32F217IG connected to a MicroSD card slot. This hardware configuration is representative of typical secure tokens [10, 12, 70, 132] or smart objects [124, 137]. The search engine code is stored on the internal NOR Flash memory of the MCU, while the inverted index is stored on a MicroSD NAND Flash card. We tested our index structure with several commercial MicroSD cards and selected two representative MicroSD cards (i.e., Kingston MicroSDHC Class 10 4GB and Silicon Power SDHC Class 10 4GB) exhibiting different performance as measured on our development board. The MCU has 128KB of available RAM. However, the *search engine only uses a maximum amount of 5KB of RAM*, to validate our design whatever the available RAM of existing secure tokens and whatever the fragment of this RAM allocated to the search engine running in competition with other embedded software.

We compare our proposed search engine with the representative indexing method of each class of approaches (see Section 3.2.3). Hence, we choose the typical inverted index [149] to represent the query-optimized index family (see Figure 3.2) and Microsearch [124] for the insert-optimized index family.

The tiny RAM and the NAND Flash storage of sensors introduce conflicting constraints on the index structure. Our solution is the only one to offer both query and update scalability under these constraints by balancing the query and the update costs for any kind of document

collection. At the same time, the loss in both the query and the update performance remains reasonable compared with the query optimized index (i.e., the Inverted Index) and the insert optimized index (i.e., Microsearch). In most cases, SSF has (much) better throughput with both insert- and query-oriented workloads, while being the sole versatile method. Our results show that the speedup of SSF (i.e., the ratio between the throughput of SSF and of the competitors) is generally high (i.e., from one to three orders of magnitude greater) for workloads containing insertions and queries in different ratios. Practically, SSF will be the preferred index method unless the expected workload contains in an overwhelming proportion either insertions or queries.

3.4 Related Contributions and Future Work

In this section, we overview our contributions related to data management within secure tokens and discuss some future work stemming from the proposed embedded search engine.

3.4.1 Scientific Contributions

Contributions related to full-text search embedded in secure tokens. As indicated at the beginning of this chapter, its content is based on [69] (see Appendix E), which itself is based on [12] and the prototype implementations presented in [68, 70].

- In [12] we present a search engine designed for smart objects. We introduce the three design principles, namely Write-Once Partitioning, Linear Pipelining and Background Linear Merging, and show how they can be combined to produce an embedded search engine reconciling high insert/delete/update rate and query scalability. This first work gives the general description of the embedded index structure and index maintenance operations with a focus on smart object scenarios.
- In [69] we extended the work in [12] to cope with the important case of the secure Personal Cloud. First, we provided all algorithms underlying our search engine, some of them being non trivial, and perform a thorough analysis of their RAM consumption to demonstrate the compliance of our design with the Bounded RAM agreement. Second, we extended the performance measurements with real datasets representative of the personal Cloud context and more generally of any context manipulating rich documents with random updates. We also extended the type of situations measured in order to demonstrate the compliance of our design with the Full scalability property. Third, we propose an extension of our work to support conditional top-k queries which are most relevant in the Personal Cloud context.
- A first operational prototype implementation of the search engine has been demonstrated in [68]. This demonstration employs conditional top-k queries (i.e., access control policies are defined using TBAC) to show the security, the bounded RAM and full scalability (for both queries and updates) properties of the proposed embedded search engine.
- The search engine has been employed in a second prototype implementation which was demonstrated in [70]. In this demonstration, we extended the previous work to provide a *secure distributed search engine*, such that applications can query securely the documents stored in a large number of PDMSs. The demonstration shows

that practical and efficient solutions can be devised to evaluate distributed searches within large communities of Personal Cloud users with a very limited privacy risk for the participants based on “gossip” based computations.

It is important to note that privacy-preserving data management based on constrained secure hardware has been an important research topic for my research group and myself. As such, the work presented in this chapter had an impact on and has been influenced by other research contributions. Thus, we also mention here the contributions on privacy-preserving mobile participatory sensing using secure hardware [130, 101, 128] as well as the work on trajectory indexing on Flash storage [129] (see more details in Section 1.2 in Chapter 1).

3.4.2 Future Work

Generalization to other indexing methods. The three design principles that we proposed can be generalized to other kinds of embedded query engines (e.g., NoSQL-like, B-tree, R-tree, ...) considering that indexing any form of data streams in secure tokens will encounter similar hardware constraints and then lead to similar requirements. Hence, we expect this contribution to lay the foundation for efficient and scalable data management with constrained hardware, which is an important challenge in the ES-PDMS context or more generally, to the smart objects context.

Advanced secure sharing models. The proposed search engine integrates a basic tag-based access control policy evaluation through the conditional top-k query processing. This allows for an efficient and scalable AC rule evaluation. However, the AC model considered in our work is quite basic and limited. Typically, the N and F_t values that are used in the *tf-idf* document score computation are determined based on the complete set of indexed documents. In some cases, these values should be computed based on the number of documents that are eligible for the application that access them and we plan to study this issue as future work. In addition, more advanced access control models may be pertinent in the PDMS context and thus should be supported by the embedded query engine.

Chapter 4

Preserving Data Confidentiality in Fully-Decentralized PDMS Systems

Summary. *The advent of Personal Data Management Systems allows individuals to integrate all their personal data in a single place and use it for their benefit and for the benefit of the community. This leads to a significant paradigm shift since de facto personal data become massively distributed. In this context, an important issue needed to be addressed is: how can users/applications execute queries and computations over this massively distributed data in a secure and efficient way, relying exclusively on peer-to-peer (P2P) interactions? In this chapter, we motivate and study the feasibility of such a pure P2P personal data management system and provide efficient and scalable mechanisms to reduce the data leakage to its minimum with covert adversaries. In particular, we introduce the different attacks possible in this context and formulate the necessary security requirements to circumvent them. Then, we focus on the important security requirement related to the verifiable random actor selection. We show that data processing tasks can be assigned to nodes in a verifiable random way, which cannot be influenced by malicious colluding nodes. We propose a generic solution which largely minimizes the verification cost. Our experimental evaluation shows that the proposed protocols lead to minimal private information leakage, while the cost of the security mechanisms remains very low even with a large number of colluding corrupted nodes.*

Contents

4.1	Context and Motivation	59
4.2	Related Work	60
4.3	Approach and Scientific Results	62
4.3.1	SEP2P Architectural Design and Threat Models	62
4.3.2	SEP2P Security Requirements	64
4.3.3	Verifiable Random Actor Selection	66
4.3.4	Experimental Validation	73
4.4	Related Contributions and Future Work	74
4.4.1	Scientific Contributions	74
4.4.2	Future Work	74

Chapter content and scientific publications:

- Sections 4.1 to 4.3 of this chapter overview the scientific results presented in [79] and as such are based on the content of that paper. The full version of the paper is included in Appendix F.
- Section 4.4 of this chapter browses through the other related publications.

Most of the scientific contributions presented in this chapter were obtained during the Ph.D. thesis [76] and one year post-doc of Julien Loudet that I co-supervised with Luc Bouganim. The Ph.D. thesis of Julien was founded as a Cifre thesis and achieved in collaboration with the Cozy Cloud company. Part of this work was supported by the ANR PerSoCloud project (2017-2021).

4.1 Context and Motivation

With the advent of Personal Data Management Systems (PDMSs) (see Chapter 2), users can leverage the power of their PDMS to benefit from their personal data for their own good and in the interest of the community. Thus, the PDMS paradigm holds the promise of unlocking new innovative usages. Let us consider three emblematic *distributed applications* based on large user communities which could greatly benefit from the PDMS paradigm: (1) mobile participatory sensing apps [130], in which mobile users produce sensed geo-localized data (e.g., traffic, air quality, noise, health conditions) to compute spatially aggregated statistics benefiting the whole community; (2) subscription-based or profile-based data diffusion apps [134], in which PDMS users provide preferences or exhibit profiles in order to selectively receive pertinent information; and (3) distributed query processing over the personal data of large sets of individuals [133], in which users contribute with their personal data and issue queries over the globally contributed data (e.g., computing recommendations, participative studies).

However, these exciting perspectives should not eclipse the security issues raised by the PDMS paradigm. Indeed, each PDMS can store potentially the entire digital life of its owner, thereby proportionally increasing the impact of a leakage. Hence, centralizing all users' data into powerful servers is risky since these data servers become highly desirable targets for attackers: huge amounts of personal data belonging to millions of individuals could be leaked or lost as illustrated by the recent massive attacks (e.g., Facebook, Yahoo or Equifax). Besides, such a centralized solution makes little sense in the PDMS context in which data is naturally distributed at the users' side [58].

Alternatively, recent works [9, 69, 42, 120] (see Chapter 2) propose to let the user data distributed on personal trustworthy platforms under users' control. Such platforms can be built thanks to the combination of (1) a Trusted Execution Environment (TEE) (i.e., secure hardware such as smart cards [4] or secure micro-controllers [9, 10, 69], ARM TrustZone [52], or Intel SGX [107]) and (2) specific software (e.g., minimal Trusted Computing Base and information flow control [71, 107]). In this chapter, we follow this approach and consider that a PDMS is a dedicated personal device that the user possesses and is secured thanks to TEE hardware.

In addition, as in many academic and commercial approaches [120], we assume that the PDMS personal device offers rather good connectivity and availability like, for instance, home-cloud solutions [120, 34, 93, 9] (e.g., a set-top box or a plug computer [9]). Thus, PDMSs can establish peer-to-peer (P2P) connections with other PDMSs, and can be used as data processor in order to provide part of the processing required in distributed applications. Hence, our objective is to study solutions based on a full distribution of PDMSs (called nodes interchangeably) which can act as data sources and data processors and communicate in a peer-to-peer fashion. *We discard solutions requiring recentralizing the distributed personal data during its processing, since this would dynamically create a personal data concentration leading to a similar risk as with centralized servers.*

Incorporating TEEs greatly increases the *data confidentiality protection* against malicious PDMS owners. However, since no security measure can be considered as unbreakable, we cannot exclude having some corrupted nodes in the system and, even worse, those corrupted nodes can collude and might very well be undistinguishable from honest nodes, acting as covert adversaries [19]. Also, since data processing relies exclusively on PDMS nodes, and given the very high scale of the distribution which disqualifies secure multi-party computation (MPC) protocols [111], sensitive data leaks are unavoidable in the presence of corrupted nodes, i.e.,

some data might be disclosed whenever a corrupted node is selected as a data processor.

In this chapter, we assess the feasibility of building a secure and efficient data processing system over a fully distributed network of PDMS housing covert adversaries. To this end, we provide mechanisms to reduce the data leakage to its minimum and make the following contributions (detailed in Section 4.3):

(1) We propose a P2P architecture of PDMSs, called SEP2P (for Secure and Efficient Peer-to-Peer), based on classical Distributed Hash Tables (DHT). We analyze the potential data leakages depending on the different levels of attacks possible on TEEs and formulate the necessary security requirements to circumvent them. We show that (i) communications between the system nodes should be *hidden*, i.e., in-transit sensitive data and metadata should be protected from an attacker; (ii) all the sensitive system data (i.e., data-at-rest and data-in-use required by the query processing) should be *dispersed* on *randomly* chosen nodes: and (iii) data tasks should be assigned to nodes in a *verifiable random* way, i.e., the assignment cannot be influenced by malicious colluding nodes.

(2) We focus on the verifiable random assignment problem and propose a generic solution (i.e., independent of the distributed computation tasks) which largely minimizes the verification cost (e.g., 8 asymmetric crypto-operations with a SEP2P network of 1M nodes of which 10K are colluding corrupted nodes).

4.2 Related Work

Efficient P2P Data Processing. Relying on a fully-distributed system induces several problems, e.g., integrating new nodes, maintaining a coherent global state, making nodes that do not know each other interact, handling churn, maintaining some metadata. It thus requires a communication overlay allowing for efficient node discovery, data indexing and search. Fortunately, these problems have already been extensively studied in the literature and the *Distributed Hash Tables* (DHTs) appear to be the solution reaching consensus.

Background 1. A **distributed hash table (DHT)** [108, 122, 83] in a P2P network offers an optimized solution to the problem of locating the node(s) storing a specific data item. The DHT offers a basic interface allowing any node of the network to store data, i.e., $store(key, value)$, or to search for certain data, i.e., $lookup(key) \rightarrow value$. DHTs proposals [108, 122, 83] share the concepts of keyspace or **DHT virtual space** (e.g., a 224 bits string obtained by hashing the key or the node ID), space partitioning (mapping space partitions to nodes, using generally a distance function), and overlay network (set of routing tables and strategies allowing reaching a node, given its node ID). For instance, the virtual space is represented as a multi-dimensional space in CAN [108], as a ring in Chord [122] or as a binary tree in Kademlia [83] and is uniformly divided among the nodes in the network. Thus, each node is responsible for the indexing of all the $(key, value)$ pairs where the key falls in the subspace it manages. Both the data storage and the lookup operations are thus fully distributed in a DHT. DHTs have interesting properties: uniform repartition of the data, scalability, fault tolerance and do not require any central coordination.

Hence, SEP2P leverages the classical DHT techniques as a basis for communication efficiency and scalability.

DHT security. Several works focus on DHT security [142] considering the following attacks: (i) Sybil attack: an attacker generates numerous false DHT nodes to outnumber the honest nodes. Introducing an (offline) certificate authority, is deemed to be among the most effective

defenses against the Sybil attack [32]. (ii) Routing table poisoning (eclipse attack): an attacker attempts to control most of the neighbors of honest nodes to isolate them. According to [142] the best strategy against such attacks is to constrain the DHT node identifiers. Again, using a central authority to provide verifiable identifiers is the simplest yet most effective way of achieving this goal [122]. (iii) Routing and storage attacks: Sybil and eclipse attacks do not directly impact the DHT, they are mainly necessary means for future attacks, like various denials of service (DoS). For instance, the objectives might be to prevent a lookup request from reaching its destination, denying the existence of a valid key, or impersonating another node to deliver false data. These DoS attacks are usually classified as routing and storage attacks and most of the mechanisms employed to negate them are based on redundancy at the storage and routing levels [142]. Thus, none of these works consider the secure and efficient actor selection for distributed processing as in SEP2P.

Secure Multi-party Computation and differential privacy. Cryptographic protocols have been proposed to protect the users' privacy in distributed computations with a focus on data confidentiality enforcement in personal data aggregation. Examples of computations related to this work are personal time-series clustering [5], kNN similarity queries [49], and location-based aggregate statistics [106]. However, MPC raises major scalability issues which in practice limit such protocols to specific types of computations [111].

Although it yields interesting results in privacy protection [45], differential privacy generally requires a central trusted aggregating node and ad-hoc adaptations depending on the targeted queries. As we search to provide a generic framework and exclude having a central actor to avoid a single point of failure, both requirements cannot be met by differential privacy. Even though local differential privacy [36] tries to address our first requirement, the solutions offered until now are still not generic, while the pertinence or the quality of the results may still be problematic with some applications [36]. Also, differential privacy exhibits intrinsic limitations with applications requiring continuous data flow aggregation (e.g., such as mobile participatory sensing) because of temporal correlation between consecutive data batches [30].

Distributed data aggregation using secure hardware. To overcome the limitations of MPC or differential privacy, several works propose using secure hardware at the user-side. Several secure protocols have been proposed for SQL aggregation [133], spatio-temporal aggregation [130], top- k full-text search [70], or privacy-preserving data publishing [6]. SEP2P also considers a secure PDMS at the user-side but our attack model considers having many colluding nodes. Moreover, the focus in SEP2P is on the secure and efficient random node selection. Differently, existing work focus on data aggregation or publishing and consider that all the nodes in the network participate in the protocol with their data being thus complementary to SEP2P.

Secure server-centric approaches. The above cited solutions are based on fully-distributed (P2P) or hybrid architectures. Alternatively, one could envision a solution based on a secured centralized server [17]. However, this raises important issues. First, users are exposed to sophisticated attacks, whose cost-benefit is high on a centralized database. Second, centralizing all users' data into one powerful server makes little sense in the PDMS context in which data is naturally distributed at the users' side. Hence, users might be reluctant to use such a massively centralized data service. Finally, new legislation such as the European GDPR [47] may hinder the development of such centralized solutions.

4.3 Approach and Scientific Results

4.3.1 SEP2P Architectural Design and Threat Models

Base System Architecture

SEP2P is a peer-to-peer system and only relies on the PDMS nodes to enable the aforementioned applications (see Section 4.1). Consequently, each node may play several roles for SEP2P applications:

Node role 1. Each node is a potential **data source**. For instance, producing sensed geo-localized data about the local traffic speed, or sharing grades used to compute recommendations.

Node role 2. Given the fully-decentralized nature of SEP2P, each node is a potential **data indexer**, also called **indexer**, providing part of the required distributed data indexing. For instance, the profile-based data diffusion apps require indexing the profile (i.e., the list of preferences) of each system node.

Node role 3. Given the fully-decentralized nature of SEP2P, each node is a potential **data processor**, also called **actor**, providing part of the required processing.

Node role 4. The initiator of a distributed processing is called the **triggering node** (T). T could be any node with participatory sensing applications, or the query issuer in distributed query or data diffusion applications.

Threat Model

Given the distributed nature of our system, the attacks can concern the communications between nodes or the nodes themselves.

At the communication level, we consider the following assumption:

Assumption 1. An attacker can observe the communications (i.e., content and metadata) between a *predefined* (i.e. that cannot change during a distributed computation) subset of nodes in the network.

This assumption indicates that the communications between nodes can be spied on by an attacker. A robust protocol should resist to a very large percentage of spied nodes, although we exclude from the scope of this threat model the “state-size attacks” in which an attacker could spy on the entire network.

At the node level, we consider the following two security assumptions as a security basis for the proposed protocols in our system:

Assumption 2. Each PDMS device is supplied with a *trustworthy certificate* attesting that it is a genuine PDMS.

Without this first assumption, an attacker can easily emulate nodes in the network and conduct a Sybil attack [46], mastering a large portion of nodes (and thus those being selected as actors), effectively defeating any countermeasure. Note that this does not require an *online* Public Key Infrastructure (PKI) since the certificate is attached to the hardware device, and not to the device owner, and hence can be pre-computed.

Assumption 3. Each PDMS is locally secured by a *Secure Hardware (SHW)* component.

This second assumption is reasonable considering that a PDMS is supposed to store the entire digital life of its owner and the large availability of this technology [9, 11], e.g., Intel SGX is present in most recent processors and ARM TrustZone in most mobile devices. It additionally provides a means to enforce the previous assumption.

Some of the most commonly available security features of SHW are: safe-keeping of secrets, tamper-resistance, and isolation and attestation of the executed code [110]. In particular, the isolation property means that the executed code and the data inside the SHW cannot be observed by an unauthorized process. This is particularly interesting in our context since it represents the building block to achieve data confidentiality.

However, no security feature can be considered as unbreakable and SHW is no exception: recent studies [139, 41, 63] showed how to execute side-channel attacks to leak part — if not all — of the computations performed within an SGX enclave. Resourceful attackers can also conduct highly advanced attacks, called *lab attacks* [126], to fully compromise a SHW with the help of laboratory equipment. In this chapter, we consider this latter, most powerful, type of attack.

Threat model (Active corruption): Malicious. Under this model, we assume that an attacker has complete control over her own PDMS, having bypassed the Secure Hardware protections.

Hence, we suppose attackers have access to laboratory equipment to perform lab attacks on their Secure Hardware and we thus impose no limit on their capabilities (except falsifying a certificate to contradict Assumption 2): observing the communications, uncovering data and even manipulating them are all actions feasible by an attacker.

Attacker Model

Having established the threat model, we need to clarify the notion of “attacker”. We consider that every owner of a PDMS is a potential attacker. This hypothesis especially means that we assume that the triggering node T is an attacker.

We also suppose that an attacker is not necessarily a single entity (e.g., an attacker can be one or several colluding malicious users) and can de facto possess more than one PDMS. For simplicity, we call *colluding nodes* the corrupted PDMSs controlled by the same attacker.

It is important to note that the worst-case attack is then represented by the *maximum number of colluding nodes in the system* (i.e., controlled by a single “attacker”). Corrupting few nodes can lead to some private data disclosure, but this disclosure is very limited in a well-designed system with a large number of nodes. Therefore, an attacker needs to increase the collusion range to fully benefit from the attack (i.e., access a significant amount of private data). The remaining question is thus: how many colluding nodes could an attacker control in the system?

Collusion Extent

There are two main difficulties to create a large group of colluding nodes: (i) the need to remain hidden and (ii) possessing the necessary equipment to perform advanced attacks on the Secure Hardware.

Indeed, colluding nodes must remain indistinguishable from honest nodes as detected malicious nodes can easily be excluded [19]. Since PDMS are associated to real individuals (e.g., by delivering the device only to real users proving their identity), collusion between individuals remains possible but can hardly scale without being minimally advertised, hence making them distinguishable and breaking their cover.

Regarding the second difficulty, this represents a heavily limiting attack factor since having access to such equipment may demand important resources. Besides, enough single attackers

must be willing to collaborate (despite mutual distrust) to create a large network of compromised nodes.

Thus, wide collusions are extremely difficult to build since it calls for significant organization between a large number of users, which, in practice, requires an extremely powerful attacker as well as extreme discretion. Nonetheless, we consider that a very powerful attacker could control a significant portion of the system nodes. Although worrisome, a situation with such a large proportion of colluding nodes does not prevent our system from functioning and from completing our objective: as we show in Section 4.3.3, our protocol for the verifiable random actor selection takes as input the maximum number of colluding nodes controlled by an attacker — *no matter that number*.

Problem Formulation

Our objective in this work is to first and foremost offer the maximum possible confidentiality protection of the users' private data under the above considered threat models and, in a second time, discuss possible optimizations so as to render the execution as efficient as possible.

Many other issues related to statistical databases (e.g., inferences from results, determining the authorized queries, query replay, fake data injection, etc.) or to network security (e.g., message drop/delay, routing table poisoning[138]) are complementary to this work and fall outside of its scope.

Issues related to integrity require a more in-depth look. Although TEEs have the means to attest that the intended computations were performed by the different nodes [66, 65], we cannot hope to maintain those capabilities when the TEE is fully compromised — specifically under the *malicious* threat model. Thus, in this conditions, we do not claim to offer integrity. Therefore, assuming that a proposed computational plan is guaranteed to offer confidentiality, it is debatable whether this confidentiality can still be achieved when the computation is altered. Assessing the loss of confidentiality is no easy task as there are numerous cases to consider: the type of data manipulated, the computations, the distribution of the data, the envisioned collusions. Note that in our context, not every and all aspects of the execution can be modified, the leeway an attacker has is limited to the addition of bogus results or the inclusion of more nodes in the query execution process. This restrained scope pleads in favor of a reduced impact but this study, mainly due to its sheer size, also falls outside of the scope of this work.

4.3.2 SEP2P Security Requirements

As indicated in Assumption 1, the communications between the nodes can be spied on (and this regardless of the considered threat model). By listening to the communications, an attacker can infer information based on the data itself (i.e., the content) or on the metadata (i.e., who is communicating with whom, when, after whom, etc.). We thus need to protect the sensitive communications which leads to the following requirement:

Requirement 1 (security). Hidden communications. Sensitive data and metadata should be protected such that an attacker cannot gain knowledge by listening to the communications of a predefined subset of nodes.

In the Malicious threat model we assume that, in addition to spying on the communications, an attacker could uncover the data normally protected by the Secure Hardware component. Hence, we consider that the set of nodes controlled by an attacker are leaking

data The objective is then to minimize this, unavoidable, data leakage. The leakage is indeed unavoidable in our particular context as (i) we rely on a fully-distributed system for both the storage of data and the query execution and (ii) corrupted nodes may be indistinguishable from other nodes. To address this, we propose the following requirement:

Requirement 2 (security). Random dispersion of data. Data-at-rest (e.g., the distributed node profile index) and data-in-use (i.e., the data exchanged during query execution) must be *dispersed* on *randomly* chosen nodes. Ideally, the overall leakage of all colluding nodes should not bring any valuable knowledge to the attacker that controls them.

Through dispersion we minimize the information each actor receives and, de facto, limit the impact of a leak. With randomization we ensure that an attacker cannot influence the selection of actors (to maximize data-in-use leakage) or of the indexers (to maximize data-at-rest leakage), thus limiting the number of corrupted nodes for a query. Note that the users' own data do not need dispersion since they are protected by the SHW and there is no gain in performing a self-attack.

Furthermore, in the Malicious threat model, besides observing the data manipulated by the SHW, the attacker could deviate from the general protocol, e.g., by executing a malicious code, in order to maximize the access to the sensitive data. Specifically, an attacker will search to influence the system to select as many corrupted nodes as possible as data processors (or actors). This leads to the following requirement:

Requirement 3 (security). Verifiable random actor selection. The selection of the data processor nodes for any system computation has to be *guaranteed to be random* (i.e., ensure that the colluding nodes cannot influence it) and *verifiable* by any node.

Obviously, these security requirements should not generate an overhead that renders the system unpractical especially for large systems and potentially a high number of colluding nodes. This leads to a final, transverse, requirement:

Requirement 4 (efficiency). Minimized security overhead. System protocols should minimize the number of costly operations, e.g., cryptographic signature verifications or communication overhead, and ensure system scalability with an increasing number of nodes or colluding nodes.

The random dispersion of data requirement is similar to the principle of compartmentalization in information security, which consists in limiting the information access to the minimum amount allowing an entity to execute a certain task. Typically, a node can execute a subtask without knowing the purpose or the scope of the global task. Dividing a given distributed computation in atomic tasks obviously depends on the precise definition of that computation. This aspect is shortly evoked in Section 4.4, while a journal article dealing with this problem was submitted and is currently under review.

Independently of the distributed protocol chosen to implement some given application, the system must delegate the data-oriented tasks to randomly selected nodes. Therefore, the random selection protocol is generic and constitutes the security basis of any distributed protocol in our system. However, under the Malicious threat model, it is challenging to design an actor selection protocol that is both secure and efficient. Section 4.3.3 addresses in detail this important problem.

4.3.3 Verifiable Random Actor Selection

Let us first detail some useful classical cryptographic tools focusing on the properties used in our protocol.

Background 2. A **cryptographic hash function** [89] is a one-way function that maps a data of arbitrary size to a fixed size bit string (e.g., 224 bits) and is resistant to collision. An interesting property of hash functions is that output distribution is uniform. In the following, $hash()$ refers to cryptographic hash.

Background 3. A **cryptographic signature** [89] can be used by a node n to prove that a data d was produced by n (authentication) and has not been altered (integrity). The signature is produced by encrypting $hash(d)$ using the private key of n . Any node can verify the signature by decrypting it using the public key of n and comparing the result with $hash(d)$. The signature includes the signer public key certificate, $cert_n$ (see Assumption 2).

We consider a system of N nodes, in which we want to randomly select A actors, despite wide collusion attacks from C colluding nodes. The main notations are summarized in Table 4.1.

N	Total number of nodes in the SEP2P system
A	Number of actor nodes (data processors)
C	Maximum number of colluding nodes ($C \geq 1$)
A_C	Average number of corrupted actors for a given protocol
A_{ideal_C}	Average number of corrupted actors for an ideal protocol
T	Triggering node (starting the execution)
k	Security degree
α	Security threshold
S	Execution Setter node, computing actor list
R_i, rs_i	DHT region R_i of size rs_i

Table 4.1: Main notations in SEP2P protocols

Effectiveness, Cost and Optimal Bounds

Ideally, we would want to ensure that all A actors are honest, but this is impossible, since colluding nodes are indistinguishable from honest nodes. Therefore, the best achievable protection is obtained when actors are randomly selected and the selection cannot be influenced by C colluding nodes, i.e., the average number of corrupted selected actors in the ideal case is $A_{ideal_C} = A \times C/N$ ($A_{ideal_C} > 0$). Thus, the impact of a collusion attack remains proportional with the number of colluding nodes, which is the best situation given our context. This guarantees that the attacker cannot obtain more private information than what she can passively get from observing the information randomly reaching its colluding nodes.

The following definitions quantify the security effectiveness and security cost of an actor selection protocol.

Definition 1. The **security effectiveness** of an actor selection protocol is defined as the ratio between A_{ideal_C} and the average number of corrupted selected actors for the measured protocol (A_C), i.e., security effectiveness = A_{ideal_C}/A_C . The security effectiveness has maximum value (i.e., 1) when $A_C = A_{ideal_C}$ and minimum value (i.e. C/N) when all the actors are corrupted.

Definition 2. A **verifier node** is a node who needs verifying the actor list before delivering sensitive data, e.g., a data source.

Definition 3. The **security cost** of an actor selection protocol is defined as the number of asymmetric cryptographic operations, e.g., signature verification, required by verifier nodes to check the selected actor list.

Note that the security cost considers only the verification of the actor list and not the cost of building the list. The rationale is that the verification cost has a larger impact on the overall performance since the number of verifier nodes can be high in a large distributed system: data sources need to verify the actor list before delivering their data.

Optimal bounds. The best possible case one could expect in terms of security effectiveness and cost in our context can be achieved using an idealized trusted server that knows all the nodes and provides a different random actor list for each system computation. This ideal solution reaches a maximal security effectiveness and a security cost of 1, since any verifier node must only check the signature of the trusted entity.

Evidently, this solution is not acceptable since it represents a highly desirable target for attackers, i.e., a central point of attack and contradicts the fully distributed nature of SEP2P. Therefore, we need distributed solutions relying only on the nodes. To underline the existing tension between security effectiveness and cost, we discuss two basic distributed protocols for the actor selection, focusing either on the security cost or on the security effectiveness. To simplify the protocols description, we initially assume a full mesh network overlay, i.e., each node knows the complete list of nodes in the system and its evolution over time.

Baseline cost-optimal protocol. The triggering node (T) selects randomly the actors. The security effectiveness is minimal: $A_C = \min(A, C)$ since T may be corrupted (which is the case when any node can trigger a computation). There is thus no necessity to provide any signature: the security cost is 0.

Baseline security-optimal protocol. Proposing an optimal protocol in terms of security is challenging in a decentralized architecture (without any supporting trusted party) and considering covert adversaries. This conjunction leads to a situation where no single node in the system can claim to securely provide a list of actors (the provider itself can be corrupted). The work in [20] proposes the CSAR protocol which provides a secure way to generate a verifiable random value under the condition that there is at least one honest node participating in the distributed protocol. Considered in our context, we can ensure generating a real random value only if there are at least $C + 1$ participating nodes. Also, once we obtain a verifiable random value, we can derive up to A random values by repeatedly hashing the initial value $A - 1$ times. The final step is to map the set of A random values to the nodes. This can be easily done, e.g., by sorting the nodes on their public key and associating the random value to a rank in the sorted list. This protocol has an optimal security effectiveness, i.e., 1, since the actors are guaranteed to be selected randomly. On the other hand, checking the CSAR results requires one signature verification per participant. Thus, the security cost is $C + 1$ asymmetric cryptographic operations per verifier node. Since C can be large, such a solution cannot scale with large systems and wide collusion attackers as it would lead to an extreme verification cost.

Moreover, to achieve these security bounds, both protocols require a full mesh network overlay which is also extremely costly to maintain in practice, especially for large networks. This contradicts the efficiency and scalability requirement formulated in Section 4.3.2. Using a DHT overlay instead of a full mesh solves the problem of communication efficiency/scalability.

However, this will impact the optimal bounds of both protocols. For the first protocol, the security cost increases from 0 to up to A since a verifier node which does not “know” any of the actors has to verify their certificates to be sure that the actors are genuine PDMSs (to avoid Sybil attacks). Similarly, for the second protocol, the security cost increases to $2(C + 1) + A$ for the same reason, i.e., checking that participant and selected actors are genuine PDMSs. Even worse, the optimal security effectiveness can no longer be guaranteed since with a DHT, there is no secure way of associating the random values to the nodes unless using secure DHT techniques [138] with a large impact on performance.

Overview of the Proposed Solution

To address all these problems, we propose a protocol that reaches maximal security effectiveness at a verification cost of $2k$. k is called the *security degree* and is very small. Also, our protocol builds directly on a classical, efficient DHT overlay without requiring any modifications. We describe some important features in SEP2P which make this possible and then sketch the protocol.

Imposed and uniform distribution of node location: the node ID, used when inserting a node in the DHT, is imposed in SEP2P, in a way that leads to a uniformly distributed node location in the DHT virtual space. Consequently, colluding nodes are also evenly distributed in the DHT, thus avoiding spatial clusters. We use extensively this property to drastically reduce the cost of security by taking localized decisions (see below), i.e., limited to the nodes situated in “small” regions in the virtual space. Achieving imposed node location is easy, based on the public key of the certificate of each node. We compute a cryptographic hash of this key, which is, by construction, uniformly distributed, and use this hash for insertion in the DHT virtual space. The advantages of using the public key are (i) its uniqueness; and (ii) the node location can be checked with a single signature verification.

Probabilistic guarantees: Given the imposed, uniform node location which applies indistinctly to honest and colluding nodes, we can have probabilistic guarantees on the maximum number of colluding nodes in a DHT subspace of a given size, called *DHT region* hereafter. We can compute the probability of having **at least k colluding nodes** (see the following Section “Providing Probabilistic Guarantees”) and choose the DHT region size such that the probability is very close to 0. In our context, we want to have a probability smaller than α , the security threshold. The main idea is to set α so that the probability of having k colluding nodes in the same region becomes so low that we can consider that it “never happens”, e.g., $\alpha = 10^{-6}$. Such a guarantee is used in the protocol sketched below and then detailed in the following subsection.

Sketch of verifiable selection protocol of A actors (see Figure 4.1)

1. Run a distributed protocol inspired from CSAR [20] to generate a **verifiable random value**, i.e., proven to have been truly randomly generated by k nodes if at least one is honest. The k nodes are selected in a DHT region R_1 , centered on the triggering node (T), whose region size rs_1 is set such that we have probabilistic guarantees to “never” (probability $< \alpha$) have k or more colluding nodes, i.e., at least one of the k nodes is honest.
2. Map the hash of that random value into coordinates to define a location p in the DHT virtual space and contact through the DHT the node, called **execution Setter** (S), managing this location.

3. S then selects k nodes (the actor list builders) in a region R_2 , centered on p , using probabilistic guarantees, such that we “never” have k or more colluding nodes. Given the uniform distribution of the node on the virtual space, we have $rs_2 = rs_1$.
4. Each actor list builder then selects A nodes in a region R_3 , centered on p , whose size rs_3 is such that R_3 includes at least A nodes with high probability.
5. Run a **distributed verifiable selection protocol** in the spirit of [20] such that the k nodes selected in (3) can: (i) check the validity of the random value generated in (1); (ii) build the actor list securely; (iii) sign both the random value and the list of A actors. This step is detailed below (see Section “Distributed Secure Selection Protocol”).

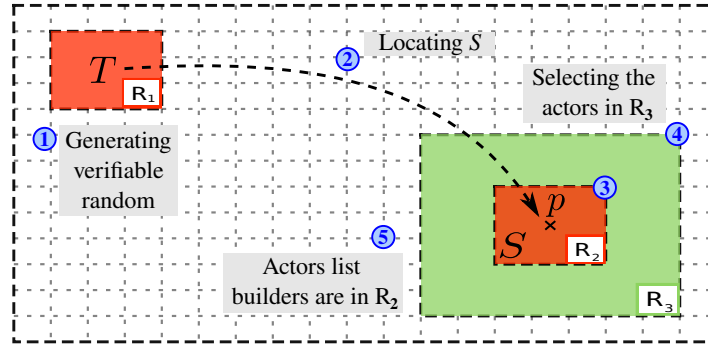


Figure 4.1: Sketch of verifiable selection

The result is a list of A actors that is signed by k nodes, among which at least one is honest. Doing so reduces the verification cost to $2k$ asymmetric cryptographic operations: k to check the certificate of the k list builders, verifying that they belong to region R_2 , centered on p ; and k to check each builder signature.

Providing Probabilistic Guarantees

To generate verifiable random values or validate the query actor selection, SEP2P employs distributed computations between a small subset of the nodes thanks to the notion of node legitimacy and probabilistic guarantees defined below using the notations in Table 4.2.

$kpub_n$	Public key of node n
$cert_n$	Trustworthy certificate of node n
$sign_n$	Signature by node n (includes $cert_n$)
TL_i	execution Trigger Legitimate node i
RND_i	Random number generated by TL_i
$(V)RND_T$	(Verifiable) random generated by T
SL_j	execution Setter Legitimate node j
RND_S	Random generated by S
CL_j	Partial candidate list of legitimate nodes w.r.t. R_3
CL	Candidate List of legitimate nodes
$(V)AL$	(Verifiable) Actor List

Table 4.2: Complementary notations in SEP2P protocols

Definition 4. Legitimate nodes. Given a region R in the virtual space of a DHT, for any node i we say that node i is legitimate w.r.t. R iff $hash(kpub_i) \in R$.

To be able to provide probabilistic guarantees as explained above, we need to estimate the number of nodes in a region:

Lemma. Let R be a DHT region of size rs in a virtual space of a DHT of total size 1 (i.e., normalized) and let N be the total number of network nodes with a uniform distribution of the node location in the virtual space. The probability, PL , of having at least m legitimate nodes in R is:

$$PL(\geq m, N, rs) = \sum_{i=m}^N \binom{N}{i} \cdot rs^i \cdot (1 - rs)^{N-i} \quad (4.1)$$

Proof (sketch): Let us consider a partition of the N nodes into two subsets containing i and $N - i$ nodes. Since the distribution of nodes is uniform in space, the probability of having the i nodes inside R and the $N - i$ nodes outside R is $rs^i \cdot (1 - rs)^{N-i}$ and there are $\binom{N}{i}$ possible combinations of generating this node partitioning. The probability of having at least m nodes in R is equal to the probability of having exactly m nodes plus the probability of having exactly $m + 1$ plus . . . the probability of having N , which leads to the equation in (4.1).

Application to colluding nodes: Let $C < N$ be the maximum number of colluding nodes. We can apply formula 4.1 to compute the probability, PC of having at least k colluding nodes in R :

$$PC(\geq k, C, rs) = \sum_{i=k}^C \binom{C}{i} \cdot rs^i \cdot (1 - rs)^{C-i} \quad (4.2)$$

We can notice that this probability only depends on C . It does not depend on the region center since we have a uniform distribution of the nodes on the virtual space.

Verifiable Random Generation

Our goal is to generate a random value, using k nodes and to guarantee that none of the k nodes can choose the final computed random value (or any of its bits). Any node in the system should be able to check the validity of this random value (i.e., to have proofs that it has been correctly generated). This is possible as soon as at least one of the k nodes is honest, this guarantee being obtained thanks to equation (4.2) by choosing the adequate size for the DHT region R and by using k legitimate nodes w.r.t. R .

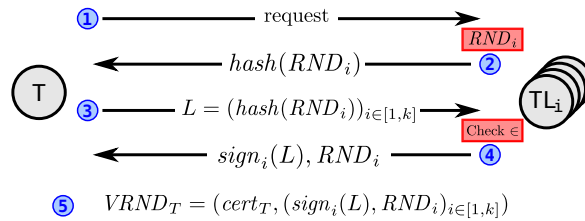


Figure 4.2: Verifiable random

A node T wanting to generate a verifiable random, selecting a region of size rs_1 with $PC(rs_1) < \alpha$ centered on itself, executes:

Verifiable random number generation protocol

1. T contacts any k legitimates nodes TL_i ($i \in [1, k]$) w.r.t. R_1 .
 2. Each TL_i sends $hash(RND_i)$ to T , where RND_i is a random number (on the same domain as the hash function, e.g., 224 bits) TL_i generates.
 3. Once T has received the k hashes, it sends back the list L of hashes to the TL_i s; $L = (hash(RND_i))_{i \in [1, k]}$.
 4. Each TL_i checks that $hash(RND_i) \in L$, and, in the positive case, returns $sign_i(L)$ and RND_i .
 5. T gathers the k messages and builds the verifiable random: $VRND_T = (cert_T, (sign_i(L), RND_i)_{i \in [1, k]})$.
-

The above random generation protocol is adapted from [20] which includes a formal proof. Note that the protocol in [20] does not include the notion of node legitimacy and thus needs $C + 1$ participating nodes instead of k . Intuitively, the nodes commit on their selected random value by sending its hash (Step 2), and all the hash values are known by each of the k nodes before providing the final signature (Step 4). Therefore, an attacker controlling $k - 1$ TL_i nodes cannot influence the final random value since these nodes cannot change their random values (committed at Step 2). Thus, the correct random value of a single honest node is enough to obtain a truly random final value RND_T .

To obtain and check the verifiable random value, any node must: (i) check $cert_T$ and compute L by hashing all RND_i ; (ii) for $i \in [1, k]$, check $cert_i$, check the legitimacy of TL_i using $cert_T$ and validate $sign_i(L)$. The final random value is $RND_T = RND_1 \oplus RND_2 \oplus \dots \oplus RND_k$.

In (i), we verify that T is a genuine PDMS, retrieve the center of the region R_1 and compute L , both being necessary for the next verification; (ii) starts by confirming that each TL_i is genuine, then it ensures that they are legitimate w.r.t the location of T and R_1 , after which it confirms the hash list by checking the signatures, and finally, it computes RND_T .

Distributed Secure Selection Protocol

The main goal of the proposed protocol is to select the A actors such that this selection cannot be influenced by colluding nodes.

Definition 5. The **execution Setter** (S) is chosen randomly based on a verifiable random generated by T . Its role is to coordinate the selection of the computation actors and to setup the execution by sending the appropriate information to each actor.

In the following, we assume that each node n in SEP2P keeps a node cache, called $cache_n$, of the IP address and certificate of legitimate nodes w.r.t. a region of size rs_3 centered on node n location.

SEP2P distributed secure actor selection protocol

1. Generates the verifiable random $VRND_T$.
2. Maps $hash(RND_T)$ into coordinates and contact S through the DHT.
3. S contacts any k legitimates nodes w.r.t. R_2 , SL_j ($j \in [1, k]$) and sends to each $VRND_T$.
4. Each SL_j sends $hash(RND_j \parallel CL_j)$ to S , where RND_j is a random number SL_j generates, and CL_j is the set of nodes from $cache_j$ which are legitimate w.r.t. R_3 .
5. Once S has received the k hashes, it sends back the list L_1 of hashes to all SL_j ; $L_1 = (hash(RND_j \parallel CL_j))_{j \in [1, k]}$.

6. Each SL_j checks that its own $hash(RND_j \parallel CL_j) \in L_1$ and, in the positive case, returns RND_j and CL_j .
7. S gathers the k messages and sends to all SL_j the list $L_2 = ((RND_j, CL_j)_{j \in [1, k]})$.
8. Each SL_j does the following:
 - (a) Checks $VRND_T$ and computes RND_T .
 - (b) Checks that each (RND_j, CL_j) from L_2 is consistent with the corresponding $hash(RND_j \parallel CL_j)$ from L_1 .
 - (c) Computes the union, after removing possible duplicates, of all CL_j to obtain a candidate list of legitimate nodes CL .
 - (d) Computes the $RND_S = RND_1 \oplus RND_2 \oplus \dots \oplus RND_k$.
 - (e) Sorts CL on $kpub_n \oplus RND_S$ (where $kpub_n$ is the public key of a node $n \in CL$) and selects the A first candidates to build the actor list AL .
 - (f) Checks the legitimacy of AL nodes w.r.t. R_3 .
 - (g) Signs (RND_T, AL) and sends it to S .
9. S gathers k results and builds the verifiable actor lists: $VAL = (RND_T, AL, (sign_j(RND_T, AL)))_{j \in [1, k]}$.

The goal of **steps 1 and 2** is to displace the DHT region, where actors will be selected, from T to S with three benefits: (1) T is likely to be corrupted (as any node is allowed to trigger a computation) while S is chosen randomly using the verifiable random protocol; (2) it distributes the potential leaks in a different region for each computation; (3) it balances the load on the whole SEP2P network thus improving the overall performance.

Steps 3 to 6 are similar to steps 1 to 4 of the verifiable random protocol, except that the signature by SL_j is delayed to Step 8.g. Delaying the signature allows SL_j s to check and attest the validity of $VRND_T$ (step 8.a). The protocol cost is increased (since k nodes verify $VRND_T$) but the verifying cost is reduced accordingly since having k SL_j s signing RND_T (step 8.g) means that it is correct (remind that at least one of the k SL_j s is honest).

Steps (8.b) to (8.e) are dedicated to the actor list building (AL) based on the candidate list (CL) and deserve a more detailed explanation: in our context, in order to securely build the actor list, the k participants first have to agree on a common basis and then execute, in parallel, a procedure that is unpredictable and gives identical results to all participants. Since it is unpredictable we are certain that the inputs cannot be manipulated beforehand so as to influence the rest of the procedure. Since it gives identical results for all actor list builders, and since at least one node is honest, we are sure that no colluding node can alter the results. By sorting the nodes in CL using a verifiable random number and the public keys of the nodes fulfills both requirements: the random number takes care of the unpredictability, while the commitment of each SL_j on their intermediary lists in step 4, coupled with the XOR operation on the public keys of CL nodes, is a simple yet effective way of producing identical results.

In **steps 8.f and 8.g**, k SL_j s check the validity of the result, i.e., that any actor of AL belongs to R_3 and attest it by signing the results. Note that this check is not necessary for any actor n in AL that was found in k CL_j since this fact attests that at least one honest node possesses n in its $cache_j$. Assuming $cache_j$ contains only genuine nodes (we say that $cache_j$ is valid) and since $rs_3 > rs_2$, most of the actors in AL will be found in k CL_j , thus diminishing drastically the actor list building cost. Actually the validity of $cache_j$ is necessary to ensure that a colluding node selected as SL cannot hide honest nodes with the hope of having a larger

proportion of colluding nodes in AL . Indeed, at least one of the SL is honest and will provide its full $cache_j$ that will be thus included in CL . We can observe that $cache_j$ can be actually seen as the relevant part (for node j) of a full mesh network, which offers its benefits without paying the whole maintenance cost.

To check the verifiable actor list (VAL), any verifier node must do: for $j \in [1, k]$, check $cert_j$, check the legitimacy of SL_j using RND_T and validate $sign_j(AL)$. Thus, the verifying cost is limited to k certificate verifications and k signature verifications, i.e., $2k$ asymmetric crypto-operations. our experimental evaluation shows that k is generally lower than 6.

4.3.4 Experimental Validation

We identified all the parameters that may impact the security and efficiency of the proposed strategies and considered all the metrics that are worth evaluating to analyze the strengths and weaknesses of the proposed strategies, i.e., security effectiveness and cost, setup cost, scalability, robustness w.r.t. failure or disconnections. Let us note that a real implementation of the SEP2P distributed system is not very useful if we consider the above listed objectives of the evaluation. Also, measuring the scalability for very large systems (e.g., 10M nodes) with many parameters is practically impossible. Therefore, as in most of the works on distributed systems [108, 122], we based our evaluation on a simulator and objective metrics. That is, the latency is measured as the number of asymmetric crypto-operations and exchanged messages between peers instead of absolute time values. This allows for a more precise assessment of the system performance than time latency, which can greatly vary in our context because of the node heterogeneity (e.g., TEE resources or network performance). Our simulator is built on top of a DHT network. Currently, we implemented Chord and CAN as DHT overlays and use Chord for the results discussed. The reader interested can find in [79] the complete set of experimental results. Nonetheless, we can list the following highlights from the implementation and its performance analysis.

- **Security effectiveness:** SEP2P achieves an ideal security effectiveness, i.e. as good as a trusted server, independently of the number of colluding nodes. Indeed, the selection of actors is truly random, thus providing the same results as the ideal case. In addition, the verification cost ($2k$) is also very low (4 to 8 asymmetric crypto-operations for $C\% \leq 1\%$, where $C\%$ is the % of colluding nodes).
- **Setup cost:** The actor list setup cost is determined by two design choices: (1) to increase the security effectiveness, we run our protocol on k SL nodes thus increasing the total setup cost; and (2) we voluntarily make most of the checks during the setup (e.g., checking the actor certificates or verifying their availability) in order to reduce, as much as possible, the subsequent verification cost. Since this verification process will potentially be performed by a (very) large set of nodes (e.g., data sources), it is in our best interest to reduce it to avoid overloading the entire system.
- **Scalability:** To study the scalability, we computed the averaged k value varying C and N . Indeed, k is the main factor in the verification cost, setup latency and total work (since everything is done k times). Our results show that: (1) **SEP2P is highly scalable w.r.t. N :** Indeed, k values are identical for small and large networks independently of α if we consider the percentage of colluding nodes and not the absolute value (e.g., 1% colluding nodes is equivalent to absolute values of $C = 100$ and $C = 100K$ for the small and large networks). (2) **k increases slowly when $C\% < 1\%$:** k remains smaller than 6 even with $\alpha = 10^{-10}$. For $N = 10M$ and $C\% = 1\%$, the k -optimization

reduces the number of participants in the verifiable random generation from $100K$ to 6.

(3) α has a small influences on k : increasing α by four orders of magnitude increases k from 1 unit (e.g., $1K$ colluding nodes for $N = 10M$) to 5 units (e.g., $1K$ colluding nodes for $N = 10K$ or $1M$ colluding nodes for $N = 10M$).

4.4 Related Contributions and Future Work

In this section, we overview our contributions related to confidentiality-preserving distributed data management in the PDMS context and discuss some future work stemming from the proposed secure and efficient P2P protocols.

4.4.1 Scientific Contributions

Contributions related to preserving data confidentiality in P2P PDMS systems. As indicated at the beginning of this chapter, its content is based on [79] (see Appendix F). A prototype implementation showing how to secure highly distributed queries based on SEP2P was presented in [78].

- In [79] we motivate the study of fully-distributed queries in the PDMS context and formulate the security and efficiency requirements in the presence of corrupted colluding nodes. We show that, under the considered threat model, there are two leverages to increase the system security, i.e., minimize both the risk and the impact of data leakages. We focus on the risk minimization problem which translates in guaranteeing a truly random actor selection for any system data processing, i.e., the colluding nodes cannot influence the selection of the data processor nodes. We show that data processing tasks can be assigned to nodes in a verifiable random way, which cannot be influenced by malicious colluding nodes. Then, we propose a generic solution which largely minimizes the verification cost. Our experimental evaluation shows that the proposed protocols lead to minimal private information leakage, while the cost of the security mechanisms remains very low even with a large number of colluding corrupted nodes.
- In [76] we study the second important issue in a fully-distributed PDMS architecture, i.e., minimizing the impact of a data leakage, in the context of aggregate queries targeting a subset of the system nodes which have a certain user profile. As in [79], we consider the presence of corrupted colluding nodes. We then gradually construct a protocol, called DISPERS, which satisfies all the formulated requirements to minimize the impact of a data leakage, i.e., hidden communications and random dispersion of data, while maintaining a low security overhead.
- A demonstration platform of DISPERS, built on top of SEP2P2, has been demonstrated in [78]. The main objective of the demonstration platform is to graphically illustrate the query execution in details to show that DISPERS leads to maximal system security with low and scalable overhead.

4.4.2 Future Work

This chapter deals with the important issue of preserving data confidentiality for computations in fully-distributed PDMS systems. While the formulated threat models and associated

security requirements are generic, the focus of this chapter contribution is on the verifiable random actor selection in the presence of corrupted colluding nodes. This problem is fundamental since generating a truly random actor list is the basis for minimizing the risk of leakage, and it is also generic since it applies to any type of computation in a distributed system.

However, to further increase the confidentiality guarantees, we dispose of a second leverage, which is to minimize the impact of a data leakage whenever it may (unavoidably) happen through random data dispersion and hidden communications. Unfortunately, these two security requirements are dependent on the computation the system has to perform, as well as the data needed to be exchanged during the process. Moreover, the considered threat model drastically impacts the way these requirements could be implemented in a secure and efficient way.

As indicated in the previous section, a first protocol, called DISPERS, has been presented in [76]. DISPERS targets applications that need to compute aggregates over the data supplied by the subset of system nodes that exhibit a certain user profile. Through random data dispersion, DISPERS manages to protect both the sensitive data-at-rest (i.e., the distributed index of user profiles) as well as the data-in-use (i.e., the user profiles and node local data exchanged during query processing).

However, DISPERS only considers the Malicious threat model and as such, provides protocols that are optimized for this context. We conducted an in-depth study to generalize the implementation of the random data dispersion to the other possible threat models in the PDMS context (see Section 4.3.1). Similarly, the hidden communication requirement also depended on the considered threat model and thus leads to different implementations. This analysis allows to adapt the distributed protocols to specific attacks and thus provide optimized versions for specific threats. Thus, we submitted recently, to a major databases journal, an article dealing with the above enumerated issues.

Finally, considering the defined security requirements in the specific context of other important applications (e.g., mobile participatory sensing apps) is another direction for short-term future work.

Chapter 5

Conclusion and Future Research Perspectives

Summary. *This chapter concludes this document by presenting a few directions of future work for my research.*

Conclusion

Worldwide smart disclosure initiatives and new regulations have opened the way to user empowerment in the personal data ecosystem. Riding this wave, many Personal Data Management System solutions are currently flourishing with the objective to offer individuals appropriate tools to collect and manage their personal data. However, there is still a long way to go before arriving at a trustful ecosystem for both the individuals and the economic, social and administrative actors. In pursuing this goal, we have presented in this document an overview of our approach to achieve a secure and extensive PDMS. At the heart of this approach there are two key elements. This first one is the massive (but natural) decentralization of personal data at the individual level, i.e., each individual holds her personal data within a PDMS instance whatever the format the PDMS platform may take. The second one is to employ a Trusted Execution Environment as the cornerstone of the PDMS security, laying out the premises of a trustworthy data platform. This is obviously necessary to protect the entire digital life of each individual, but also to provide mutual trust whenever individuals may exchange data within (large) communities of PDMS users or with third parties. Walking on these two feet, we have addressed different important problems in the PDMS context ranging from architectural issues, to secure personal computations or collective computations.

Specifically, in Chapter 2, we have discussed the set of functionalities and related security requirements that any PDMS solution should consider and have proposed a preliminary design for an extensive and secure PDMS reference architecture satisfying the considered requirements. We have also presented some concrete ES-PDMS instances based on existing TEEs. In Chapter 3, we have considered a PDMS instance based on a specific TEE, i.e., a tamper-resistant hardware device (also called secure token). In this context, we have tackled the problem of designing a scalable embedded search engine for secure tokens, which allows PDMS users to securely manage and share their documents. Finally, in Chapter 4, we have studied the feasibility of a pure peer-to-peer personal data management system for secure execution of queries and computations. In particular, we have focused on the important security requirement related to the verifiable random actor selection and have provided a generic solution which largely minimizes the verification cost even with a large number of colluding corrupted nodes.

In the remainder of this chapter, I will briefly introduce some important challenges for my future research. For the sake of conciseness, I will only discuss the future works in relationship with the scientific aspects developed in the three main chapters of this document (i.e., Chapters 2 to 4): (i) issues related to the physical instances and implementation of an ES-PDMS; (ii) issues related to securing the personal computation in the ES-PDMS architecture; and (iii) issues related to securing the collective computations in a fully-decentralized PDMS system.

ES-PDMS Physical Instances

The first class of challenges directly stems from the reference architecture presented in Chapter 2 and concerns its modular design and its robust concrete implementation notably in Trusted Execution Environments. As sketched in Section 2.3.3, the Core engine is expected at least to provide authentication and secure personal data storage and access, to enforce security and privacy policies, and to sequence the execution of Data Tasks installed under the user's control. While existing work (e.g., relational algebra, iterator model and classical query

rewriting to access data) can be reused to this end, a particular focus must be put on *minimizing the code size and complexity* such that the Core engine can be proven through formal methods. This leads to delegate non-critical operations to Data Tasks potentially based on existing (untrusted) code modules or libraries, interacting with the Core without endangering the global security of the computations.

Embedding Core primitives or Data Task code in SGX enclaves also raises interesting research problems. The SGX enclave technology holds great promise for secure and powerful data processing. Typical limitations are however the cryptographic overhead of accessing persistent data outside the enclave [28], the limited RAM amount of each enclave [40] and the cost of external function calls [107] and memory access overhead which both may be orders of magnitude higher than in a regular environment. A precise analysis of SGX enclaves w.r.t. data oriented operation remains to be done, but will undoubtedly raise many interesting data management challenges. Typically, core DB mechanisms linked to transaction atomicity, buffer management, fine grain in-place updates, and memory intensive operations like joins must be revisited. EnclaveDB [107] makes a first step into this direction and proposes a smaller footprint SQL database engine where all database structures are managed in an Intel SGX enclave, except the database log which is managed inside the enclave by a secure logging and check pointing protocol and stored in a protected manner outside the enclave to achieve durability. Distributed data processing may also take advantage of Intel SGX enclaves [118, 40].

We have started to implement the proposed PDMS architecture using Intel SGX enclaves [38] as a basis for security (see Section 2.3.4) and recently submitted, to a major databases conference, a demonstration of a prototype platform presenting important PDMS functionalities (i.e., data collection and processing) and the related security properties. A first objective of our current prototype is to leverage SGX enclaves to deal with the mutual trust required for both the data collection and the local data processing functionalities. Indeed, given the peculiar ecosystem that the PDMS paradigm creates by concentrating user data at the user-side, security is of paramount importance with a mutual security goal to be attained. On the one hand, the individuals need a trustworthy platform capable of protecting their data against all typical privacy threats (e.g., data snooping and secondary data uses). On the other hand, an important specificity in the user-centric PDMS context is that apps “move” towards the data as opposed to personal data migrating towards remote services as it happens with most existing cloud services. Third party services to which the owner may want to subscribe (e.g., pay-as-you-drive car insurance premium computation or smart meter electricity billing) require guarantees regarding the results produced by a PDMS, i.e., the assurance that the result was indeed produced by a certain computation code using the required input data. In particular, the mutually trusted personal computation property: (i) guarantees to the PDMS owner that the minimal collection principle enacted in laws protecting personal data (e.g., GDPR) is fulfilled and that the computation cannot leak unexpected data nor be influenced by a potentially corrupted user environment; (ii) conversely, it guarantees to a third party (e.g., an energy provider willing to compute the owner’s bill) that the result has been faithfully computed by the expected extension code. Later on, we will focus on integrating in the platform the collective computation and administration functionalities (see Section 2.3).

I am involved together with the permanent members of Petrus in the design of the ES-PDMS platform on SGX. Floris Thiant, one of the Petrus engineers, is particularly committed to the development of this platform with the help of Robin Carpentier, one of the PhD students in Petrus. On the longer term, our objective is to link most of the research actions in Petrus

to the ES-PDMS platform and also provide implementations for other TEEs such as ARM TrustZone.

ES-PDMS Bilaterally Trusted Personal Computations

To achieve the expected set of functionalities covering the complete data life-cycle with a limited Trusted Computing Base, most of the advanced computations have to be implemented as Data Task in our architecture. This way, a PDMS can execute any arbitrarily complex but untrusted computation code with access to some (large amounts of) PDMS raw data. The Core ensures that the Data Task computation code only accesses the required raw data, and that only the result is shared and attested with a third-party app. A first important observation in this context is that TEEs offer code and data confidentiality inside the enclave but assume code is trusted. Since the code of Data tasks is generally untrusted, Data Tasks require another level of isolation, i.e., enforcing reverse isolation to provide in-enclave sandboxing. This will prevent a malicious Data Task to voluntarily leak the raw data to a third-party using a side-channel (e.g., writing data to a specific area of the RAM). Recent works such as [54, 143] provide this kind of reverse isolation inside SGX enclaves. For instance, the authors of Ryoan [54] propose mechanisms to secure the processing of user data on third-party platforms. In particular, by exploiting Intel SGX, they allow these third-party platforms (such as Amazon Cloud) to perform processing while preserving the confidentiality of user data and the secrecy of the proprietary algorithms that process it. The characteristics of Intel SGX mean that a program, while inaccessible from outside the enclave, has capabilities that may be undesirable in terms of isolation: i) it can communicate with programs outside the enclaves; ii) it has access to many of the features offered by the CPU (random number, syscalls, access to secondary storage, etc.). Thus, the authors complete the SGX native isolation by sandboxing: a Ryoan module is first instantiated in an SGX enclave which then loads the program into a sandbox. This more complete isolation is essential for the execution of our Data Tasks.

An ES-PDMS architecture based on a trusted and isolated (i.e., running in an SGX enclave) Core, and isolated and sandboxed Data Tasks, drastically limits the possibility for an attacker to access private data even if the attacker controls the operating system or the apps, or has direct access to the machine, or even programs the code of Data Tasks, unless she breaks the hardware security (e.g., through side channel attacks [139, 41, 63]). However, even if a Data Task is sandboxed, it can still leak private information through the computation result. We consider an active attacker who wants to obtain the maximum amount of data stored in the enclave via the results declassified by the Core and received by the App. This leakage channel cannot be completely closed without preventing any processing on the data set. In a first step, we will focus on an attacker who is not interested in some specific data in the user private data set but who searches to obtain the maximum possible amount of private data. We study more particularly the cases where the attacker can request the execution of a large number of queries in which even small leaks will result in the disclosure of a large part of the data set. We want to prevent these attacks automatically without using semantic analysis of data or results, or code auditing. These solutions are complementary to ours. The difficulty is increased by the Core minimality which is required to maintain a high level of trust (as discussed in Section 2.3.3). Hence, we assume that the Core cannot include modules for semantic interpretation of the data set or the results produced, nor modules for analyzing the Data Tasks code. These impose costly treatments and increase the Core complexity, which

extends its attack surface.

In this context, we will focus on techniques to limit the amount of data that an attacker could retrieve via the result and find answers to the following questions. How can an attacker exploit repeated execution of queries to extract a large amount of data from the user under this attack model? Without using semantic analysis (over the data or the computations result) or code auditing, how can the data leakage be limited? Are there cases where no stronger guarantees can be provided? Assuming some constraints are imposed to limit the leakages, can we maintain sufficient (programming) extensibility for the Data Tasks and have reasonable security overhead?

Furthermore, these security constraints depict an architecture employing a joint and asymmetrical data management between the secure but limited Core and rich but untrusted and isolated Data Tasks. The Core can only support basic, generic data management operations (e.g., in the spirit of a KVS) and has access to storage, while Data Tasks implement advanced data processing techniques but require to map them to the Core API for persistence. This PDMS architecture requires redesigning traditional data access methods to achieve security and efficiency. I have started addressing this type of issues in the context of the PhD thesis of Robin Carpentier which I co-advise with Nicolas Anceaix and Guillaume Scerri (see Appendix A).

ES-PDMS Mutually Trusted Collective Computations

The PDMS paradigm promises to pave the way for new innovative uses developed around personal data, including distributed computations on a large number of PDMS (e.g., automatic classification, recommendations, participatory studies). Such examples often require the training of an artificial intelligence (AI) model based on a large volume of user data, also raising important challenges. Thus, organizing secure and efficient distributed computing across a large number of PDMSs can be complex, especially in the presence of a potentially large number of corrupted nodes. Hence, our objective is to provide appropriate solutions to efficiently train an AI model (e.g., a deep neural network) in a fully distributed system while providing strong security guarantees to the participating nodes.

In the current model, it is necessary to collect the personal data on which the AI model is built. However, the data necessary for AI learning requires, as for Big Data, three "Vs": Volume, Veracity and Variety. The constitution of such a database for a learning calculation faces four families of problems to consider:

- *Data confidentiality protection.* Learning an AI model requires a large volume of good quality data. However, it is risky to centralize huge amounts of personal data belonging to millions of people on powerful servers, as they become highly desirable targets for attackers and are regularly subject to attacks as illustrated by the recent massive data leaks (Yahoo, Equifax, ...).
- *Ease of contribution.* An AI model requires personal and potentially sensitive data: health, travel, consumption, behaviors, purchases, social interactions, etc. Also, under the current European legislation that requires *a priori* user consent to collect and exploit these data, it becomes increasingly difficult to create a learning database with reasonable resources and time, especially when these data cross multiple domains (e.g., health data and consumption habits).
- *Data quality.* The quality of the AI model goes hand in hand with the quality of the

training data. However, the anonymization of data, often used as a solution to the first problem, has the disadvantage of also reducing the relevance of the data and thus the quality of the model.

- *Integrity of the learning process.* The service that performs the learning computation could intentionally bias the data set or the learning algorithm to influence the AI in a way that serves its own interest. It is important to note that the main players in this field are companies whose business model is based on advertising, i.e. influence. Their interest in biasing an AI is therefore strong and will occur, voluntarily or not, so that the recommendations go in the direction of their business model.

The PDMS approach, in which each individual is equipped with a PDMS holding her personal data, allows to resolve the above listed problems:

- *Data confidentiality protection.* Raw data is no longer permanently exposed to sophisticated attacks on centralized servers. In a distributed approach, the data is only used during the learning phase of the AI model and in a secure manner (assuming a secure computation protocol is provided).
- *Ease for individuals to contribute.* The personal cloud paradigm, armed with the right to portability, makes it easy for individuals to retrieve their data. Therefore, the friction to contribute to the learning computation is limited to consent during a distributed computation (and not at the time of collection).
- *Data quality.* Assuming a secure distributed aggregation protocol, it is no longer necessary to degrade the data in order to anonymize it, thus improving the quality of the results.
- *Audit and proof of computation integrity.* In a centralized learning approach, users are dependent on large organizations without any means of control, with potential biases mentioned above. In a peer-to-peer approach, each PDMS is a source and potential computational node, or even a transmitter of the learning computation plan. This configuration allows both to democratize the learning computation and to provide guarantees on the computation itself and on the criteria used to select the participants to the latter. Eventually, the idea is to be able to audit the learning process and offer strong guarantees on its integrity.

However, this approach also raises significant challenges. Organizing secure and efficient distributed computing among PDMS nodes can be a difficult task, especially in the presence of a potentially large number of corrupted nodes. Moreover, learning an AI model in the same context remains an open and challenging question. In particular, it is necessary to decompose the learning computation into "atomic" tasks, minimizing the information needed to accomplish them, distributed over the execution actors. Thus, in conjunction with the random delegation of these tasks to nodes in the system (see Chapter 4), the risk of leakage becomes sub-linear with respect to the number of corrupted nodes in the system. The difficulty is compounded by the extension of a learning computation that potentially requires a large data set from each participant and potentially repeated computations a large number of times.

To tackle these challenges, I plan to follow an incremental approach. In a first step, we will focus on basic aggregation primitives before looking at more advanced machine learning algorithms. Aggregation primitives (e.g., sum or average) are obviously essential to compute basic statistics on user data but are also a fundamental building block for machine learning algorithms. To enable new usages on personal data, we need scalable, privacy-preserving protocols implementing data aggregation primitives with selective (i.e., consenting) participants. Ideally, the proposed protocol should provide an accurate result that fully takes advantage of

high-quality data available in PDMSs. Efficiency (i.e., protocol latency and total load of the system) is of prime importance and the protocol should adapt to several contexts: the PDMSs could be limited by their communication speed or by their computation power. Finally, given the scale of such decentralized aggregation, such protocols must also be robust to node failures.

To summarize, our first goal is to propose an aggregation protocol for basic aggregate functions that fulfills the following properties:

- highly scalable and fully decentralized (peer-to-peer),
- privacy-preserving (protecting the user data confidentiality),
- accurate (no privacy versus accuracy trade-off),
- adaptable (to PDMSs participation and characteristics),
- reliable (handles node failures or voluntary disconnections).

Secure aggregation is an intense research area since many years and many approaches have been proposed: secure multi-party computation (SMC) and (fully) homomorphic threshold encryption (HTE), (local) differential privacy and gossip-based protocols. However, the existing solutions are not adapted to the PDMS context and fail to cover all the required properties listed above.

HTE and SMC-based solutions [27, 39, 50] generally target applications in which central servers orchestrate and coordinate the participating nodes (e.g. federated learning). Such solutions are not scalable with a large number of participants and a fully decentralized setting such as in the PDMS context (e.g., the server(s) load is linear [39] or quadratic [27] with the number of participants).

Local differential privacy (LDP) has gained significant momentum in the recent years addressing problems such as machine learning [148] or basic statistics based on range queries [37]. However, LDP requires more noise than classical DP [7], either affecting accuracy or requiring a large number of participants to reduce the impact of noise, contradicting adaptability to selective participation.

Gossip-based protocols are scalable, fully decentralized, reliable and have an adjustable accuracy. Unfortunately, classical gossip-based protocols do not protect the user privacy. In [23], participants collectively learn a machine learning model in a privacy preserving way by gossiping differentially private models, impacting accuracy. In [91], participants introduce noise in the first iterations and gradually remove it in subsequent iterations. This approach makes such solutions unreliable w.r.t. node failures. Finally, we are not aware of gossip protocols tolerating selective participation and trivial adaptations produce inaccurate results.

Given the limitations of the existing solutions, I have started to work on a protocol allowing secure aggregation in a massively distributed PDMS environment which adapts to selective participation and PDMSs characteristics and is reliable with respect to node failures, with no compromise on accuracy. This research perspective is developed in the context of the PhD thesis of Julien Mirval which I co-advise with Luc Bouganim (see Appendix A).

Appendix A

Ph.D. Students

I have co-supervised 3 Ph.D. students and am currently supervising 2 theses.

A.1 Saliha Lallali

Saliha Lallali defended her Ph.D. thesis, entitled *A Scalable Search Engine for the Personal Cloud* [67], on January 28, 2016. This Ph.D. was co-supervised with Nicolas Anciaux and Philippe Pucheral and led to the following publications or communications: [12, 68, 69, 13].

Saliha Lalalli is an R&D engineer at C&S Group (Communication & Systèmes) in France.

A.2 Dai-Hai Ton-That

Dai-Hai Ton-That defended his Ph.D. thesis, entitled *Efficient Management and Secure Sharing of Mobility Traces* [127], on January 29, 2016. This Ph.D. was co-supervised with Karine Zeitouni and led to the following publications or communications: [128, 129, 130, 131, 136, 135].

Dai-Hai Ton-That was a postdoc at CEA (CEA-LIST Institute, CEA-Saclay, Paris Saclay University, France) from 2016 to 2017 and a postdoctoral researcher at CDM (the College of Computing and Digital Media at Depaul University, USA) from 2017 to 2020. He is now a research scientist at University of Alabama in Huntsville.

A.3 Julien Loudet

Julien Loudet defended his Ph.D. thesis, entitled *Distributed and Privacy-Preserving Personal Queries on Personal Clouds* [76], on October 24, 2019. This Ph.D. was co-supervised with Luc Bouganim and led to the following publications or communications: [79, 78, 77, 81, 80, 82].

After having fulfilled a one-year postdoc in the Petrus team, Julien Lodet works as an R&D engineer at AdLink in France.

A.4 Robin Carpentier

Robin Carpentier started his Ph.D. thesis in October 2018 on the topic of *Structures and Algorithms for Secure Data Management in a Trusted Execution Environment for the Personal Cloud*. He is co-supervised with Nicolas Anciaux and Guillaume Scerri. Preliminary results have been given in [31].

A.5 Julien Mirval

Julien Mirval started his Ph.D. thesis in November 2020 on the topic of *DISSEC-ML: Towards Distributed and Secured Machine Learning in the Personal Cloud*. He is co-supervised with Luc Bouganim.

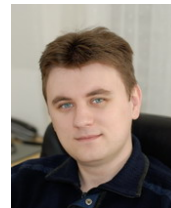
Appendix B

Curriculum Vitae

CURRICULUM VITAE – IULIAN SANDU POPA

PERSONAL INFORMATION

Iulian SANDU POPA
Born on July 19, 1981.
French Citizen, Romanian Citizen.
Married (Pacsé), 2 children.



CONTACT INFORMATION

Project PETRUS – Inria Saclay Ile-de-France, Université de Versailles Saint-Quentin (UVSQ)
David lab, bât. Buffon, UFR des Sciences, UVSQ
45 avenue des Etats-Unis
78035 Versailles Cedex
France

Phone : +33 1 39 25 40 85

Email : iulian.sandu-popa@uvsq.fr

Web : <http://www-smis.inria.fr/~isp/>

CURRENT SITUATION

Associate Professor (Maître de Conférences, Classe Normale), Computer Science Section (CNU 27) at UVSQ since September 2011. Permanent member of the *Personal and Trusted Cloud* (PETRUS) join-team (Inria, UVSQ) since September 2011.

RESEARCH THEME

My current research is focused on the **privacy and security of Personal Data Management Systems**. Specifically, I study different research problems revolving around : (i) decentralized architectures for privacy-preserving data management leveraging Trusted Execution Environments ; (ii) securing the data collection, storage and computations, and the interactions of the user's app with her data ; (iii) secure and efficient massively distributed computations over personal data.

Other Research

Spatiotemporal Databases, Mobile Data Management, Mobile Participatory Sensing.

Keywords

Databases, Personal Cloud, Privacy, Trusted Execution Environment, Secure Decentralized Computation, Secure Hardware.

EDUCATION

Politehnica University of Bucharest (Romanian top school in Computer Engineering) and **Université de Versailles Saint-Quentin** (UVSQ).

- **Doctor in Computer Science**, UVSQ, December 2009.
Ph.D. dissertation : *Modeling, Querying and Indexing Moving Objects with Sensors on Road Networks*.
Advisors : Karine Zeitouni and Georges Gardarin.
- **Master of Science**, UVSQ, September 2006.
- **Computer Science Engineer**, Politehnica University of Bucharest, September 2000 - August 2005.

Distinction

- **International Physics Olympiad**, Honourable Mention, Leicester, UK, July 2000.

SCIENTIFIC CAREER

- September 2011-... : Associate Professor at UVSQ, David lab, and member of the PETRUS team (former SMIS team) at Inria Saclay. Advisors : Karine Zeitouni and Georges Gardarin.
- February-March 2018 : Visiting researcher at New Jersey Institute of Technology (NJIT), New Jersey, working with Pr. Cristian Borcea and Pr. Vincent Oria.
- February-April 2011 : Visiting researcher at NJIT, New Jersey, working with Pr. Cristian Borcea and Pr. Vincent Oria.
- September 2010-February 2011 : Research Engineer, PRiSM lab, UVSQ.
- October 2009-August 2010 : Research and Teaching Assistant (ATER) at UVSQ.
- October 2006-September 2009 : PhD candidate at UVSQ, Data Integration and Management (DIM) team. Research assistant and instructor (Moniteur) at UVSQ. Doctoral grant from Ile-de-France region.

PH.D.
SUPERVISION

- Julien Mirval, started in November 2020, on the topic "DISSEC-ML : Towards Distributed and Secured Machine Learning in the Personal Cloud", supervisor Luc Bouganim. My supervision participation : 50%.
- Robin Carpentier, started in October 2018, on the topic "Structures and Algorithms for Secure Data Management in a Trusted Execution Environment for the Personal Cloud", supervisor Nicolas Anciaux. My supervision participation : 40%.
- Julien Loudet, defended on October 24th, 2019, entitled "Distributed and Privacy-Preserving Personal Queries on Personal Clouds", supervisor Luc Bouganim. My supervision participation : 50%.
- Dai-Hai Ton-That, defended on January 29th, 2016, entitled "Efficient Management and Secure Sharing of Mobility Traces", supervisor Karine Zeitouni. My supervision participation : 50%.
- Saliha Lallali, defended on January 28th, 2016, entitled "A Scalable Search Engine for the Personal Cloud", supervisor Philippe Pucheral. My supervision participation : 40%.

MSc AND
RESEARCH
INTERNSHIP
SUPERVISION

I have co-supervised 4 MSc student final thesis and 5 research internships. Among these, 5 students continued with a Ph.D. thesis under my co-supervision.

- MSc student final thesis : Julien Mirval (M2 Data Mining, Université Lyon2, 7 months, 2019), Robin Carpentier (M2 SeCRets, 6 months, 2017), Poulmanogo Illy (M2 Datascale, UVSQ, 7 months, 2017), Saliha Lallali (M2 IRS, UVSQ, 7 months, 2012).
- Research internships : Julien Mirval (pre-thesis internship, 1 year, 2019-2020), Razvan Nitu (Politehnica University of Bucharest, 5 months, 2016), Julien Loudet (Télécom ParisTech, 6 months, 2015-2016), Thu Le (Hô Chi Minh City University of Technology, Viêt-Nam, 4 months, 2015), Dai-Hai Ton-That (Hô Chi Minh City University of Technology, Viêt-Nam, 5 months, 2012).

RESEARCH
PROJECTS

ANR (*Agence Nationale de la Recherche*) is the French national research funding agency. H2020 RISE (Research and Innovation Staff Exchange) is an European funding instrument aiming to encourage international and intersectoral collaborations through exchanges of research and innovation staff. Digiteo-OMTE (*Opérations de Maturation Technico-Économique*) is a regional funding structure aiming to accelerate the transfer of scientific projects. RNTS (*Réseau National des Technologies pour la Santé*) aims to promote the application of new technologies in the field of health, around strategic themes, by supporting projects led by industry, researchers and clinicians.

- Projet H2020 RISE MASTER (MultipleASpects Trajectory management and analysis) (2018 - ...). Contributor to the privacy-preserving management of holistic trajectories task. Two secondments (at Universidade Federal de Santa Catarina, Brazil, and at University of Piraeus Research Centre, Greece) were planned for 2020 and 2021, but were postponed because of the COVID pandemic situation.
- ANR PerSoCloud (Personal and Social Trusted Cloud) (2017 - 2021). Contributor to reference architecture for the Personal Cloud task and the secure data sharing and distributed querying task.
- ANR KISS (Keeping your Information Safe and Secure) (2011 - 2015). Contributor to advanced spatiotemporal data and query models for secure tokens task, access methods to efficiently manage large document collections task, and large-scale distributed data management with trusted devices task.
- Digiteo-OMTE MobiScope (2009). Main designer of the software platform and principal programmer of the software stack.
- RNTS RHÉA (2004-2006). Contributor to the automatic classification of hospital reports task.

INDUSTRIAL
COLLABORATION

My research group is involved in a long term collaboration with Cozy Cloud, a startup developing Cozy (<https://cozy.io>), a privacy-friendly open-source Personal Cloud solution. I am involved in this industrial-academic partnership which is concretized in the form of research projects (e.g., ANR PerSoCloud 2017-2021) or industrial projects (e.g., PIA Secsi 2016-2017). Moreover, this collaboration has offered me the opportunity to co-advise two CIFRE (*Convention Industrielle de Formation par la Recherche*) Ph.D. thesis (of Julien Loudet, defended in October 2019, and of Julien Mirval, started in November 2020).

VISITING
RESEARCHER
HOSTING

- Pr. Cristian Borcea from New Jersey Institute of Technology (NJIT) was a CNRS visiting researcher at PRISM laboratory, UVSQ, in June-August 2011. Being on complementary topics - databases on my side and mobile computing on his - we worked on real-time routing (guidance) of road traffic and proposed an efficient method offering traffic balancing. The results have been published in an international conference and in two international reference journals in the field.
- Pr. Vincent Oria from NJIT was hosted as a visiting professor by RTRA DIGITEO in 2010. During his stay, we worked on the indexing of trajectories in a road network. This collaboration led to a paper at the ICDE conference in 2010, then to a publication in the international journal VLDBJ. This collaboration continued and we have subsequently published an article in the international journal Geoinformatica on a complementary topic, namely the compression of trajectories in a road network.

SCIENTIFIC &
ADMINISTRATIVE
SERVICE

*Organizing
Committees*

- IEEE International Conference on Mobile Data Management (IEEE MDM), juin 2020, Versailles. Organizing Co-chair.
- *Colloque National Capteurs et Sciences Participatives (CASPA)*, Paris, April, 2019. Member of the Scientific Committee.
- *Conférence sur la Gestion de Données - Principes, Technologies et Applications (BDA)*, Bucharest, Romania, October 2018. Organizing Co-chair.

*International
Conference
Program
Committees*

- International Conference on Management of Data (SIGMOD Conference) (2021)
- IEEE International Conference on Data Engineering (ICDE) (2019, 2020)
- Scientific and Statistical Database Management Conference (SSDBM) (2019, 2020, 2021)
- IEEE MobileCloud (since 2015)
- International Conference on Data Science, Technology and Applications (DATA) (since 2016)
- International Joint Conference APWeb-WAIM (2017)
- MobilWare (2012, 2013, 2015, 2016)

*International
Workshop PCs*

- Workshop on Fairness, Accountability, Transparency, Ethics and Society on the Web (FATES) (2021)
- International Workshop on Mobile Cloud Computing systems, Management, and Security (MCSMS) (2016)

*National
Conference PCs*

- *Conférence sur la Gestion de Données - Principes, Technologies et Applications (BDA)* 2019. Demonstration track.
- *Atelier sur la Protection de la Vie Privée (APVP)* (2015)

Journal Reviewer

- SIGMOD Record (2021)
- ISPRS International Journal of Geo-Information (2021)
- Journal of Internet Services and Applications (2019)
- ACM Transactions on Spatial Algorithms and Systems (2017, 2018)
- Transactions on Parallel and Distributed Systems (2018)
- GeoInformatica (2018)
- Journal of Intelligent Transportation Systems (2017)
- International Journal of Geo-Information (2017)
- IEEE Transactions on Knowledge and Data Engineering (2015, 2016)
- International Journal of Digital Earth (2016)
- ACM Transactions on Storage (2015)
- Information Systems (2012, 2013)
- Journal of Systems and Software (2013)
- Earth Science Informatics (2012)

*Other Conference
Activities*

- *Conférence sur la Gestion de Données - Principes, Technologies et Applications (BDA)* 2011. Webmaster.

- Ph.D. Committees
- Julien Loudet, *Distributed and Privacy-Preserving Personal Queries on Personal Clouds*, UVSQ, October 24th, 2019.
 - Dai-Hai Ton-That, *Efficient Management and Secure Sharing of Mobility Traces*, UVSQ, January 29th, 2016.
 - Saliha Lallali, *A Scalable Search Engine for the Personal Cloud*, UVSQ, January 28th, 2016.
 - Juan (Susan) Pan, *Vehicle Re-routing Strategies for Congestion Avoidance*, NJIT, December 18th, 2013.
 - Juan (Susan) Pan, *Vehicle Re-routing Strategies for Congestion Avoidance*, Ph.D. Proposal Defense, NJIT, April 5th, 2013.

- Selection Committees & Expertise
- Selection Committees are in charge of hiring permanent researchers (e.g., on Associate Professor positions). Expertise activity consists in reviewing research or international collaborative projects.
- Selection committee, Associate Professor position 0208, Conservatoire national des arts et métiers (CNAM), 2019.
 - Selection committee, Associate Professor position 4133, UVSQ, 2016.
 - ANR expert, evaluation committee CE23, 2016.
 - Expertise on bilateral exchange projects for COFECUB (Comité Français d'Evaluation de la Coopération Universitaire et Scientifique avec le Brésil), 2013.

- Inria Service
- The *Commission du Développement Technologique* (CDT) is in charge of (i) selecting the Inria project proposals demanding engineering support, validating the candidates identified by the teams, and evaluating the renewal of financial support, and (ii) assigning the *Services d'Expérimentation et de Développement* (SED) engineers to Inria teams requiring specific software development support.
- Member of the CDT commission for Inria Saclay Ile-de-France, since 2019.

- UVSQ Service
- The Board of the David computer science laboratory at UVSQ has the role of assisting the laboratory Chair in preparing the agenda before every General Assembly of the laboratory. The laboratory Chair may convene the Board, or consult it by email, for any other matter concerning the governance of the laboratory.
- Elected member of the David laboratory Board at UVSQ, since 2021.

SCIENTIFIC DISSEMINATION

- Tutorials, Invited Talks, Workshops, ...
- Tutorial at VLDB'2019 (International Conference on Very Large Data Bases) : N. Ancaux, L. Bouganim, P. Pucheral, I. Sandu Popa, G. Scerri : *Personal Database Security and Trusted Execution Environments : A Tutorial at the Crossroads*. Los Angeles, August 26-30, 2019.
 - Participation to the writing of the *Access Control* chapter of the *Inria white paper on Cybersecurity and major scientific challenges*, Inria, January 2019.
 - Invited talk : *Highly Distributed Queries on Personal Data Management Systems with Strong Privacy Guarantees*, New Jersey Institute of Technology (NJIT), April 9th 2018.
 - Invitation to present representative demos for the research of the Inria Saclay-Ile-de-France center at the inauguration of the Turing building, Inria Saclay-Ile-de-France, February 14th, 2017. At this event, I represented the Petrus team and presented the PlugDB demo.
 - BIS'2016 workshop <https://project.inria.fr/siliconvalley/workshops/bis2016/> : Iulian Sandu Popa, Inria SMIS & CityLab@Inria : *Distributed Architectures for Privacy-Aware Mobile Participatory Sensing*, Paris, June 8-10, 2016.
 - Invited tutorial at the international conference ADBIS (Advances in Databases and Information Systems) : N. Ancaux, B. Nguyen, I. Sandu Popa : *Towards an Era of Trust in Personal Data Management*, ADBIS'15, Poitier, France, September 2015.
 - Tutorial at EDBT'2014 (International Conference on Extending Database Technology) : Ancaux, N., Nguyen, B., Sandu Popa, I. : *Tutorial : Managing Personal Data with Strong Privacy Guarantees*. March 24-28, Athens, Greece, 2014.
 - Advanced seminar at MDM'2013 (IEEE International Conference on Mobile Data Management) : Ancaux, N., Nguyen, B., Sandu Popa, I. : *Personal Data Management with Secure Hardware : How to Keep Your Data at Hand*. Milan, Italy, June 3-6, 2013.

- Future en Seine : world digital festival exposing the latest French and international digital innovations to professionals and the general public. During this festival, I presented to the visitors our work on personal data protection using secure and portable devices. June 2013.
- Invited talk : *Integration, optimisation and analysis of data flows from mobile sensors*. NJIT, February 2011.
- Invited seminar : *Intégration, optimisation et analyse des flux de données de capteurs*. at LIRIS lab, INSA Lyon, April 2010 and at CNAM Paris, Mai 2010.

TEACHING

I have been teaching at least 192h per year at UVSQ since September 2011, except for the school years 2017/2018 and 2019/2020 when I had two half-year delegations at Inria Saclay and hence, I only taught 96h per year. The detailed statistics for these years are presented in the table below. Before, I taught 96h in 2009/2010 as a teaching assistant (ATER) and 64h per year during the three years of my Ph.D. period (i.e., October 2006 to August 2009) also at UVSQ. Finally, I taught 36h in 2004/2005 at Politehnica University of Bucharest (Computer Science department).

In my teaching activities I have been involved in various courses for a varied public - from first year undergraduate students to last year master students, at the University (UVSQ) or in Engineering Schools (ISTY at UVSQ, ENSIIE in Evry, or Politehnica in Bucharest). However, a large part of my teaching load is allocated to the courses listed below, which allows me to match the demand for courses in the Computer Science department of the UFR des Sciences (UVSQ) with the specificity of my profile :

Main Taught Courses

1. Databases, 3rd year undergraduate students, the Computer Science department.
2. Initiation to Databases, 2nd year undergraduate students, Mathematics, Mathematics-Physics and Mathematics Applied to Human and Social Sciences departments.
3. Relational Databases and Logic, 1st year master students, Datascale MSc.
4. Spatio-temporal Data Management and Analysis, 2nd year master students, Datascale MSc.
5. Security of Personal and Corporate Data, 2nd year master students, Datascale MSc.

I am the leader of module (1) and co-leader of module (4). I have also been the leader of module (2) for 7 years.

Year	Total	Courses	Labs	Undergraduate	Master
2020-21	234.75	78.75	156	180	54.75
2019-20*	96	51	45	63	33
2018-19	192	27	165	132	60
2017-18*	111	45	66	65.25	45.75
2016-17	207.5	45	140	114.75	92.75
2015-16	203	42	140	129	74
2014-15	220.5	42	157.5	112.5	108
2013-14	212	36	158	113	99
2012-13	225.25	22.5	191.5	102.75	122.5
2011-12	202.5	9	189	126	76.5

*Inria half-year delegation during the 2017/2018 and 2019/2020

Appendix C

Personal Bibliography

Overview

This section provides the list of my publications regrouped in four categories: international journals, French journals, international conferences and French conferences and workshops (see the table below). The international and the French conferences include research papers, demonstration papers and tutorials. The demonstration and tutorials are indicated as such in the list. The same goes for the invited papers (tutorials). Please note that top level conferences in Databases are equivalent to top ranked journals (A/A*), generally having acceptance ratios under 15% (the demonstration papers also being selective). The CORE ranking is indicated whenever this information is available for both international journals and conferences. All publications except those indicated as invited (i.e., one tutorial at ADBIS'15) have undergone a peer-review process. The speaker (or speakers for some demonstration papers or tutorials) is indicated for all the international and French conferences.

Publication Type	Total
International Journals	8
French Journals	2
International Conferences	20
French Conferences and Workshops	14

International Journals

Iulian Sandu Popa, Dai Hai Ton That, Karine Zeitouni, and Cristian Borcea. “Mobile participatory sensing with strong privacy guarantees using secure probes”. In: *GeoInformatica, online first* (2019), p. 48

Nicolas Anciaux, Philippe Bonnet, Luc Bouganim, Benjamin Nguyen, Philippe Pucheral, Iulian Sandu Popa, and Guillaume Scerri. “Personal Data Management Systems: The security and functionality standpoint”. In: *Inf. Syst.* 80 (2019), pp. 13–35 (CORE: A*).

Saliha Lallali, Nicolas Anciaux, Iulian Sandu Popa, and Philippe Pucheral. “Supporting secure keyword search in the personal cloud”. In: *Inf. Syst.* 72 (2017), pp. 1–26 (CORE: A*).

Susan Juan Pan, Iulian Sandu Popa, and Cristian Borcea. “DIVERT: A Distributed Vehicular Traffic Re-Routing System for Congestion Avoidance”. In: *IEEE Trans. Mob. Comput.* 16.1 (2017), pp. 58–72 (CORE: A*).

Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, and Ahmed Kharrat. “Spatio-temporal compression of trajectories in road networks”. In: *GeoInformatica* 19.1 (2015), pp. 117–145

Dai Hai Ton That, Iulian Sandu Popa, and Karine Zeitouni. “TRIFL: A Generic Trajectory Index for Flash Storage”. In: *ACM Trans. Spatial Algorithms and Systems* 1.2 (2015), 6:1–6:44

Susan Juan Pan, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. “Proactive Vehicular Traffic Rerouting for Lower Travel Time”. In: *IEEE Trans. Vehicular Technology* 62.8 (2013), pp. 3551–3568

Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, Dominique Barth, and Sandrine Vial. “Indexing in-network trajectory flows”. In: *VLDB J.* 20.5 (2011), pp. 643–669 (CORE: A*).

French Journals

Dai Hai Ton That, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. “Un protocole basé sur des mobiles sécurisés pour une collecte participative de données spatiales en mobilité réellement anonyme”. In: *Revue Internationale de Géomatique* 26.2 (2016), pp. 185–210

Iulian Sandu Popa, Ahmed Kharrat, Karine Zeitouni, and Guillaume Saint-Pierre. “Base de données de capteurs à localisation mobile Modèle et langage”. In: *Ingénierie des Systèmes d’Inf.* 14.5 (2009), pp. 35–58

International Conferences (Papers, Demos and Tutorials)

Nicolas Ancaux, Luc Bouganim, Philippe Pucheral, Iulian Sandu Popa, and Guillaume Scerri. “Personal Database Security and Trusted Execution Environments: A Tutorial at the Crossroads”. In: *Proc. VLDB Endow.* 12.12 (2019), pp. 1994–1997 (Tutorial) (CORE: A*) (Speakers: Nicolas Ancaux, Iulian Sandu Popa, Guillaume Scerri).

Julien Loudet, Iulian Sandu Popa, and Luc Bouganim. “SEP2P: Secure and Efficient P2P Personal Data Processing”. In: *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. OpenProceedings.org, 2019, pp. 145–156 (CORE: A) (Speaker: Julien Loudet).

Julien Loudet, Iulian Sandu Popa, and Luc Bouganim. “DISPERS: Securing Highly Distributed Queries on Personal Data Management Systems”. In: *PVLDB* 12.12 (2019), pp. 1886–1889 (Demonstration Track) (CORE: A*) (Speaker: Julien Loudet).

Dai Hai Ton That, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. “PAMPAS: Privacy-Aware Mobile Participatory Sensing Using Secure Probes”. In: *Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM 2016, Budapest, Hungary, July 18-20, 2016*. ACM, 2016, 4:1–4:12 (CORE: A) (Speaker: Dai Hai Ton That).

Thu Le, Nicolas Ancaux, Sébastien Guilloton, Saliha Lallali, Philippe Pucheral, Iulian Sandu Popa, and Chao Chen. “Distributed Secure Search in the Personal Cloud”. In: *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016*. OpenProceedings.org, 2016, pp. 652–655 (Demonstration Track) (CORE: A) (Speaker: Iulian Sandu Popa).

Nicolas Ancaux, Saliha Lallali, Iulian Sandu Popa, and Philippe Pucheral. “A Scalable Search Engine for Mass Storage Smart Objects”. In: *Proc. VLDB Endow.* 8.9 (2015), pp. 910–921 (CORE: A*) (Speaker: Iulian Sandu Popa).

Saliha Lallali, Nicolas Ancaux, Iulian Sandu Popa, and Philippe Pucheral. “A Secure Search Engine for the Personal Cloud”. In: *ACM SIGMOD*. ACM, 2015, pp. 1445–1450 (Demonstration Track) (CORE: A*) (Speakers: Saliha Lallali, Iulian Sandu Popa).

Dai Hai Ton That, Iulian Sandu Popa, and Karine Zeitouni. “PPTM: Privacy-Aware Participatory Traffic Monitoring Using Mobile Secure Probes”. In: *16th IEEE International Conference on Mobile Data Management, MDM 2015*. IEEE Computer Society, 2015, pp. 295–298 (Demonstration Track) (CORE: C) (Speaker: Dai Hai Ton That).

Nicolas Ancaux, Benjamin Nguyen, and Iulian Sandu Popa. “Tutorial: Towards an Era of Trust in Personal Data Management”. In: *Proceedings of the 19th East-European Conference on Advances in Databases and Information Systems (ADBIS '15)*. Poitiers, France. 2015 (Invited Tutorial) (CORE: B) (Speakers: Nicolas Ancaux, Benjamin Nguyen).

Nicolas Ancaux, Benjamin Nguyen, and Iulian Sandu Popa. “Tutorial: Managing Personal Data with Strong Privacy Guarantees”. In: *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*. Open-Proceedings.org, 2014, pp. 672–673 (Tutorial) (CORE: A) (Speakers: Nicolas Ancaux, Iulian Sandu Popa).

Nicolas Ancaux, Benjamin Nguyen, and Iulian Sandu Popa. “Personal Data Management with Secure Hardware: How to Keep Your Data at Hand”. In: *2013 IEEE 14th International Conference on Mobile Data Management, Milan, Italy, June 3-6, 2013 - Volume 2*. IEEE Computer Society, 2013, pp. 1–2 (Tutorial) (CORE: C) (Speakers: Nicolas Ancaux, Benjamin Nguyen, Iulian Sandu Popa).

Nicolas Ancaux, Philippe Bonnet, Luc Bouganim, Benjamin Nguyen, Iulian Sandu Popa, and Philippe Pucheral. “Trusted Cells: A Sea Change for Personal Data Services”. In: *Sixth Biennial Conference on Innovative Data Systems Research, CIDR 2013, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*. www.cidrdb.org, 2013 (CORE: A) (Speaker: Philippe Bonnet).

Susan Juan Pan, Mohammad A. Khan, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. “Proactive Vehicle Re-routing Strategies for Congestion Avoidance”. In: *IEEE 8th International Conference on Distributed Computing in Sensor Systems, DCOSS 2012*. IEEE Computer Society, 2012, pp. 265–272 (CORE: B) (Speaker: Susan Juan Pan).

Iulian Sandu Popa and Karine Zeitouni. “Modeling and Querying Mobile Location Sensor Data”. In: *The Fourth International Conference on Advanced Geographic Information Systems, Applications, and Services (GEOProcessing 2012)*. 2012, 18pp (Speaker: Karine Zeitouni).

Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, Dominique Barth, and Sandrine Vial. “PARINET: A tunable access method for in-network trajectories”. In: *ICDE*. IEEE Computer Society, 2010, pp. 177–188 (CORE: A*) (Speaker: Iulian Sandu Popa).

Ahmed Kharrat, Karine Zeitouni, Iulian Sandu Popa, and Sami Faïz. “Characterizing the Traffic Density and Its Evolution through Moving Object Trajectories”. In: *KDIR 2009 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*. INSTICC Press, 2009, pp. 319–322 (Speaker: Ahmed Kharrat).

Ahmed Kharrat, Karine Zeitouni, Iulian Sandu Popa, and Sami Faiz. “Characterizing Traffic Density and Its Evolution through Moving Object Trajectories”. In: *Fifth International Conference on Signal-Image Technology & Internet-Based Systems, SITIS 2009*. IEEE Computer Society, 2009, pp. 257–263 (Speaker: Ahmed Kharrat).

Iulian Sandu-Popa, Karine Zeitouni, G. Saint-Pierre, F. Dupin, and S. Glaser. “Using in-Vehicle Sensor Data for Naturalistic Driving Analysis”. In: *World Congress on Intelligent Transportation Systems*. United States, 2008, p. 6 (Speaker: Iulian Sandu Popa).

Ahmed Kharrat, Iulian Sandu Popa, Karine Zeitouni, and Sami Faiz. “Clustering Algorithm for Network Constraint Trajectories”. In: *Headway in Spatial Data Handling, 13th International Symposium on Spatial Data Handling*. Springer, 2008, pp. 631–647 (CORE: C) (Speaker: Ahmed Kharrat).

Iulian Sandu Popa, Karine Zeitouni, Georges Gardarin, Didier Nakache, and Elisabeth Métais. “Text Categorization for Multi-label Documents and Many Categories”. In: *20th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2007)*. IEEE Computer Society, 2007, pp. 421–426 (Speaker: Iulian Sandu Popa).

French Conferences & Workshops

Julien Loudet, Iulian Sandu-Popa, and Luc Bouganim. “SEP2P: Secure and Efficient P2P Personal Data Processing”. In: *BDA 2019 - 35ème Conférence sur la Gestion de Données - Principes, Technologies et Applications*. Lyon, France, 2019 (Speaker: Julien Loudet).

Julien Loudet, Iulian Sandu-Popa, and Luc Bouganim. “DISPERS: Securing Highly Distributed Queries on Personal Data Management Systems”. In: *BDA 2019 - 35ème Conférence sur la Gestion de Données - Principes, Technologies et Applications*. 2019, Demo paper. (Speaker: Julien Loudet).

Julien Loudet, Iulian Sandu-Popa, and Luc Bouganim. “SEP2P: Secure and Efficient P2P Personal Data Processing”. In: *APVP 2019 - Atelier sur la Protection de la Vie Privée*. Cap Hornu, France, 2019 (Speaker: Julien Loudet).

Julien Loudet, Luc Bouganim, and Iulian Sandu Popa. “Privacy-Preserving Queries on Highly Distributed Personal Data Management Systems”. In: *34ème Conférence sur la Gestion de Données – Principes, Technologies et Applications*. Proceedings of the BDA 2018 Conference. Bucharest, Romania, 2018 (Speaker: Julien Loudet).

Robin Carpentier, Nicolas Anciaux, Iulian Sandu Popa, and Guillaume Scerri. “Performance of Large Scale Data-Oriented Operations under the TEE Constraints”. In: *34ème Conférence sur la Gestion de Données – Principes, Technologies et Applications (BDA 2018)*. Bucharest, Romania, 2018 (Speaker: Robin Carpentier).

Nicolas Anciaux, Saliha Lallali, Iulian Sandu-Popa, and Philippe Pucheral. “A scalable search engine for mass storage smart objects”. In: *31èmes journées Bases de Données Avancées (BDA)*. Île de Porquerolles, France, 2015 (Speaker: Saliha Lallali).

Dai Hai Ton That, Iulian Sandu-Popa, Karine Zeitouni, and Cristian Borcea. “PAMPAS: Collecte participative respectueuse de la vie privée basée sur des mobiles sécurisés”. In: *31èmes*

journées Bases de Données Avancées (BDA '15). Île de Porquerolles, France, 2015 (Speaker: Dai Hai Ton That).

Dai Hai Ton That, Iulian Sandu-Popa, and Karine Zeitouni. “PPTM: Privacy-aware Participatory Traffic Monitoring Using Mobile Secure Probes”. In: *31èmes journées Bases de Données Avancées (BDA '15)*. Île de Porquerolles, France, 2015, 4 pages (Demo paper) (Speaker: Dai Hai Ton That).

Georges Gardarin, Benjamin Nguyen, Laurent Yeh, Karine Zeitouni, Bogdan Butnaru, and Iulian Sandu-Popa. “Gestion efficace de séries temporelles en P2P: Application à l’analyse technique et l’étude des objets mobiles”. In: *Bases de Données Avancées*. Namur, Belgium, 2009 (Speaker: Benjamin Nguyen).

Iulian Sandu-Popa and Karine Zeitouni. “Modélisation et interrogation de données de capteurs à localisation mobile”. In: *Bases de Données Avancées*. Namur, Belgium, 2009 (Speaker: Karine Zeitouni).

Karine Zeitouni, Iulian Sandu-Popa, and Ahmed Kharrat. “Exploitation des traces sur la mobilité”. In: *Conférence francophone sur les Technologies de l’Information, de la Communication et de la Géolocalisation dans les Systèmes de Transports*. 2009 (Speaker: Iulian Sandu Popa).

Ahmed Kharrat, Iulian Sandu Popa, Karine Zeitouni, and Sami Faïz. “Caractérisation de la densité de trafic et de son évolution à partir de trajectoires d’objets mobiles”. In: *Actes des 5èmes journées francophones Mobilité et Ubiquité 2009, UBIMOB’09, 7-8 Juillet 2009, Lille, France*. Vol. 394. ACM International Conference Proceeding Series. ACM, 2009, pp. 33–40 (Speaker: Ahmed Kharrat).

Iulian Sandu-Popa, A. Kharrat, and Karine Zeitouni. “CALM : Un système de gestion de données de CAPteurs à Localisation Mobile”. In: *Journées SAGEO*. Montpellier, France, 2008 (Speaker: Iulian Sandu Popa).

Iulian Sandu-Popa, A. Kharrat, Karine Zeitouni, F. Dupin, and G. Saint-Pierre. “Gestion de données spatiotemporelle des observations sur la conduite en situation naturelle”. In: *2ème atelier SIT (Systèmes d’Information en Transport)*. Fontainebleau, France, 2008 (Speaker: Iulian Sandu Popa).

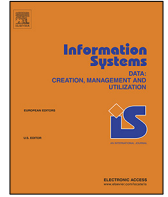
Appendix D

Personal Data Management Systems: The security and functionality standpoint



Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Personal Data Management Systems: The security and functionality standpoint



Nicolas Anciaux^{a,b,*}, Philippe Bonnet^c, Luc Bouganim^{a,b}, Benjamin Nguyen^{d,e},
Philippe Pucheral^{a,b}, Iulian Sandu Popa^{a,b}, Guillaume Scerri^{a,b}

^a INRIA Saclay-Ile-de-France, Université Paris-Saclay, 1 Rue H. d'Estienne d'Orves, 91120 Palaiseau, France

^b University of Versailles Saint-Quentin-en-Yvelines, Université Paris-Saclay, 45 avenue des Etats-Unis, 78035, Versailles Cedex, France

^c IT University of Copenhagen, Copenhagen, Denmark

^d INSA Centre Val de Loire, 88 bd Lahitolle, 18022 Bourges, France

^e Université d'Orléans, France

HIGHLIGHTS

- Review and comparison of various Personal Data Management Systems (PDMS) in terms of functionality provided and targeted threats.
- Definition of an extensive (combining all features) and secure (bypassing all threats) PDMS.
- Reference architecture for extensive and secure PDMSs meeting the given definition and security properties.
- Research challenges related to PDMS architectures.

ARTICLE INFO

Article history:

Received 27 July 2018

Received in revised form 11 September 2018

Accepted 12 September 2018

Available online xxxx

Keywords:

Secure personal cloud

Trusted execution environments

ABSTRACT

Riding the wave of smart disclosure initiatives and new privacy-protection regulations, the Personal Cloud paradigm is emerging through a myriad of solutions offered to users to let them gather and manage their whole digital life. On the bright side, this opens the way to novel value-added services when crossing multiple sources of data of a given person or crossing the data of multiple people. Yet this paradigm shift towards user empowerment raises fundamental questions with regards to the appropriateness of the functionalities and the data management and protection techniques which are offered by existing solutions to laymen users. These questions must be answered in order to limit the risk of seeing such solutions adopted only by a handful of users and thus leaving the Personal Cloud paradigm to become no more than one of the latest missed attempts to achieve a better regulation of the management of personal data. To this end, we review, compare and analyze personal cloud alternatives in terms of the functionalities they provide and the threat models they target. From this analysis, we derive a general set of functionality and security requirements that any Personal Data Management System (PDMS) should consider. We then identify the challenges of implementing such a PDMS and propose a preliminary design for an extensive and secure PDMS reference architecture satisfying the considered requirements. Finally, we discuss several important research challenges remaining to be addressed to achieve a mature PDMS ecosystem.

© 2018 Elsevier Ltd. All rights reserved.

Contents

1. Introduction.....	14
2. Existing personal cloud solutions	15
2.1. Online personal cloud solutions.....	15
2.2. Zero-knowledge-based personal clouds.....	16
2.3. Home cloud software	17
2.4. Home cloud plugs.....	18
2.4.1. Tamper-resistant home cloud.....	18

* Corresponding author at: University of Versailles Saint-Quentin-en-Yvelines, Université Paris-Saclay, 45 avenue des Etats-Unis, 78035, Versailles Cedex, France.
E-mail address: nicolas.anciaux@inria.fr (N. Anciaux).

2.5.	Synthesis of the existing approaches	19
3.	Definition of an Extensive and Secure Personal Data Management Systems (ES-PDMS)	19
3.1.	Specificities of data management in the PDMS context	19
3.2.	Security properties of a PDMS	21
3.3.	Extensive and secure personal data management systems	22
4.	Extensive and secure PDMS architecture	23
4.1.	Logical architecture	23
4.2.	Building blocks	23
4.2.1.	Piped data collection	24
4.2.2.	Mutual data at rest protection	24
4.2.3.	Bilaterally trusted personal computations	25
4.2.4.	Mutually trusted collective computations	26
4.2.5.	Controlled data dissemination	27
4.3.	Overall architecture design	28
4.4.	Concrete PDMS instances	28
4.5.	Discussion	31
5.	Research challenges towards an ES-PDMS architecture	32
6.	Conclusion	33
	Acknowledgments	34
	References	34

1. Introduction

Behaviors, movements, social relationships and interests of individuals are now constantly recorded, evaluated and analyzed in real-time by a small number of data aggregator companies [1]. This concentration negatively impacts privacy preservation and self-determination of individuals as well as innovation and fair competition between companies. Smart disclosure initiatives (e.g., Blue and GreenButton in the US,¹ Midata in the UK,² MesInfos in France³) are rising worldwide, aiming to restore individuals' control over their data and improve fairness in personal data management practices. The smart disclosure principle allows individuals to freely retrieve their personal data through a simple click, in a computer readable format, from the companies and administrations hosting them. According to the US government, this is a means to “help consumers make more informed choices; give them access to useful personal data; power new kinds of digital tools, products, and services for consumers; and promote efficiency, innovation, and economic growth” [2]. This fundamental principle has been recently translated into law with the *right to data portability* of the European General Data Protection Regulation (GDPR) [3].

Not only does smart disclosure allow individuals to be aware of the information collected about them, it also holds the promise of new services of high social and societal interest [2]. Indeed, individuals can now gather their complete digital environment in a so-called *Personal Cloud* or Personal Information Management Systems [4], Personal Data Server [5] or Personal Data Store [6]. A Personal Cloud is not only composed of data from many (previously) isolated information silos (e.g. secondary copies of data issued by their bank, employer, supermarket, hospital) but also of primary data (e.g. produced by quantified-self devices and smart meters, photos taken with their smartphone or documents stored on their PC). This unprecedented concentration of personal data opens the way for new value-added services when crossing multiple data of a given person (e.g., crossing medical data with eating patterns or bank statements with shopping history) or crossing the data of multiple people (e.g., conducting an epidemiological study), all this under the concerned individual's control. While this will certainly not stop data aggregator companies' current practices, the Personal

Cloud introduces an alternative way to develop *fairer* personal data management services using richer personal data. Hence, smart disclosure and the related Personal Cloud concept have become the cornerstone of what is called today *user empowerment*.

However, we should be cautious of a potential boomerang effect of user empowerment : returning individual's their data without providing them with the appropriate environment to exercise their control over it. Several companies are now riding the Personal Cloud wave and the spectrum of proposals in the internet sphere is highly diverse. *Online personal cloud* solutions (e.g., CozyCloud, Digi.me, BitsAbout.Me to only cite a few) propose a centralized web hosting of personal data combined with a rich set of services to collect personal data from various sources, store them and cross-exploit them. This approach assumes, by construction, that individuals do not question the honesty of the hosting company (including the honesty of the employees) nor its capacity to defeat severe attacks, since centralization creates by essence a massive honeypot. *Zero-knowledge personal cloud* solutions (e.g., SpiderOak [7], Sync) mitigate this strong trust assumption by offering a fully encrypted (yet still centralized) data store, but impose a new responsibility (managing the encryption keys) on the individuals, thus trading user friendliness and rich services for improved security. *Home cloud software* solutions (e.g., OpenPDS [6], DataBox [8,9]) build upon the paradigm of local/edge computing by considering that the raw personal data should remain stored physically close to the user. Hence, they advocate for a decentralized approach where individuals install local personal servers on their own equipment (e.g., PC). *Home cloud plugs* (e.g., CloudLocker, Helixee) go further in this direction by offering a dedicated box that can store TBs of data, run a server and simply be plugged on an individual's home internet gateway, alleviating the burden of installing and administering a server. Finally *tamper-resistant home clouds* are home cloud plugs integrating secure chips on their hardware board to improve their resistance to confidentiality attacks, viruses and ransoms. All these alternatives belong to the large, fuzzy, personal cloud system family but neither provide the same set of functionalities nor consider the same threat model.

This diversity of solutions raises important questions. Which functionalities are really mandatory in the personal cloud context? Which threat model better captures the various uses and architectural environments of the personal cloud? Do solutions exist combining the required set of functionalities and appropriate threat model? If not, where does the difficulty stem from? Can solutions be devised by adapting existing corporate cloud-based techniques or is the personal cloud problem fundamentally different, thus imposing a deep rethink of these techniques? These questions are

¹ <https://www.healthit.gov/topic/health-it-initiatives/blue-button>.

² <https://www.gov.uk/government/news/the-midata-vision-of-consumer-empowerment>.

³ <http://mesinfos.fing.org/>.

important for the data management research community and for the individual as well. By leaving these questions without answers, the risk is high to see the Personal Cloud paradigm be nothing but a missed attempt to reach a better regulation of the management of personal data, and maybe one of the last.

This paper precisely tries to answer these questions by making the following contributions:

- *Personal cloud solutions categorization*: we review, compare and categorize the various personal cloud alternatives sketched above in terms of provided functionalities and targeted threat model. The functionality axis is organized so as to cover all important steps of the personal data life cycle, from collection to storage, recovery, individual and collective exploitation of personal data. The threat model axis tries to encompass all actors contributing to the use of a personal cloud platform, from the personal cloud service provider, the applications providers, to the individual and its storage and computing environment. Beyond identifying the main expected features and privacy threats that need to be addressed, we also show that existing alternatives do not cover all of these features and threats, and cannot be combined for this purpose.
- *Definition of an extensible and secure personal data management system (ES-PDMS)*: we propose a definition of what an *extensive* (combining all functionalities) and *secure* (circumventing all the threats) *Personal Data Management System* should be. Therefore, we analyze the specificities of each functionality in the light of the individual context considered in this paper, and we deduce the corresponding security properties to achieve them. This has not been done until now, as existing personal cloud solutions have mostly been derived from their corporate counterparts.
- *Definition of an ES-PDMS reference architecture*: we have the strong belief that many security issues are rooted in architectural choices. Thus, we propose an abstract design for an *ES-PDMS* reference architecture satisfying the properties we will have defined. We then illustrate how this abstract architecture can be instantiated in different concrete settings. For the sake of generality, this design makes no assumption on the PDMS data management model itself (models and languages to define, manipulate and share objects entering in a PDMS).
- *Research issues related to PDMS architectures*: finally, we review a set of important research issues which remain to be investigated concerning the definition and security of PDMS architectures.

The rest of the paper follows this same structure, one section being devoted to each contribution, followed by a conclusion highlighting the expected impact of this work.

2. Existing personal cloud solutions

The Personal Cloud concept originally appeared under different names such as *Personal Information Management Systems* [4], *Personal Data Server* [5] or *Personal Data Store* [6]. It attracts today significant attention from both the research and industrial communities. This section provides a review of existing personal cloud solutions, representative of current approaches, to help understand the fundamental aspects in terms of functional requirements and security/privacy threats.

We distinguish cloud data management solutions designed for the corporate/enterprise context from those targeting individuals, i.e., tailored for personal use and referred to as Personal Clouds in this section. Corporate cloud solutions offer digital tools to

employees including, e.g., file storage space, email/agenda applications, file sharing with other employees based on permissions, user management and authentication based on LDAP repositories. They come as enterprise cloud tools either implemented as-a-service in the cloud or hosted on a server owned by the company and managed by internal administrators, such as SeaFile, Pydio, ownCloud/NextCloud, Sandstorm or Tonido/FileCloud. In terms of data management, the primary foci are multi-user features, workspace and user management, authentication, access control, privilege settings, administration and analysis tools. Such solutions do not apply to the personal cloud case and hence are not further detailed in this state of the art. Showing to which extent and detailing the major differences between individual and corporate-oriented solutions, is precisely one of the goals of the paper.

In contrast, solutions tailored for personal (and generally private) use are mono-user and seek to help users manage their entire digital life, i.e., by providing connectors to external data sources (e.g., bank, hospital, employer, social network, etc.), by allowing cross-data computation usages (e.g., linking the bank records of the individual with corresponding bills and email confirmations) and community uses based on groups of users sharing data for a social benefit (e.g., epidemiological study in a community of patients), by permitting their installation and configuration by laymen, and by helping individuals (rather than IT experts or administrators) understand and control data dissemination.

In the rest of this section, we thus deliberately focus on solutions tailored for individuals. We first review the online personal cloud solutions resorting to a personal cloud provider or remote storage service, then present variants offering additional security guarantees including zero-knowledge data stores, and finally analyze some more decentralized ‘home cloud’ proposals where the data is stored user-side, using purely software-based solution, hardware plugs or tamper resistant devices. We conclude this overview with a summary of the main features and security/privacy threats considered by existing solutions. This state-of-the-art analysis provides the needed material to derive in the next section the main underlying data management functionalities and related security goals of any extensive (in terms of database functionalities) and secure PDMS.

2.1. Online personal cloud solutions

Many online personal cloud solutions flourish today such as CozyCloud, [Digi.me](#), Meeco, [BitsAbout.Me](#) or Camilistore/Perkeep to name a few. Governmental programs like [MyData.org](#) in Finland, [MesInfos.fing.org](#) in France or in the UK, target the same objective. These initiatives provide online personal cloud solutions to help users gather and store all their personal data in the same place and in a usable format, with the possibility to cross-exploit it through various applications. In terms of privacy and security, a common claim of these solutions is to proscribe any secondary usage (and in particular monetization of personal data) by the personal cloud provider and to guarantee to their users that their personal data is never disclosed to third parties except on their explicit request. The main functionalities advertised and the corresponding privacy promises of such systems are further described below.

Data collectors. Most of the solutions mentioned above build on recent regulations like the GDPR (including the EU ‘right to personal data portability’) or smart disclosure initiative (e.g., Blue and GreenButton in US), and provide *data collectors* which can automatically feed a personal cloud with user’s data originating from different online services. Data collectors act as data bridges, which retrieve personal data on the behalf of the user (i.e., using her credentials) from external data sources. Usually, data collectors rely on web scrapping technologies (e.g., Cheerio, Weboob) which act on the behalf of the user on a target website to collect their

personal data. For example, CozyCloud (through the ‘CozyCollect’ application) and [Digi.me](#) provide their users with a catalog of connectors to retrieve many kinds of personal data, including financial data (e.g., from banks or PayPal), administrative data (e.g., electricity or telco bills and consumption traces, insurance contracts), social network data (e.g., from Facebook or Pinterest accounts), music (e.g., Spotify), medical information (e.g., from social security institutions, hospitals, or Blue Button compliant sites) or fitness data (e.g., Fitbit), to cite only a few. [BitsAbout.Me](#) also provides data collectors but focusses instead on the web activity trails of the user and targets personal information such as Google geolocation histories, Facebook ‘likes’ graphs and YouTube histories.

Cross-data computation services. Online personal cloud solutions allow integrating personal data usually scattered across distinct and closed data silos, which opens the way for novel applications able to cross-exploit it. [Digi.me](#) provides transversal data services, transversal data searches, and data sharing between different apps. In CozyCloud, the apps interact with the Cozy data system to access documents stored in a CouchDB engine, where each document is stored in a JSON format with an associated ‘doctype’ family (e.g., bank, photos, bills, etc.). The list of existing doctypes are defined and published by Cozy such that Cozy app developers can conform to a common scheme and easily identify the documents of interest (e.g., a finance app may be granted access to all the ‘bank’ and ‘bills’ doctypes and cross-exploit them to link each bill to a corresponding bank transaction record). In Meeco, personal data is organized within *life-tiles*, i.e., datasets defined by the user which gather specific personal information or files uploaded by the user (e.g., photos taken around Christmas in a life-tile entitled ‘Christmas’), and *web-tiles*, i.e., user defined goals (e.g., purchase intents) or user’s activities on specified websites (e.g., amazon.com).

Trusted data storage. The personal data associated with a given personal cloud user is stored online, but within a data store which belongs to a single user. In CozyCloud and Meeco, the user’s personal cloud is hosted and secured by the cloud service provider on behalf of the personal cloud owner (e.g., using server-side encryption). For instance, Meeco has chosen a cloud server in Australia to comply with the strict local privacy regulations, while Cozy can be deployed using any cloud service provider. In some cases (e.g. Cozy), expert users can opt for a self-hosted instance, which is close to the home cloud approach considered in Section 2.2. In [Digi.me](#), data is encrypted and stored where the user wishes (e.g., on Dropbox, Google Drive or MS OneDrive). Encryption keys are stored on a user’s device and derived from a user password. Keys are unlocked at connection time and used server side by data collectors and personal applications and are only retained for the duration of the session. In [BitsAbout.Me](#), a *Personal Data Store* is dedicated to each user and is hosted encrypted in a data-center in the EU or Switzerland (chosen for the high level of legal privacy protection offered by the GDPR and Swiss laws). Camilstore/Perkeep has a rather different approach as its goal is to provide a personal storage for life (above 100 years) to individuals. The solution thus focuses on providing users with easy means to generate a (searchable) personal data archive (storing all their, e.g., Tweets, social media photos, etc.) on a personal (device or cloud) store, independently of the websites hosting the data (e.g., Twitter, Instagram, etc.).

Trust model. All the above-mentioned solutions make strong privacy promises to the users in order to gain their trust. In particular, the personal cloud provider commits to never observe nor exploit the users’ personal data for *secondary usages* not advertised to the personal cloud owner such as data monetization. Although the different proposals vary widely in the way the personal cloud provider effectively tries to gain the users’ trust, it mainly relies on three arguments. The first argument is linked to the trust users

may put in the *security standards* of authentication, communications and data encryption. For instance, [Digi.me](#) authenticates applications, encrypts communication channels using SSL and encrypts passwords with RSA 2048-bit (FIPS compliant). CozyCloud also follows the current security best practices in terms of users’ passwords, connections and data at rest encryption. [BitsAbout.Me](#) declares that the decryption keys are expunged from its servers at user’s disconnection and therefore the server can only access the data during the time of a session. The second argument is related to the *virtuous legal and economic frameworks* to which the cloud provider is bound. Typically, most providers underline a high degree of independence enacted by contract between the storage provider and the data owner. As explained above, Meeco or [BitsAbout.Me](#) argue that the location of their servers is chosen for the high level of legal privacy protection, and [Digi.me](#) lets users select the cloud storage of their choice. In some cases, users can choose to have their personal cloud instance hosted directly by the cloud provider (e.g., CozyCloud) or by any other trusted third party of their choice (e.g., OVH, Dropbox, etc.). CozyCloud follows a similar principle, following the motto that ‘users will stay because they can leave’. In addition, such personal cloud providers claim to adopt an economic model creating a virtuous circle for privacy and promise not to monetize the personal data provided by the users (or unless they explicitly demand it). The third argument to gain users’ confidence is linked to the *transparency or auditability* of the code of the applications and the personal cloud platform. For example, CozyCloud’s applications and data system code are open source. The main consequence is that expert users can review the code or audit it such that any black box effect is avoided.

Overall, these solutions focus on a very similar set of functionalities, which cover the collection of personal data, storage in an individual personal data store, and the integration of the data such that transversal information processing is made possible. However, while most initiatives claim to guarantee users’ privacy, these approaches mainly rely on legal and economic frameworks with an unclear impact on the technical means to enforce security and privacy guarantees. Moreover, these approaches implicitly rely on very strong hypotheses in terms of security: (i) the personal cloud provider, employees and administrators are assumed to be *fully-honest*, and (ii) the overall personal cloud code as well as the whole set of personal applications and services running on top of it are considered *trusted*. Common security and privacy threats remain thus insufficiently addressed. Typically, data leakage resulting from attacks conducted against the personal cloud provider or the applications (which could be granted access to large subsets of raw personal data), or resulting from human errors, negligence or corruption of personal cloud employees and application developers, cannot be avoided in practice. This is critical because such solutions rely on a centralized cloud infrastructure settings which exacerbate the risk of exposing a large number of personal cloud owners, and hence may be subject to many sophisticated attacks.

2.2. Zero-knowledge-based personal clouds

Zero-knowledge personal clouds such as SpiderOak or Sync and to a certain extent MyDex or [Digi.me](#) mentioned above, propose architectural variations of the online Personal Cloud solutions in particular to mitigate some of the internal privacy issues raised by the strong assumption that the service provider is trusted. These solutions focus in particular on **secure storage** and **backup**. These functionalities, as well as elements of the corresponding privacy threats, are sketched below.

Secure storage. In most of the personal cloud solutions offering zero-knowledge storage, data is stored encrypted in the cloud and the user inherits the responsibility to store and manage the

encryption keys elsewhere (and never transmit it to the outside nor to the personal cloud provider). In SpiderOak, the personal cloud provider knows the number of encrypted data blocks produced by a given user, but not their content nor the associated metadata information (e.g., folder or file names). The encryption key of a user is derived from the user's password (i.e., password-based encryption). Note that file deduplication (i.e., storing a file only once at server side if several users hold that file) is not feasible as it would result in increasing the knowledge of the server. Sync uses password-based encryption techniques as well, but focuses more on the synchronization issues between the different devices of a same user. MyDex also offers a zero-knowledge service following privacy-by-design principles, but the system is here separated in two parts: (i) a front-end service which is in charge of retrieving the private keys from the users at connection time, encrypting and decrypting the personal data of the client during the session, and expunging the keys at disconnection; and (ii) a back-end service which stores collections of encrypted files. Note that an implicit assumption here is that the two parts cannot collude, and that relying on a front-end service at server side (instead of a client-side implementation) to manage the cryptographic keys weakens the zero-knowledge claim.

Secure backup. A common asset advertised by many zero-knowledge personal cloud services is secure backup, as a means to recover personal data when faced with a ransomware attack, personal device failure or unexpected data deletion. Thus, most zero-knowledge solutions, like SpiderOak, propose point-in-time recovery, such that users can recover any previous version of their personal files at a given date in the past. Note, however, that the user must assume the responsibility of storing and managing the encryption keys, since managing them server side would conflict with the zero-knowledge nature of these solutions.

Trust model. The threats considered by these solutions include (i) an attacker who compromises the personal cloud provider (i.e., addresses the case of *data snooping* and *data leakage*), (ii) a personal cloud provider that would want to make non-advertised usages on the customers' data content (i.e., *secondary usages*, e.g., personal data monetization), and (iii) a *client device failure* or corruption (e.g., *ransomware attack*). Note however that the term 'zero-knowledge' does not refer here to its cryptographic definition counterpart, since the personal data access patterns are not supposed to be hidden from the personal cloud provider. This consideration recently led to adopt the term 'no-knowledge' data store. A recent analysis of the threat model considered by SpiderOak (which applies to the zero-knowledge personal cloud providers as discussed here) is presented in [7]. In a nutshell, the threat model assumes *Honest-but-curious* or *Malicious* personal cloud providers, and a *trusted client application and device* (at least, considered trusted before the time of failure or before a ransomware acts).

In conclusion, compared to the basic online personal cloud (see Section 2.1), the zero-knowledge personal cloud solutions offer a higher level of security since the personal cloud provider cannot access personal data in clear. However, the price to pay is a *minimalist functionality*, i.e., the difficulty to develop advanced services on top of zero-knowledge personal clouds, which reduces the uses of a zero-knowledge personal cloud to those of a robust personal data safe. Moreover, since data processing (beyond basic storage) cannot be delegated to the server, data-oriented treatments are embedded into the client applications, shifting the security and privacy issues to the client's side. On the other hand, the assumption of an honest client application (which has access to the decryption keys and to the raw data in clear) on which such cloud solutions rely, may be too strong to hold in practice (due to, e.g., the ubiquity of viruses), thus creating a vicious circle.

2.3. Home cloud software

Other personal cloud initiatives, called 'home cloud' hereafter, build upon the paradigm of local/edge computing. These initiatives consider that raw personal data should remain stored at the extremities of the network (e.g., within the user's equipment or close to the IoT device which produced it) as a means to circumvent the intrinsic security risks of data centralization (i.e., corruption of the server resulting in massive data leakage and illicit data usages). In this Section, we first discuss purely software-based solutions, and then move on to the case of hardware-based proposals in the next two Sections.

Some remarkable representatives of home cloud software solutions are OpenPDS [6] and DataBox [8,9]. Both focus mainly on new privacy models allowing users to reduce the amount of personal data exposed to remote parties (i.e., data services or other users) and audit data exchanges. The core of these proposals is based on a *trusted storage* hosted locally on the user's device or at the edge of the network, combined with *cross-computations* and *data sharing* such that users may consent revealing only query results to third parties instead of disclosing large amounts of sensitive raw data.

Trusted storage. OpenPDS is a personal cloud solution which allows a user to accumulate personal data about her (e.g., web or shopping preferences, location traces) on her device (e.g., smartphone) and which provides a privacy preserving framework to explore this data. In Databox, the raw data storage is based on several isolated local stores, each data store being associated with a given source of personal raw data (e.g., one store per IoT device or sensor equipping the user), all located at the edges of the network. In both cases, data storage is considered trusted because it is managed locally on a user device.

Cross-computations and data dissemination. For both functionalities, the goal is circumventing the problem of third-parties being granted access to large amounts of user's raw data. Data sharing in OpenPDS is based on a framework called *Safe Answer* [6], a query-answering system used to analyze (i.e., cross-compute) the personal data collected by the individual and minimize the information about the data exposed to third parties or applications. The idea is to answer precise questions rather than externalize the complete set of raw data on which the queries are processed. In the same vein, DataBox [8,9] proposes techniques inspired by the Human-Data Interaction (HDI) paradigm to enable individuals to understand what data is collected about them and how it is processed. The proposal is based on separating the raw data stores from other stores dedicated to materialize aggregated query results, which can be made accessible to remote third parties. The proposed model relies on the ability to log all data accesses and data flows, and provides audit capabilities to simplify the users' control and understanding of the effective dissemination of their personal data.

Trust model. The focus of both OpenPDS and DataBox is not on security and enforcement, but on the study of the aforementioned functionalities under the angle of new privacy models and edge computing. An implicit trust assumption is that the (local/edge located) data stores hosting the personal raw data are *trusted* as well as the data system used to implement these functionalities (i.e., the cross-computation engine used to produce the aggregated results to answer queries, as well as the sharing model and the audit framework). The software architecture of Databox advocates the use of Docker containers (MirageOS unikernel being mentioned as a long-term alternative) to isolate certain data computations on the raw data. However, formal security guarantees are not discussed and the proposals do not focus on solutions to ensure that the proposed privacy models cannot be bypassed.

Compared with zero-knowledge solutions, formal security guarantees (in particular on the backup service) are lost. But

interestingly, the impact in terms of ‘minimalist functionality’ is alleviated since advanced data services could potentially be provided as part of the personal cloud platform (which is obviously not compatible with the zero-knowledge guarantee). In addition, such solutions target user privacy protection against over-privileged third-parties and applications under the strong security assumption of having a fully-secured personal cloud software user side.

2.4. Home cloud plugs

Home cloud plug solutions such as Lima,⁴ Helixee,⁵ CloudLocker⁶ and MyCloud,⁷ distinguish themselves from the previous approaches by providing solutions helping the users to self-host their personal data at home on a dedicated hardware platform. These solutions take the form of hardware plugs that can store TBs of data, which are synchronized with all the devices of the owner and can be accessed online.

Trusted storage and backup. In Lima, the hardware plug is connected to the user’s internet home-box and to an external disk drive. Other solutions, like Helixee, directly integrate the disk drive into the plug. Personal data is stored encrypted locally and is made accessible by the home cloud plug, which holds the encryption keys, to a set of personal devices authorized by the user (e.g., her smartphone and laptop). Usually, a central server can be used as DNS (as in Lima) to establish a remote connection between the users’ devices and the hardware plug. The system can then be backed-up automatically using a second plug or a remote encrypted archive locked by the user password.

Trust model. The main privacy benefit of home cloud plugs comes from the absence of delegation to a central cloud server. In terms of security however, the implicit assumption is strong, as the home cloud hardware and software platform must be *trusted*. This hypothesis is supported by the fact that the hardware plug constitutes a rather closed (dedicated) platform since no application except the software managing the plug is supposed to run on it. This limits the attack surface compared with richer devices (such as a smartphone). However, no formal security guarantees are provided to the user concerning the self-hosted platform and the application services running on top. This can put the user’s raw data at risk considering that unsecured end-user devices are accessing it. For instance, the DynDNS attack of November 2016, which infected unsecured end-user devices (e.g., printers, IP cameras and residential gateways) with a malware, illustrates the vulnerability of such home-based solutions.

To conclude, the focus in terms of personal data uses is again mainly on trusted storage and backup, and to a certain extent basic data sharing to support data synchronization between all the devices of the user. However, collaborative uses involving personal data from multiple individuals are, to the best of our knowledge, outside of the scope of these approaches. In terms of privacy and security, the gain compared to home cloud software solutions is, to some extent, a better controlled client execution environment for the personal cloud software, which nevertheless does not provide strong security guarantees. These two complementary approaches pose a problem in terms of safely extending the data related functionalities. Indeed, implementing a new advanced data service would require either to extend the trusted code hosted on the hardware plug, which should be considered as a closed platform for security reasons, or to add it as an external app, which in this case would run on vulnerable client devices.

2.4.1. Tamper-resistant home cloud

To improve the security of home cloud plugs, research proposals like Personal Data Server (PDS) [5] and Trusted Cells [10] introduce secure (i.e., tamper-resistant) hardware at the network edges to manage the user’s personal data. These approaches propose to embed a minimal Trusted Computing Base (TCB) dedicated to data management in the secure element of smart phones, set-top boxes or portable USB tokens to form a global decentralized secured data platform.

Secure storage. The PDS approach builds upon the tamper resistance of secure chips (e.g., smart cards, secure tokens). A DBMS engine is embedded in a secure chip, and hence inherits its security properties. The database metadata are stored in the internal memory of the chip are thus become tamper resistant, and the database itself (the data, indexes, logs, etc.) is cryptographically protected and stored in an external Flash memory (e.g., a raw NAND Flash chip or a MicroSD card linked by a bus to the microcontroller like in PlugDB⁸).

Secure cross-computations. Simple query evaluation and an access control engine can be integrated into the PDS engine running in the secure chip [11,12]. The secure cross-computations are however limited to simple database queries.

Secure distributed computations. The possibilities of crossing data belonging to multiple individuals (e.g., performing statistical queries over personal data, computing queries on social graphs or organizing participatory data collection) while providing strong privacy guarantees have been explored in the context of a network of PDSs so that each user can keep control over her data. The personal data is stored locally in each user’s PDS and the execution takes place on a hybrid infrastructure called an asymmetric architecture: on the one hand the PDSs of the participants are secure (i.e., behave honestly) but have low computation power, on the other hand, they are supported by an untrusted cloud infrastructure (e.g., honest-but-curious) implementing an IaaS or PaaS with significant storage and computing power. Different algorithms and computing paradigms have been studied on this architecture, from SQL aggregates [13] to special aggregation in a mobile participatory sensing context [14]. In all cases, the challenge is to trade privacy for performances depending on the equilibrium between the secure computations executed by the secure PDSs and the ones delegated to the untrusted cloud infrastructure.

Trust model. In this line of work, each PDS is assumed to be trusted. This trust assumption comes from several factors: (i) the PDS software inherits the tamper resistance of the hardware and can be certified according to the Common Criteria, making hardware and software attacks highly difficult, and (ii) the embedded database can be auto-administered due to its simplicity (in contrast with multi-user server counterparts) which precludes DBA attacks. However, a PDS cannot provide all the required database functionalities without resorting to an external infrastructure (e.g., distributed queries involving several PDSs as described above require external supporting servers). While the PDSs can be assumed to be fully trusted, the supporting communication and computation infrastructure is considered as the main adversary. It is considered as a *malicious adversary* having *weakly malicious intents* [15]. This means that it may deviate from the protocols it implements in order to infer personal information, but only tries to cheat when it cannot be detected by any PDS user. This assumption is commonly made for cloud services, as publicly advertising data leaks would cause important (financial) damages to the underlying service provider.

In conclusion, a high security level can be achieved since tamper resistant hardware is used. However, only rather simple queries can be addressed, and the underlying query engine being part

⁴ <https://meetlima.com/tech.php?lang=en>.

⁵ <http://www.helixee.me/>.

⁶ <https://www.cloudlocker.eu/en/index.html>.

⁷ <https://mycloud.com/>.

⁸ <https://project.inria.fr/plugdb/en/>.

of the TCB (running inside the secure microcontroller) must be proven secure. In addition, this approach leads to a non-extensible data system for two main reasons. First, supporting any new data and query model would require redesigning the underlying data storage, indexing and query processing techniques to comply with the strong constraints of tamper resistant hardware. Second, any advanced and potentially extensible database processing (e.g., large pieces of code implementing user defined database functions, stored procedures, database workflows or involving existing libraries supporting data intensive processes) is proscribed as it cannot be integrated as part of the TCB. These two main drawbacks drastically limit the practicality and the genericity of the PDS approach. The security is hence achieved at the price of extensibility. Hence, such solutions mainly target ad-hoc applications managing highly sensitive data, e.g., personal Electronic Health Records.

2.5. Synthesis of the existing approaches

The solutions presented above address different functionalities of the personal cloud and consider different trust models, which are both summarized in Tables 1 and 2.

In terms of functionalities (see Table 1), two important conclusions can be drawn. First, **the whole personal cloud data life-cycle must be covered**. We observe that the different solutions tackle different stages of the life cycle of the personal data in a personal cloud. In particular, all solutions discussed above address *data collection, storage, backup, cross-computations and data dissemination*. An *extensive* personal cloud solution should hence include all these functionalities to cover the whole personal data life cycle.

Second, **distributed computations should be part of the covered functionalities**. We also note that the *distributed computations* step is currently poorly covered. Is this because this functionality is less useful or because it is too difficult to be covered in practice in the personal cloud context? Regarding the utility of this functionality, we argue the opposite. Distributed computations over the personal data of (very) large sets of individuals unquestionably pave the way for Big –personal – Data computations with many applications in a personal cloud context, like computing recommendations, launching participative studies, learning information using the data of users belonging to a community (e.g., training a neural network in a patient community) or making collective decisions. However, this also requires privacy preserving implementations. A primary condition under which large sets of individuals would contribute with their own private data to collective uses is the guarantee that neither the other participants nor the infrastructure can access individual data. This probably explains why the only line of work addressing this step is focusing on security and proposes solutions based on tamper resistant hardware.

Trust is another essential concern of the personal cloud (see Table 2). Two conclusions can be drawn for the state-of-the-art analysis.

First, **all the privacy threats considered in the state-of-the-art solutions must be circumvented** to protect user's privacy and security in a meaningful way. Indeed, several threats are addressed by the different proposals, such as data snooping and secondary data uses performed by cloud providers (e.g., data monetization), corrupted applications or client devices (e.g., ransomware), or personal device failure. They all makes sense from a personal user point of view, since her whole digital life is managed and controlled using the platform.

However, a second (negative) conclusion is that **unifying these different solutions does not lead to a secure personal cloud architecture**. This is the case because building the union of the proposals would undeniably face irreconcilable architectural choices. Indeed, we observe that the existing personal cloud solutions cover

a rather wide spectrum of architectural choices, but this leads to different – and sometimes contradictory – trust models and security measures. More precisely, each class of solutions addresses a specific subset of functionalities while considering a specific threat model. For example, how is it possible to combine zero-knowledge encrypted storage with online personal cloud data-oriented computation facilities without returning all an individual's data to the client side and putting them at risk?

In conclusion, we can derive from this state-of-the-art analysis the set of functionalities to be implemented in the underlying Personal Data Management Systems (PDMS for short) to cover the complete data life-cycle, and the list of privacy threats the PDMS must circumvent. However, existing solutions do not address the whole functionality/threats spectrum and cannot be combined. Our goal in the next section is to progress towards a clearer definition of what an *extensive* (covering all the functionalities) and *secure* (addressing all the threats) PDMS should be.

3. Definition of an Extensive and Secure Personal Data Management Systems (ES-PDMS)

Currently, the principles underlying most existing solutions seem to be directly inherited from those considered in the context of the corporate cloud and have not been rethought with personal use in mind. For example, many solutions examined in the previous section rely on data encryption but nothing is said about restoring the master key in case of damage, except resorting on trivial unsecure protocols or on so-called-trusted third parties. Similarly, how to securely collect personal data from web sites through a myriad of unsecure wrappers without leaking both the user credentials needed to connect to the remote site and the collected personal data? We argue that the way the PDMS functionalities are implemented and secured is determined by the intrinsic personal use of the PDMS, and must be deeply redesigned with this statement in mind.

In what follows, we first review in Section 3.1 each of the PDMS functionalities identified in the state of the art as needed to cover the whole data life-cycle and we analyze their intrinsic specificities. Then, in Section 3.2, we derive the fundamental security property attached to each functionality (and hence to each step of the data life-cycle). This analysis leads to the definition of an *Extensive* (i.e., providing the needed functionalities to cover the whole data life-cycle in a personal cloud) and *Secure* (i.e., achieving all the expected security goals) *Personal Data Management System* (ES-PDMS), which is provided in Section 3.3.

3.1. Specificities of data management in the PDMS context

As identified in Section 2.5, the PDMS functionalities are expected to cover the main stages of the personal data life-cycle and should thus integrate *data collection, storage and recovery, personal computations, distributed computations and data dissemination management*. For each functionality, we discuss their main specificities, and highlight to which extent they differ from their corporate data management system counterpart. Note that some operational aspects (e.g., how to interact with the PDMS, how to engage into a collective computation, etc.) are more platform dependent and will be addressed in Section 4 where physical instances of PDMS will be discussed.

Data collection. The data collection functionality concerns both primary copies of user data (e.g., quantified-self data, smart home data, photos, videos, documents generated by the user, etc.) and secondary copies (e.g., banking data, health, employment, insurance, etc.). While the primary copies can be directly fed to the PDMS from data sources under the user's control, secondary copies

Table 1
Main functionalities of the state-of-the-art personal cloud solutions.

		Representative Personal Cloud approaches				
		Online personal cloud	Zero-knowledge personal cloud	Home cloud software	Home cloud plug	Tamper resistant home cloud
Functionality	Storage	Regular DBMS technology	Zero-knowledge cloud storage	Data stored on a generic user-side device, separated stores for different data sources	Data stored on a dedicated user-side device	Data stored in a tamper resistant device at the user-side
	Backup	Regular DBMS technology	Encrypted archive, point-in-time recovery	Replication / offline storage	Replication / offline storage	Replication / offline storage
	Data collection	Web scrapping	Files pushed or synchronized by the user to the cloud provider	Personal data inserted by the user or automatically to the home cloud	Personal files inserted by the user to the home cloud	Personal data pushed by the user or third parties to the home cloud
	Cross-computations	Transversal DB queries	At application level	Question answering, local data aggregation	At application level	Simple transversal DB queries
	Distributed computations					Simple distributed SQL statistics at large scale
	Data dissemination	[synchronization]	At application level	Minimum privileges for third parties and applications	[synchronization]	Minimum privileges for third parties and applications, secure access control

Table 2
Trust considerations in the state-of-the-art of personal cloud solutions.

		Representative Personal Cloud approaches				
		Online personal cloud	Zero-knowledge personal cloud	Home cloud software	Home cloud Plug	Tamper resistant personal server
Trust	Considered threats	secondary usages (monetization) performed by the cloud provider	Data snooping or leakage, secondary usages performed by the cloud provider, client device failure and ransomware attacks	Massive data snooping or leakage, secondary usages, over-privileged third parties and applications	Massive data snooping or leakage, pecuniary usages, home plug device failure	Massive data snooping or leakage, secondary usages, over-privileged third parties and applications
	Trust model	Fully-honest personal cloud provider, trusted personal cloud code, trusted applications	Honest-but-curious to Malicious cloud provider, trusted applications, trusted client device (before time of failure or ransomware attack)	Trusted personal cloud code, trusted client device, untrusted applications	Trusted personal cloud code, trusted home plug, trusted client applications	Trusted personal cloud code, honest-but-curious central supporting infrastructure, untrusted applications
	Privacy and security measures	Security standards, virtuous legal and economic framework, transparency and auditability of the code (apps/PDMS)	Client-side encryption, 'no-knowledge' cloud store	SafeAnswers, logical separation of the personal data stores, audit	Closed platform (dedicated device), physical ownership	Secure hardware, physical ownership small TCB, secure distributed protocols

have to be scrapped from the online services holding them. Collecting data from external sources is a basic operation of any corporate data management system. This task is usually handled thanks to a well-known and predefined set of carefully audited, patched and supported wrappers, under the control of data and security administrators who guarantee the quality and integrity of the integrated data. In the PDMS context, the situation is totally different. The PDMS owner is confronted with a large variety of scrappers (e.g., Web Outside of Browsers⁹) capable of capturing various types of data from a myriad of online services, the code of which cannot be trusted due to code complexity, diversity of contributors and sometimes closed source. However, by construction, such scrappers have access to highly sensitive data, from the credentials required to connect to the online service to the scrapped data itself (e.g., bank records, pay slips, invoices, medical records). Moreover, the user environment (e.g., operating system,

network, other apps) in which the wrappers run is by far less trusted than an enterprise administered environment.

Storage and recovery. As any data management system, a PDMS has to securely store and ensure the durability of the data it manages. This means setting up encryption protocols to protect the data at rest against piracy and backup/recovery mechanisms to protect them against accidental loss. In a corporate DBMS, all these tasks are handled by DBA and DSA, having a recognized expertise in data management and security. Such expertise cannot be assumed for the PDMS owner. However, the risk of confidentiality and integrity attacks on private data has never been so high, as demonstrated by massive ransomware attacks affecting both individuals and organizations. Hence, the PDMS owner is facing the choice between endorsing the responsibility of data administration tasks that she cannot reasonably undertake or delegating these tasks to a (trusted) third party and thus abandoning the empowerment she just received from the PDMS paradigm. Zero-knowledge storage providers argue they have a solution to this problem, but this is by ignoring the issue of restoring the master

⁹ weboob.org.

key protecting the data in case of loss. They simply suggest deriving the master key from a password, but the password entropy is such that the cryptographic protection becomes highly questionable in this case. Cloud encryption is however not new and robust corporate solutions exist, like resorting on expensive Hardware Security Modules (HSM) to secure the master-key, a solution that individuals can unfortunately not afford.

Another distinguishing issue between the corporate and personal contexts is the liability regarding the hosted data. A company hosting personal data must guarantee the confidentiality of this data and can be subject to sanctions by a court in the event of a data leak. Thus, companies usually take appropriate measures to protect themselves. The legal responsibility of a PDMS owner remains unclear today, yet a PDMS can host personal data from many other people (e.g., contact details of doctors and relatives or external personal data gathered during a collective computation). Thus, the PDMS must protect this data on the owner's behalf and might even have to prevent her from accessing some of this data. Hence, the PDMS owner must not be granted access to the full content of her PDMS. Consequently, the PDMS owner must not even be granted a direct access to the master key required to decrypt the backup archive in case of a crash. This is a typical example of new issue raised in the PDMS context.

Personal computations. Personal computations in a PDMS usually refer to apps crossing various data of a single individual: the PDMS owner. We can distinguish between two types of such apps reflecting two specific uses of a PDMS: (i) apps used directly by the PDMS owner (e.g., for quantified-self, health and wellbeing, statistics and analyses related to smart home data or user's mobility, etc.); and (ii) apps representing external services to which the owner willingly subscribes (e.g., billing apps allowing a car insurance company to compute the premium based on the owner's car trajectory data — as in pay-as-you-drive, or an electric company to compute the bill based on the user's electric smart meter traces). Therefore, an important specificity in the PDMS context is that apps “move” towards the data as opposed to personal data migrating towards remote services as it happens with most existing cloud services.

This has two main implications. First, the apps manipulate sensitive raw data, but neither the apps nor the environment in which they run can be trusted in general, leading to similar security problems as the ones discussed for data collection. *By-default auditing mechanisms* are thus required to detect malicious apps deviating from their manifest. These mechanisms must be easily understandable by non-expert users, disqualifying advanced audit tools based on complex models and formal languages usually employed by audit experts and security administrators in an enterprise context. Second, some external service apps need strong guarantees regarding the results produced by a PDMS (e.g., the billing apps discussed above). That is, an *attestation* process is required to ensure that the result was indeed produced by a certain computation code using all the required input data, and that the PDMS owner cannot tamper with the computation process nor the inputs.

Collective computations. Collective computations relate to various types of big personal data processes computed over a large set of PDMSs (e.g., contributing to participative studies, training a neural network, building an anonymous dataset). In addition to the security issues already discussed regarding the intrinsic untrusted nature of apps and computing environment, collective computations introduce a new difficulty. Gathering all the participants' data in a single place to perform the computation introduces a single point of vulnerability and maximizes the incentive to attacks. Conversely, decentralizing the processing implies to temporarily transfer personal data among participants, transforming each into a potential attacker. In this latter case, two guarantees must be

provided: (i) data confidentiality, i.e., any PDMS owner cannot access the data in transit of other participants, and (ii) distributed computation integrity, i.e., any participant PDMS can attest that any result it supplies corresponds to the assigned computation. Classical distributed computation techniques used in enterprise systems cannot apply here due to the unusual scale of the distribution (i.e., the computation may target a fraction of the population of a country). Generic secure multiparty computation protocols based on cryptographic techniques (MPC) are disqualified for the same reason (performance does not scale with the number of participants). Conversely, the participants cannot trust each other since participants are unknown a priori (and probably wish remaining anonymous).

Data dissemination management. The purpose of a PDMS is to enable the owner to make the necessary decisions regarding the dissemination of his or her personal information. In particular, according to the aforementioned functionalities, the user should be able to give the appropriate permissions on the data or documents to share with acquaintances and distant third parties, to decide which personal computations she will authorize and which collective computations she accepts to contribute to. In a corporate context, such decisions would be managed and enforced by central authorities and would once again rely on IT experts (DBA and DSA) who define appropriate roles, set access control policies (e.g., following RBAC, MAC, ABAC or TBAC models), and provide system security and audit to ensure that everything goes as planned. On the contrary, the PDMS context puts such decisions and their enforcement into non-expert users' hands, with the risk of generating more security holes than solving them. For example, the aforementioned access control models are not well adapted to a non-expert administrator having to manage a highly dynamic set of interactions with a myriad of other users and third parties. Specific tools have been suggested to let individuals manually define their sharing preferences (e.g., thanks to PGP, Web of Trust models or FOAF dissemination rules) but they provide little consistency guarantees about the final outcome. Conversely, relying on a trusted third party to manage personal data dissemination would be contrary to the very notions of user control and empowerment. Hence, whatever the way the PDMS owner defines her sharing preferences, the PDMS must provide ad-hoc tools to help her easily understand the net effects of her decisions related to data dissemination and sanitize the policy accordingly when required. In addition, the PDMS owner is not a super-user having all privileges over the full content of her own PDMS content. As already stated, a PDMS may host other users' personal data and the PDMS must protect this data against unintended actions of the owner of the PDMS herself.

3.2. Security properties of a PDMS

As a conclusion of the preceding analysis, the PDMS context sketches an open and rich ecosystem of new untrusted data processing apps in interaction with an unsecure execution environment and a layman PDMS owner. This significantly contrasts with corporate data management systems where the applications and computing environment are significantly more static and carefully controlled by data and security administrators. Additionally, typical distributed computation infrastructures (cluster/cloud) strongly differ from a fully decentralized infrastructure of PDMSs (e.g., in terms of scale, ownership, legal agreements, deployed hardware and software, etc.). As a consequence, the PDMS must integrate by default novel security measures to overcome the inherent weaknesses of the PDMS owner and tackle the specific threats to this open and untrusted ecosystem. We detail below the security properties expected from a PDMS, and linked to each functionality described in the previous section (one security property is

thus associated with each step of the data life-cycle). We make no assumption about the technical means to enforce these properties, delaying this discussion to the next sections.

Piped data collection. Under the hypothesis of untrusted collection code and untrusted user computing environment, a PDMS is said to enforce *piped data collection* iff:

1. the only PDMS data accessible by the collection code are the credentials allowing access to the related data providers;
2. the credentials and the collected data related to a given data provider cannot be leaked outside the PDMS and the data provider.

This property guarantees that the only channel to the outside world provided to the data collector is a specified data provider and that the code is suitably isolated so as not to be able to leak data to a potentially corrupted user environment.

Mutual data at rest protection. Under the hypothesis of a layman PDMS owner, a PDMS is said to enforce *Mutual data at rest protection* iff:

1. the PDMS unconditionally protects the hosted raw data and the backup archive against any form of confidentiality and integrity attacks or accidental damages conducted by external adversaries or by the PDMS owner herself;
2. The secret protecting the backup archive is recoverable;
3. This secret is not accessible to the owner nor any other party except another PDMS belonging to the owner and providing all the expected PDMS functionalities and security properties (typically, an *Extensive and Secure PDMS* as defined in Section 3.3).

Since a PDMS stores raw data from the owner and also personal data from other users, data protection must also operate against the PDMS owner, thus the term *mutual*. To be effective, the PDMS must enforce this property automatically, without any owner intervention which could open the door to administrator attacks. Moreover, it requires that the archived data can only be interpreted by a PDMS, whatever the way the secret protecting it is produced, made resilient and recovered.

Bilaterally trusted personal computation. Under the hypothesis of untrusted external code and untrusted owner computing environment, a PDMS is said to enforce *bilaterally trusted personal computation* iff:

1. a personal computation may access only the owner's raw data specifically required for the computation;
2. only the final result of the computation – not the raw data – may ever be exposed to a third party;
3. the execution of the computation produces trustworthy audit trails accessible to the owner;
4. the PDMS can provide a proof that the result of the computation was produced by the expected code.

This property provides *bilateral guarantees* to the PDMS owner and the third party willing to execute code on the owner's data. It guarantees to the former that the minimal collection principle enacted in laws protecting personal data (e.g., GDPR) is fulfilled, that the computation cannot leak unexpected data and finally that she will have the ability – not the obligation – to audit the compliance to this property. Conversely, it guarantees to the latter (e.g., an energy provider willing to compute the owner's bill) that the code remotely sent to the PDMS has been accurately computed. If this computation combines several tasks, the proof produced by the PDMS must guarantee that the orchestration of these tasks cannot be tampered with without the caller being able to detect it. However, the PDMS cannot attest by itself that the data targeted by this code is genuine. Such attestation remains under

the responsibility of the computation code, assuming that the data has been properly signed by their producer.

Mutually trusted collective computation. Under the hypothesis of untrusted external code and untrusted user computing environment, a PDMS is said to enforce *mutually trusted collective computation* iff:

1. a collective computation may access only the participants' raw data specifically required for the computation;
2. only the result of the computation – not the raw data – may ever be exposed to a third party or to any participant;
3. the execution of the computation on a participant generates trustworthy audit trails accessible to that participant;
4. a proof can be provided that the result of the computation was produced by the expected code over the expected set of participants.

This property targets the same objective as its bilaterally trusted personal computation counterpart. It must integrate the fact that participants can contribute to the collection phase and/or the processing phase of the computation with no assumption on the cardinality of these two sets of participants and on their intersection.

Controlled data dissemination. Under the hypothesis of a lay PDMS owner and of an untrusted execution environment, a PDMS is said to enforce *controlled data dissemination* iff:

1. the integrity and confidentiality of interactions between the PDMS and its owner are guaranteed, when defining the dissemination policy and auditing its effects, and when decisions are made regarding the regulation of data dissemination
2. the decisions are enforced by the PDMS and cannot be circumvented;

This property guarantees to the PDMS owner that all decisions (e.g., entrust a secret share to a remote user for recovery purposes or allow a collective computation) are faithfully captured (point 1), which calls for integrity guarantees to ensure that the decisions cannot be corrupted when captured, and confidentiality to ensure that the decisions themselves and the personal data on which they may rely cannot be leaked. Thus, the effects of these decisions in terms of data dissemination are enforced by the PDMS and cannot be circumvented (point 2) neither by any third party, nor by any potentially untrusted parts of the execution environment, and not even by the PDMS owner who may have restricted privileges (e.g., over data generated by other users, or over cryptographic secrets or metadata generated for administrative purposes such as ensuring mutual trust as explained above). Finally, audit tools (point 1) are provided in order to help layman PDMS owners understanding all the effects of the taken decisions in terms of data dissemination and allow her to act to update decisions or rectify their effects if needed.

3.3. Extensive and secure personal data management systems

The definition of an *Extensive and Secure Personal Data Management Systems* is directly derived from the previous sections and is expressed as follows:

Extensive and secure PDMS. An *Extensive and Secure Personal Data Management Systems* provides the expected set of functionalities to cover the complete data life cycle in a personal cloud, namely *data collection, storage and recovery, personal cross-computations, collective computations and data dissemination management*, and is compliant with their respective security properties counterparts, namely *piped data collection, mutual data at rest protection, bilaterally trusted personal computation, mutually trusted collective computation and controlled data dissemination*.

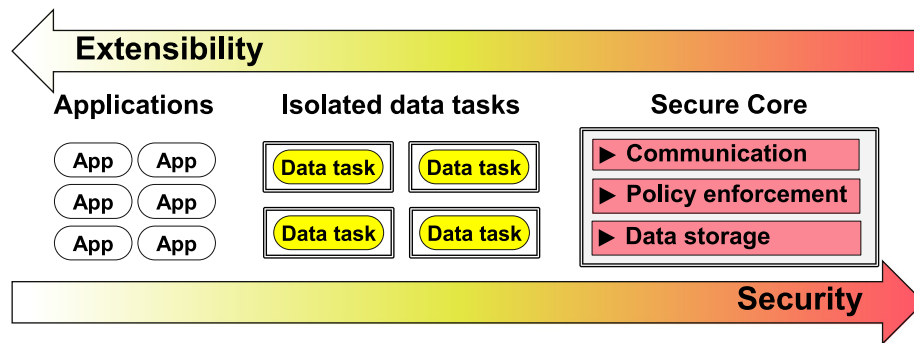


Fig. 1. Global architecture.

4. Extensive and secure PDMS architecture

Designing an extensive and secure PDMS architecture providing the functionalities and the five security properties defined above is highly challenging, given the fundamental tension between the security expectations of a layman PDMS owner and the need for supporting an open ecosystem of applications running on an untrusted environment. In this section, we introduce the fundamentals of a PDMS reference architecture tackling this tension and detail its building blocks. We then show that physical instances of this reference architecture can be already envisioned today and discuss how existing and forthcoming software and hardware mechanisms may impact the satisfaction of our security properties.

4.1. Logical architecture

Ideally, the support of potentially complex manipulations of personal data while preventing unexpected data leaks could be achieved by securely collecting, storing and manipulating the data in a secure subsystem, managed under the control of the holder, while never letting the (untrusted) applications or third parties directly access the raw data. Such a clean separation can only be based on the assumption that there exist a few generic functions that can be used to manipulate personal data without violating privacy. Such an assumption is obviously a fantasy, because the most interesting manipulations of personal data are application-specific and consequently, privacy violations are also application-specific.

To solve this problem, we propose a three-layer logical architecture where a minimal *Secure Core* (Core) implementing basic operations on personal data is extended with *Isolated Data Tasks* (Data tasks) themselves accessed by *Applications* (Apps) on which no security assumption is made (see Fig. 1). The objective is to control the flow of raw personal data from the Core to the outside, such that only expected results are declassified to untrusted applications or third parties. The general description of the architectural layers follows:

- **Core.** The Core is a secure subsystem that is a Trusted Computing Base (TCB) ideally minimal, inextensible, proven correct through formal methods and isolated from the rest of the system. The Core must provide all basic operations required to enforce the confidentiality, integrity and resiliency of the personal data hosted by the PDMS. It must be the unique entry point to manipulate this data. The Core must thus implement a *data storage* module. A *policy enforcement* module must be integrated in the Core to regulate the data access performed by the other layers of the architecture. A *communication manager* is also needed to securely communicate with other users, applications and third parties. In what follows, we give an empirical view of the Core minimality, by

identifying which combination of functionalities is (strictly) mandatory in each of the three parts of the Core to guarantee the security properties introduced in Section 3.2.

- **Data Tasks.** Data tasks are introduced as a means to deal with application-specific personal data management. The idea is to control complex data-oriented tasks by (1) splitting their execution into data tasks evaluated in a sufficiently isolated environment to maintain control on the data accessed by the Core and delivered to the Apps in order to avoid any side effect in terms of data leaks, and (2) scheduling and verifying the execution of data tasks by the Core such that security and privacy can be globally enforced.
- **Apps.** Any developer should be able to develop an application to ensure a wide and diverse application panel. However, the complexity of these applications (large code base, extensible and not proven) and their execution environment (web browser, smartphone, etc.) make them vulnerable. Therefore, no security assumption is made on applications, which manipulate only authorized data resulting from data tasks but have no privileges on the raw data.

4.2. Building blocks





This section details how each security property introduced in Section 3.2, namely *pipelined data collection*, *mutual data at rest protection*, *bilaterally trusted personal computation*, *mutually trusted collective computation* and *controlled data dissemination*, can be practically satisfied. For each property, we identify the required elementary building blocks and explain how they should be combined to reach the expected goal without introducing security breaches.

Each building block in turn relies on a set of *common security primitives* provided by the Operating System (OS) and/or the hardware platform hosting the PDMS. Hence, these primitives are the foundations of our empirical minimal definition of the Core. Since they are commonly used by various building blocks, we present them first. As their implementations differ across platforms, we concentrate below of the security primitives they provide.

Common security primitives:

- **Isolation.** A component of the architecture is said to be isolated if (i) the internal execution state of the component cannot be accessed nor influenced from the outside of the component except with the collaboration of the system administrator (e.g., the PDMS owner) and (ii) the component may not observe nor influence the behavior of any external system except through its own inputs/outputs behavior. See for example [16] for a survey of ways to implement code isolation in a partly untrusted context.

- **Attestation.** A component is said to be attestable if a trustworthy certificate can be produced to demonstrate that the component output was indeed produced by the specific code of this component. This common security primitive is usually considered for a whole complex system (e.g., [17]). It was formalized in the case of a single task running on a complex system in [18].
- **Confidentiality.** A component is said to ensure data confidentiality if neither the internal execution state nor the input and output data of the component can be leaked to any system other than the party initiating the component (in our case the PDMS Core) even with the collaboration of the system administrator. This property is described in [19] and formalized together with isolation and attestation as ‘secure outsourced computations’ in [18].
- **Peripherals isolation.** A component is said to satisfy peripherals isolation if the data exchanged between that component and the peripherals cannot be leaked outside of the component except with the intervention of the PDMS owner. For illustration purpose, [20] and [21] show how text messages can be securely displayed even in the presence of an untrusted OS using ARM TrustZone.

For the sake of clarity, each the security primitive defined here will be represented by a simple pictogram in the next figures showing the building blocks required to implement each of the security property introduced in Section 3.2. The pictograms used are  for code isolation,  for attestation,  for confidentiality and  for peripherals isolation.

4.2.1. Piped data collection

The piped data collection property requires the ability to execute arbitrary data collection code (e.g. a scrapper) in a secure manner. The objective of this property is actually twofold.

First, it should guarantee that the collection code will not access any data stored in the PDMS other than the credentials required to connect to the related service provider (e.g., the web site to be scrapped). This specific privilege must be part of the manifest declared at the time this collection code is registered in the PDMS. The resulting authorization itself will be enforced at execution time thanks to the controlled data dissemination property (see Section 4.2.5 for details).

Second, it should guarantee that the collected data and the credentials related to a given data provider cannot leak to any third party (including another data provider targeted by the same collection code). According to the reference architecture sketched in Fig. 1, this guarantee can be provided by considering the collection code as an isolated data task and granting a write access to this data task only to the destination PDMS. This requires data *authentication* to be implemented in the Core. In other words, this means restricting the write capacity of the data task to the insertion of the collected data into the Core. The enforcement of this restriction at execution time relies itself on the code isolation property.

While executing the collection code as an isolated data task inside the PDMS ensures that the effects on the Core are controlled, further measures are required in order to ensure the absence of leakage of both credentials and collected data. Indeed, the collection code needs to communicate with the outside world in order to reach the data provider. To preclude the collection code to leak data to any other party than the related data provider, the Core must be able to establish a (TLS for example) secure channel between that specific data provider and the data task. Regarding the minimality objective, the complete network stack (e.g., TCP/IP, DNS) does not have to be in the Core, only the critical security operations of the creation of the secure channel, named *TLS trusted* in Fig. 2, need to be.

Remark that each data collector task should be dedicated to a single remote site (e.g., my bank), to avoid a malicious data task from leaking credentials or personal data (e.g., the bank credentials/data to another site) through an authorized communication channel. Note that in addition a malicious data collector task may use the owner’s credentials on the remote site to perform unexpected actions (e.g., the data collector retrieving the bank related data could trigger a money transfer using the bank credentials). However, such issues are mostly related to the definition of a weak security policy at the data provider side, rather than a problem to be addressed at the PDMS architectural level. We thus assume here that the credentials delivered by the data provider for data collection purposes will grant only read access to the reduced dataset of interest (e.g., bank account history).

Summing up the architecture presented in Fig. 2, when performing data collection for a specific source, the Core launches an isolated data task executing the collection code for said source, provides it with the credentials and a secure channel to the source, which requires the implementation of data task authentication and secure channel set up (TLS-trusted) in the Core. Once collection is finished the collector returns the data to the Core which stores it appropriately. This ensures the absence of leakage (through isolation and secure channel), and proper behavior of the data collector in terms of input and output data (through access control). Another statement is that the safety properties are not independent from each other. Piped data collection indeed relies on controlled data dissemination and mutual data at rest protection to make sense when considered in a complete scenario.

4.2.2. Mutual data at rest protection

This property should be ensured by the Core which is the only entity authorized to have a full access to the PDMS data (recall that, even the owner does not have such an access, since the PDMS data may include, for instance, data from other users). *Secure storage* is thus ensured by the Core, potentially using classical cryptographic techniques (encryption, hashing) if the storage medium is not part of the Core, to protect the data against confidentiality and integrity attacks. In the following, we focus on the backup and recovery issues which are less classical in the PDMS context. We note that the secret protecting the backup archive can obviously not be stored on the PDMS (otherwise, it would be lost in case of failure). As already mentioned, relying on a password is not adequate due to the generally reduced entropy and to the difficulty and the risks associated with the memorization of complex passwords, with no way to reinitialize it. In addition, the owner should not be capable of restoring this secret alone since he does not have full access to the PDMS data, further disqualifying password-based solutions. A reasonable solution to this problem is to split the secret in a number of secret shares using a secret sharing scheme (for instance [22]). We provide below a simple example of how such a task could be performed, using standard cryptographic techniques and the *common security primitives* introduced at the beginning of Section 4.2.

We split the backup and recovery process in three steps. First the setup, which consists in generating and distributing all cryptographic material necessary for performing the encrypted backup and recovery. Then comes the backup phase. Finally, assuming one’s PDMS has been lost/compromised, comes the recovery phase.

Setup phase:

1. The PDMS generates a master key, inaccessible to the PDMS’s owner, which will be used for performing encrypted backup
2. The PDMS owner chooses a number n of trustees (among her friends/family), who will hold shares of her master secret key. She also chooses a security threshold s . This threshold is the number of trustees who need to cooperate to recover the owner’s PDMS content.

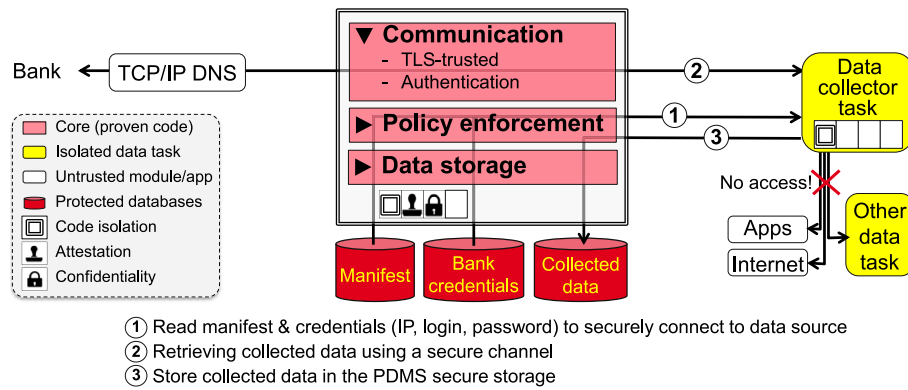


Fig. 2. Data collection.

3. The PDMS executes an s out of n secret sharing scheme (e.g., Shamir's secret sharing). The shares are subsequently distributed through a secure channel (using the TLS-trusted module) to the PDMSs of the trustees. We need to ensure that secure channels are indeed established with the trustees' PDMS and no other parties (i.e. to avoid other people from getting the shares). This can be done using a certificate that proves that a PDMS is genuine or using the Core hardware attestation mechanism of the PDMS if available. Note that it is essential that the trustees themselves cannot access the share directly, even though it is stored in their PDMSs. Indeed, if they could access these shares, they could, in collaboration with the PDMS owner, decrypt all encrypted backups outside a PDMS, and thus access data that is meant to not be accessible to the PDMS owner herself. Therefore, the permissions for these shares need to be set to only be accessible to the recovery functionality.

Backup phase: this phase is not specific to the PDMS context. The encrypted backup of the PDMS is performed on an untrusted third-party server by the backup module of the PDMS. One should be careful to choose a backup scheme that preserves integrity as well as secrecy (e.g., SpiderOak backup service [7], see Section 2.2).

Recovery phase: This phase is triggered when the user has lost access to its PDMS and needs to obtain a new copy of her data. It proceeds as follows (note that the practical means to realize this protocol may be adapted to minimize the PDMS's owner burden).

1. The user acquires a new empty PDMS.
2. The user contacts s trustees, and informs them that they need to perform the recovery procedure, with the identification data of the new PDMS, using an out-of-band communication channel (e.g., phone call, chat or email message).
3. The PDMS recovery modules of the s trustees communicate their shares of the master secret to the recovery module of the new PDMS, using the provided identification data of the new PDMS, on a secure channel (using the TLS trusted module again). It is essential to ensure that these shares are indeed communicated to a genuine PDMS, belonging to the user. As in step 3 of the setup phase, to achieve this guarantee, either a certificate or remote attestation is used by the trustee's PDMS in order to ensure that they are indeed communicating with the recovery module of a specific PDMS.
4. The recovery module of the new PDMS reconstructs the master secret using the share provided by the trustees' PDMSs.
5. The new PDMS retrieves the encrypted backup and recovers the data using the master secret.

As illustrated in Fig. 3, the main impact on the architecture is the need for *backup and recovery* modules in the Core. As the recovery can only be performed inside a legitimate PDMS, the user is never given direct access to restricted data. Additionally, neither the user nor the trustees may gain access to the secret shares, which ensures that the encrypted backup may never be decrypted outside the recovery process. Finally, the s out of n secret sharing ensures that the owner's secret is recoverable, even if the PDMS of a number of trustees fail. One can choose the threshold according to her confidence in the hardware and the reliability of her trustees and may even include one or several mandatory trustees in the secret recovery process.

4.2.3. Bilaterally trusted personal computations

First, as required in the security property (see Section 3.2) the personal computation functionality needs to be able to execute (arbitrary) code ensuring that only raw data required for the computation is made available to the code. The minimality requirement on the Core prohibits the execution of such complex tasks inside the Core. As a consequence, extensibility in terms of code execution is achieved by executing personal computations as data tasks. In order to restrict access to raw data, we require that any personal computation task comes with a manifest specifying precisely the data needed for said computation. This manifest should be approved by the user (see the controlled data dissemination property in Section 4.2.5). Subsequently, when executing a personal computation task, the Core's *reference monitor* provides only the data required by the manifest, ensuring by construction that the manifest is respected.

Second, only the result of the computation should be made available to third parties. As the user's system may be corrupted, this implies that the personal computation task should be isolated from the environment. This prevents the user's system from accessing the internal state of the data task, hence preventing leaks of raw data. Additionally, the data task should be authenticated by the Core and only be able to store the result in the Core. Any subsequent disclosure of the result to the outside world should be done by the Core and subject to control from the *reference monitor*.

Third, the computation should provide an audit trail accessible by the owner. This is again achieved through the *reference monitor*, which should update the audit trail accordingly through the *audit* component of the Core. In order to guarantee the minimality of the *audit* component of the Core, only simple actions should be logged, such as a handle on the data used in the computation and a handle on the result together with a handle pointing to the manifest of the data task which has computed the result. This avoids providing the complex relationship between the data and the result directly in the log while retaining the ability to reconstruct this relationship from data safely held in the PDMS.

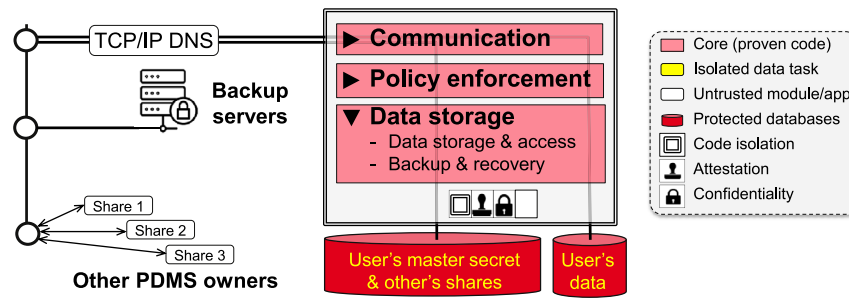


Fig. 3. Mutual data at rest protection.

Finally, it should be possible to provide a proof that the result is indeed produced by the expected code. This is achieved through *attestation* of the data task which exactly provides this guarantee. Note that, while *attestation* provides guarantees that the result was indeed produced by a specific computation task, this does not in any way provide guarantees on the raw data that was used in order to compute said result. If one wants to obtain guarantees on the data used, the checks should be included in the personal computation code. For example, if an energy provider wants to compute the monthly energy consumption from certified data from the energy meter using a personal computation task, this computation task should include checking the certificates for the data; otherwise the user might execute the data task on counterfeit data and the attestation would still certify that the result transmitted was indeed produced by the right computation task.

In order to perform computations that are not represented by one atomic task but rather by a succession of tasks, the task can leverage attestation in order to provide guarantees on the end result in a manner similar to checking certificates for data. Indeed, if a computation task T is supposed to be executed on some data resulting from the execution of a previous computation R , T 's code can verify that its input data is attested as the result of task R . Using this mechanism iteratively, it is possible to guarantee the integrity of a result coming from an arbitrary combination of computation tasks.

This functionality is presented in Fig. 4. To sum up, this implementation of the functionality ensures that only the required data is accessed (through *reference monitor*), that this data may never be leaked to the outside world (through isolation), that the operation flow is auditable by the user (through the *audit* component), and finally that a proof can be provided that a result corresponds to a specific computation (through *attestation*). This leads to the introduction of a *reference monitor*, an *attestation* module and an *audit* module in the Core.

4.2.4. Mutually trusted collective computations

This property targets the same objective as its bilaterally trusted personal computation counterpart. It must integrate the fact that the participants can contribute to the collection phase and/or the processing phase of the computation with no assumption on the cardinality of these two sets of participants and on their intersection.

As for the personal computations, our architecture implements collective computations as data tasks (or more precisely as a set of data tasks communicating through the network) in order to achieve system extensibility. The *reference monitor* ensures that a collective computation data task can only access the data required for the computation (point 1 of the *Mutually trusted collective computation* property in Section 3.2). Note that this data may include intermediate results transferred by other PDMSs participating in the collective computation. As in the personal computation case the data access requirements are specified by the manifest of the collective computation, which has to be accepted by the user. Also,

to enable the PDMS owner to audit her computations (see Section 4.2.5), the access audit module records information related to the execution of collective data tasks (point 3 of the *Mutually trusted collective computation* property in Section 3.2).

Another requirement of collective computation (point 2) is that only the result of the computation can be disclosed, and not the raw data. To this end, a first measure is to execute the collective computation data tasks as isolated data tasks. This protects a participant's raw data against an untrusted system environment. However, a collective computation requires the transmission of intermediate results between data tasks belonging to various participant's PDMSs. In order to ensure the confidentiality of these intermediate results, the *reference monitor* takes the following measures. First, it only gives access to the intermediate results to the other PDMSs executing the collective computation which are allowed to get these results as specified in the respective computation manifest. Second, at local level, it only gives access to the intermediate results to the specified data task and forbids any access by the PDMS' owner. Hence, to guarantee the enforcement of access control against malicious PDMS owners, the collective data tasks have to implement the confidentiality security property. Also, transferring intermediate results to other participating PDMSs is done using the *TLS-trusted* module to ensure that data is transmitted on a secure channel.

The final requirement for collective computation (point 4) is to ensure that a proof can be provided that the result of the computation was produced by the expected code running on the expected set of participants. This requires a global manifest (i.e., a formal description of the computation including the participants and their delegated tasks) allowing to check that messages originating from other participants were produced as expected. The global manifest is registered in the Core of each participant involved in the collective computation. We then need to propagate trust between nodes during the execution of the global computation protocol. Therefore, each data task output involved in the distributed protocol should be certified through attestation, such that each Core can check the local integrity of its local data task, and propagate it to the other participating Cores in order to incrementally establish the global integrity of the distributed protocol, with acceptable performance. Note indeed that instead of being verified by the third party exploiting the result as it is the case for local computations (see Section 4.2.3), the attestation must be verified here by the participants involved in the collective computation.

In conclusion, mutually trusted collective computations (as defined in Section 3.2) can be implemented in the proposed PDMS logical architecture by using isolated and confidentiality protected data tasks which are controlled and attested by the Core. As shown in Fig. 5, the main modules needed for this type of computations are the *reference monitor* and *audit* modules to ensure that only proper data access is performed, and the *attestation* module to obtain global integrity guarantees for the computation.

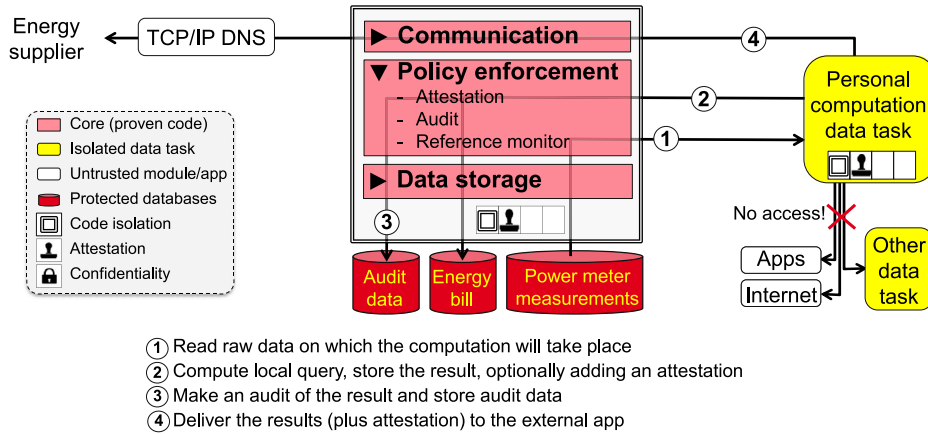


Fig. 4. Bilaterally trusted personal computations.

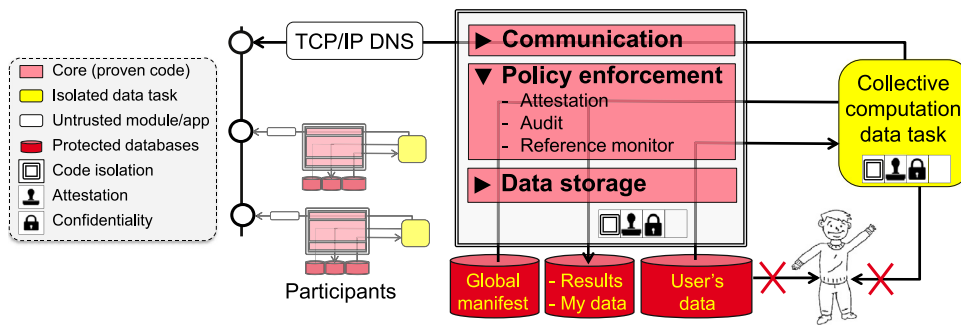


Fig. 5. Mutually trusted collective computations.

4.2.5. Controlled data dissemination

A PDMS must provide a secure way to capture the decisions of the owner, accordingly implement and enforce the resulting effects on data dissemination, and help users to understand these effects and apprehend what effectively happens during data dissemination. Of course, the overall process is highly complex and opens up several challenges (see Section 5, Challenge 2).

Our goal here is to introduce the main architectural elements induced by *controlled data dissemination* and guarantee the two points of the corresponding security property as defined in Section 3.2.

First, securing a decision impacting data dissemination forces the PDMS to provide means for users to truthfully view the needed information to make the decision (e.g., view a manifest describing a computation, the underlying personal data to be authorized, etc.), capture it (at the very least, a button to accept/decline that manifest) and potentially check its effects and/or audit its effective use. From a security viewpoint, this means relying on the use of visualization software (e.g., file or image readers, up to more complex visualization tools according to the semantics of the manifest or the audit trails), which may contain external code with security breaches that could be exploited by an attacker. Therefore, this code cannot be part of the Core, but should be executed as part of a data task, called here a *decision-making* data task. In addition, in order to be sure that the code of this data task behaves as expected, the integrity of the execution should be guaranteed against any potentially malicious external entities or corrupted runtime environments. Moreover, the decisions which are made may also be considered personal for the PDMS owner or depend on personal information, and should thus not be leaked outside the PDMS (except through a deliberate intervention of the owner). To ensure both the integrity of the execution and the confidentiality of the decision, decision making data tasks must be run in *isolation*.

Beyond this, making decisions by nature relies on interactions with the PDMS owner via peripherals (e.g., screen, keyboard). This requires *peripherals isolation* to ensure that the data exchanges between the peripherals and the data task cannot be altered nor leaked outside the data task. Subsequently, in order to correctly interact with decision making data tasks, the Core must ensure that the data task effectively runs the expected (original) version of the code. This requires authenticating the decision-making data task through the *authentication* module integrated into the Core (already introduced above).

Second, all the decisions must be unconditionally enforced. The decisions captured by the PDMS are thus translated into low level data dissemination policies and enforced by the Core. Enforcing decisions typically leads in our system to implement access control policies managed by the access control module and relying on a trusted *reference monitor* to enforce the effects of these access control policies. Of course, the form of the policies (e.g., access control lists, execution privileges on computations, etc.) and their use typically depend on the decision to be enforced.

We argue that the overall architectural design presented on Fig. 6 offers an interesting backbone to operate data dissemination decisions in the PDMS context. This requires a *reference monitor* to store and enforce decisions, and an *audit* module to store audit data on the effects of these decisions. We give two example below showing how controlled data dissemination could be operated in the case of the recovery protocol described in Section 4.2.2 and in the case of local computations presented in Section 4.2.3:

Controlling data dissemination in data recovery. When a PDMS owner Alice wants to share a secret share S with another PDMS owner Bob, a decision-making data task with access to the contact files of Alice's PDMS is launched, is authenticated by the Core of Alice's PDMS and displays a list of Alice's contacts from which she chooses Bob as a recipient of the secret share S. The effect of that

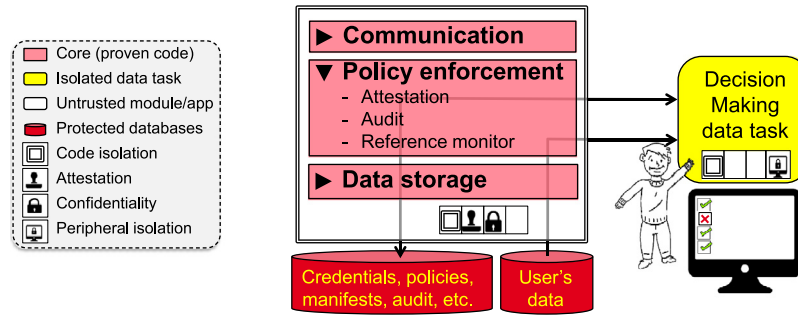


Fig. 6. Controlled data dissemination.

decision on the *reference monitor* of Alice is to grant a read privilege on S to Bob's PDMS. On Bob's PDMS, Bob makes the decision to accept Alice's secret share S (using a *decision-making* data task). As a result, Bob's *reference monitor* grants a read permission on S to the *recovery* data tasks (no other data task nor Bob himself have granted access to S). At recovery time, Alice contacts Bob (out-of-band communication) and Bob's PDMS contacts Alice's the new PDMS. Bob accepts his PDMS to declassify S to Alice's using a *decision-making* data task and assuming that B confirms to its PDMS that the recipient of S is indeed the new PDMS owned by Alice.

Controlling data dissemination in local computations. Alice wants to authorize her energy supplier *EnergySupp* to run a data task *consum* which aggregates the electricity consumption of Alice on the last 30 days. A *decision-making* data task shows the manifest of *consum* to Alice which includes a description of the raw energy data taken as input (energy trail of the last 30 days, called *raw-data-30*). Alice accepts it and the resulting effect on Alice's *reference monitor* are an *execute* privilege on *consum* granted to *EnergySupp*, a read privilege on *raw-data-30*, a write privilege on its result to *consum*, and a read privilege on the result to *EnergySupp*. Alice's PDMS also audits the execution of *consum* when it is launched by *EnergySupp* and the access to its successive results by *EnergySupp*. Alice, when looking at the audit trail realizes that *EnergySupp* accesses *consum* every day and is thus able to infer her energy consumption on a daily basis. In consequence, Alice may either revoke this decision or apply a 'fix' (which could have been provided by *EnergySupp* or by a community of users) to update the read privilege on *raw-data-30* of *consum* such that only a fixed size batch of data is included in *raw-data-30* (e.g., the data of the previous month).

4.3. Overall architecture design

The union of the above-mentioned building blocks constitutes a baseline for a logical architecture to manage the data life cycle in a personal cloud, while answering the main security goals.

Depending on its type, each data task requires an execution environment that provides the appropriate *common security primitives* introduced in Section 4.2, namely *Isolation*, *Attestation*, *Confidentiality* and *Peripherals isolation*.

Ideally, the code of the Core must be formally proven to avoid security breaches leading to unexpected behaviors. Equally important, the Core must run in an execution environment that satisfies isolation, attestation and confidentiality. The isolation property is required for the Core to protect it from all the other software components running on the same personal cloud platform (in particular the Apps and the data tasks). The attestation property is required for the collective computations which are orchestrated and/or executed by the Core. Finally, to provide mutual guarantees of security between PDMS users and third parties (see Section 3.2), the environment of the Core also has to provide confidentiality since it coordinates the distributed data tasks with potential access

to private personal data supplied by other nodes, which remain hidden from the PDMS owner. This leads to the architecture presented in Fig. 7.

4.4. Concrete PDMS instances

The goal of this subsection is to show that the logical architecture presented above can be instantiated in practice, using existing software and hardware solutions, and that its modular aspect helps defining physical PDMS instances easily. We first discuss here existing security primitives (namely *isolation*, *attestation*, *confidentiality* and *peripheral isolation*). Second, we show how to combine them into physical ES-PDMS architectures which instantiate three different configurations: (1) PDMS on a home box, (2) on a mobile device and (3) in the cloud. Third, we provide an example of a preliminary implementation of the proposed architecture.

There is no unique way of implementing the common security primitives on which our architecture relies, but rather various solutions with different guarantees regarding the enforcement of these primitives. We investigate below existing software and hardware targets.

Software-based security solutions. Let us first consider the security properties which could be provided by pure software-based solutions. The first of these properties is *isolation*, which is intensively investigated as it constitutes a foundation of secure software architectures. Isolation is usually provided by an operating system (e.g., Linux, seL4, etc.), a virtual machine monitor (VMM or hypervisor, e.g., XEN, KVM) or a container manager (e.g., Docker, Kubernetes). Such solutions have the advantage to provide a high level of *extensibility* (in the sense that potentially complex/external code can be run). But enforcing the isolation security primitive means considering the underlying software components (the OS, VMM or container manager) as part of the trusted computing base (TCB), i.e., critical part of the software that, when compromised, can jeopardize the security of the entire system. The TCB must therefore be made completely free of vulnerabilities (e.g. bugs, buffer overflows, etc.) and ideally must be formally proven. This is a difficult task with large and complex code. Existing solutions rely on reducing the attack surface by minimizing the code which is part of the TCB (e.g., microkernels like seL4 and unikernels like MirageOS) or by hardening it (e.g., by adding access control, an IDS or a firewall [23]). Today, software solutions offer some form of *isolation* and *peripheral isolation* (discussing to which extent being beyond the scope of this paper). However, assuring strong isolation guarantees in software is still an open issue (see [16] for a recent survey) and side channel attacks remain prominent [15] (thus, the mention 'satisfied with limitations' in Table 3). Moreover, since using software as a root of trust is still an unresolved problem [24], *confidentiality* against the PDMS owner and *attestation* of the code execution cannot be achieved purely through software. As a conclusion, the advantage of using VMMs in terms of *extensibility* is

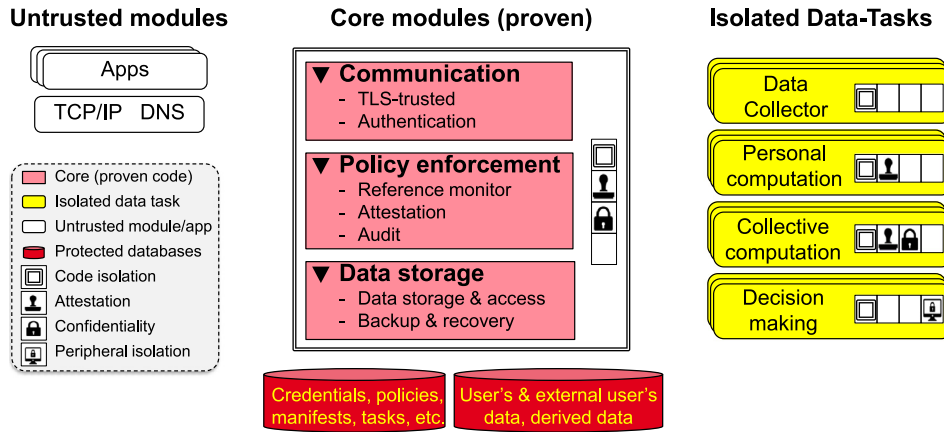


Fig. 7. Logical architecture.

Table 3
Claimed properties of different execution environments (left) and required properties of the different modules of our ES-PDMS architecture (right).

Architecture	Code isolation	Attestation	Confidentiality	Peripherals isolation	Extensibility
Secure Element	✓	✓	✓		
TrustZone	✓	✓		✓	≈
Intel SGX	✓	✓	✓		✓
Software	≈			≈	✓

Modules	Code isolation	Attestation	Confidentiality	Peripherals isolation	Extensibility
Core	✓	✓	✓		
Data collector	✓				✓
Personal computation	✓	✓			✓
Collective computation	✓	✓	✓		✓
Data dissemination	✓			✓	✓
Applications					✓

The properties can be satisfied (✓) or satisfied with limitations (≈)

obvious, but such solutions should be considered in combination with other solutions (typically, secure hardware) to achieve the desired confidentiality and attestation properties of a PDMS (as summarized in Table 3).

Hardware-based trusted execution environments (TEE). Hardware TEEs target many different kinds of devices, from personal computers and IoT devices to cloud servers. The most prominent TEEs include secure elements as SIM cards (e.g., in smartphones), ARM TrustZone [25] for SoCs and CPUs integrated into smartphones, tablets and smart appliances, and Intel SGX [26] embedded in all recent Intel CPUs present in personal computers and cloud servers. Although there is still no unique security definition of TEE and their capabilities vary depending on proposals [27], most TEEs offer capabilities to run code in isolation and remote attestation, which allows it to prove required properties of the code running to third parties. Table 3 summarizes the security tools offered by three technologies: SGX, TrustZone and secure elements (smartcard). These three solutions mainly vary in the way confidentiality and peripheral isolation are achieved and in terms of the possible level of extensibility of the code they can run:

- **Intel SGX [26]** integrates cryptographic primitives in hardware to isolate applications within enclaves, while providing confidentiality and attestation of the code executed in the enclave. Additionally, it provides mechanisms for managing the information flow from code running in the TEE to the outside world, both secure external storage primitives thanks to encryption and management of the communications between

processes running in TEEs. It can be used both for securing Cloud apps and in the context of personal computers, leading to an advanced form of extensibility.

- **ARM TrustZone** technology isolates sensitive code executed in the *secure area* of the CPU, from application code executed in the *rich area*. It also aims to *isolate the device's peripherals* (typically, the screen of a smartphone) once code inside the secure area is executed. The code executed inside the secure area has some limitations in terms of resources (e.g., TrustZone ARM1176JZ based on Cortex A series is clocked at 772 MHz and can access several tens of MBs of RAM) leading to a certain form of *extensibility*.
- **Secure elements**, on the other hand, offer much less resources (e.g., advanced secure elements like ST33 based on ARM SecureCore SC300 Cortex M series is clocked at 60 MHz and has only 50 KB RAM), thus having a negative impact on extensibility, if used to run a data task. In terms of security, on the contrary, such components provide – in addition to isolation and attestation – confidentiality (with strong guarantees due to tamper-resistance) for running code and data.

The common security primitives needed by the different modules of our ES-PDMS architecture and the primitives provided by these TEEs are reported in Table 3. Several conclusions can be drawn regarding the implementation of the logical architecture we envision: (i) it cannot be implemented using a single technology since none currently ensures all the required properties,

and therefore has to combine several elements; (ii) the Core relying on isolation, confidentiality and attestation, can be run on a secure element or on SGX, or be implemented by a combination of execution environments (e.g., pioneer works like TrustVisor [24] propose a secure hypervisor based on combining a secure element with an hypervisor to provide confidentiality and attestation); and (iii) the data tasks needing peripherals isolation (i.e., decision-making) can only be run on TrustZone or a hypervisor/VMM.

An important remark concerns another implicit property in our architecture, which cannot be directly ensured by secure hardware alone. That is the data tasks are exclusively controlled by the Core and the communications between the Core and the data tasks are protected, despite the OS (executing said tasks, e.g., within an SGX enabled CPU) being potentially compromised. In itself neither attestation nor isolation directly provide this property, and some code encapsulation has to be provided in order to make sure that the input/output behavior of the data task cannot be observed by the untrusted OS. A solution for SGX enclaves can be found in [28]. It leverages secure channels and the attestation mechanisms provided by SGX in order to allow for protected execution of an arbitrary remote computation with confidentiality of the input/output behavior.

The above listed software and hardware solutions can be combined in many ways to achieve the security primitives needed to implement the proposed ES-PDMS logical architecture, but a complete study of the potential architectures, their impact on the data management tasks and their limitations, constitute open challenges. In the following, we limit to presenting a few basic examples of physical architectures encompassing the security properties of an ES-PDMS.

Physical architectures of an ES-PDMS. In Fig. 8 we propose three illustrative physical instances of our logical architecture adapted to three different ES-PDMS configurations based on a home box, a personal device and the cloud:

- **Home box** (Fig. 8a). The data is stored in the box and is cryptographically protected. The Core runs on a secure element as well as the collective computation tasks¹⁰ to benefit from its security and achieve the attestation and confidentiality requirements. Personal computation and data collector tasks are executed on a TrustZone CPU equipping the box, and thus benefit from extensibility. If no peripheral is available on the box to interact with the user (e.g., no touch screen), the apps and the decision-making tasks run on the smartphone of the PDMS owner, in the rich area and the trusted area respectively, to safely capture the user's I/O.
- **Mobile device** (Fig. 8b). Similarly to the home box, the Core and collective computation tasks are operated in the secure element of the smartphone (i.e., an additional SIM card slot is needed), the other tasks (personal computation, data collector and decision-making) run in the trusted area of the TrustZone CPU of the smartphone, while the apps run in the rich area of the CPU.
- **Cloud** (Fig. 8c). The SGX-based instance pictured on the right part of is running both the Core and the data tasks in distinct SGX enclaves. The enclave running the Core takes advantage of the attestation capabilities of SGX to control the other enclaves, and collective computations can be performed seamlessly (as well as other personal computation and data collector tasks) given the confidentiality property offered by SGX,

with the Core relaying the remote attestation guarantees to the other participants of the protocol. To protect the user's I/O with the decision-making console, this architecture also needs a smartphone with a TEE (e.g., ARM TrustZone).

Before concluding this section, we describe a preliminary implementation which prefigures the home box architecture instance presented above and constitutes a reasonable base for a first validation of the ES-PDMS concept.

A concrete ES-PDMS instance. A consortium, built around Inria, UVSQ and a French company, has been set up in 2018 with the goal to conceive and deploy a secure decentralized social–medical folder facilitating the coordination of social and medical care at home for elderly people. In a first step, 10 thousand patients from the Yvelines district in France are targeted. In terms of hardware, the solution is close to the home box case presented above. The box is based on a combination of a secure element (SE) hosting the Core and a microcontroller hosting basic data tasks. Applications dedicated to social and medical workers as well as the patient's decision making app run on smartphones. The microcontroller is a STM32F based on an ARM Cortex-M4 CPU with 168KB of RAM and 1MB of embedded NOR where data tasks code resides. The Core itself will run in a ST33J MCU, a certified tamper-resistant SE (Common Criteria EAL6+) equipped with an ARM SC300 CPU, 50 KB of RAM and 1 MB of NOR, where the Core code and metadata linked to its secure functioning reside. The porting of the Core on the ST33J is under way, while in the current version of the box a reduced, essential set of security properties is provided by a TPM (Trusted Platform Module). The three main modules of the Core are implemented as follows:

- **Data storage.** The data storage module incorporates an embedded database engine based on PlugDB [11,12,29] which manages an encrypted database hosting the personal data on a large Flash memory (GBs microSD card) and a recovery procedure based on the protocol described in Section 4.2.2.
- **Communication.** The communication module includes an authentication procedure relying on certificates delivered by a PKI and a preliminary implementation of *TLS trusted* to communicate with the remote services (the application of the practitioners running on their smartphones and external servers hosting social–medical personal data).
- **Policy enforcement.** The policy enforcement module is based on RBAC policies defining differentiated views of the personal data to the connected user depending on his role (physician, nurse, social worker, etc.), with a reference monitor enabling the PDMS owner to visualize and control all the enacted authorizations [30] from her smartphone.

The choice of a STM32F microcontroller to run the data tasks has been dictated by economic and energy saving constraints. It limits the scope of the box to a reduced set of predefined *Data collectors* and *Personal computations*, with strong assumptions made on isolation (i.e., considered in our specific context of a rather 'closed' and ad-hoc platform). *Collective computations* are not implemented yet. Since collective data tasks cannot be operated inside an MCU which does not protect data confidentiality, they will be operated inside the secure element as simple sequences of SQL queries evaluated by the Core. Although within this architecture, the level of extensibility is still limited and specific security assumptions are made, the limitations are clearly identified and overcoming them is an integral part of the roadmap. More powerful versions of the box are already planned, to support a full range of data tasks with no impact on the global design.

In conclusion, several existing technologies implement today, at least partially, the security properties required by the proposed logical architecture and combining them to obtain appropriate

¹⁰ The collective computation data tasks being operated on a secure element are highly secure (tamper resistant) at the price of extensibility. However, advanced distributed data processing tasks must be done with the support of a more powerful CPU integrated in the box and connected to the secure element, which is an open issue.

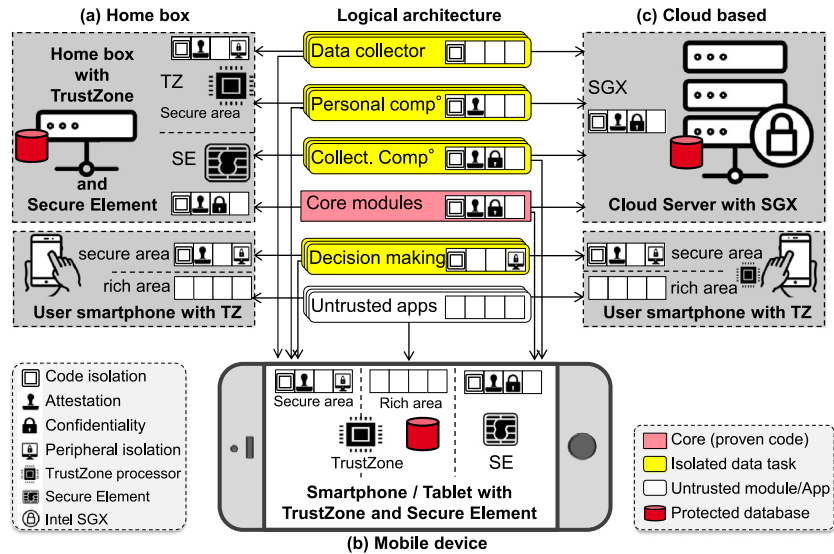


Fig. 8. Physical ES-PDMS instances: (a) home box; (b) mobile device; (c) cloud based.

physical instances of an ES-PDMS in different usage contexts is already possible. Regarding the near future, the current trend suggests that the availability and diversity of TEE technologies will increase. New solutions are already envisioned, like heterogeneous multicore platforms [31] in which security/isolation oriented cores (*à la* SGX) would cohabit with other all-purpose cores, allowing for separation of tasks inside the CPU. We expect the modularity of our logical architecture to be of great help in accommodating new physical instances for such upcoming solutions. While this opens to practical ES-PDMS instances, database challenges linked to the underlying data management features and the PDMS context specificities need to be explored. We discuss below some dimensions of the problem, and introduce certain database challenges linked to the architecture in Section 5.

4.5. Discussion

Different practical solutions based on secure hardware can already give rise to some form of ES-PDMS. However, the problem of implementing an ES-PDMS is large and as such there are important related issues which have to be considered as well. We discuss below some dimensions of the problem and limitations of our proposal, not addressed so far for simplicity and conciseness reasons, or because they are rather orthogonal to the architectural dimension considered as the cornerstone of the ES-PDMS introduced in this paper.

A first aspect concerns the security level of a physical instance used to deploy our logical architecture. We discussed above a few examples of instances by taking into account the required set of security primitives (i.e., a physical instance is valid if it covers all the required properties of the targeted ES-PDMS module) but without a deeper consideration of the security guarantee level associated with the properties. Thinking that a security property can hold in any condition is wishful even for TEEs. Generally, it is wise to think security in terms of an attack's cost/benefit ratio. From this viewpoint, a Cloud based PDMS instance (e.g., running on Intel SGX) could be considered as more exposed to attacks than a home box instance since in the former case the physical platform is shared by many PDMS instances (although each PDMS uses its own dedicated enclaves), while in the latter case the physical platform is dedicated to running a single PDMS. Also, different environments can offer different protection levels for a same security primitive. For example, a software implemented security primitive

is generally considered to have a weaker security level than if it were hardware implemented (e.g., TEE), while some TEEs can be considered more secure than others (e.g., smartcard versus ARM TrustZone).

A second important aspect specific to our architecture is to ensure a clean separation between the application level and the data computation level. Indeed, the apps are considered untrusted in our architecture mainly because one cannot trust in general the user computing environment (typically the OS and browser). Therefore, the apps should ideally have access only to the computation results and never to the raw data. Hence, app developers should push as much as possible of the complex app-related data computations into data tasks leaving thus the sensitive task of app policy enforcement to the Core. A few recent works in the domain of web application security are following a similar approach. For example, systems like Amber [32], DIY [33] and π Box [34] consider isolating and sandboxing application subparts to increase security and user's control with untrusted applications. Amber [32] separates web services from data processing by introducing a data-sharing model relying on secure distributed query mechanisms implementing a reduced subset of SQL. Following a similar approach, π Box [34] proposes to split the app into client and cloud sides, establishes restricted communication channels (read-only, write-only, collect statistics) between the two and relies on sandboxes (also coupled with differential privacy) to counter attacks conducted through applications. Our three-tier logical architecture generalizes this separation principle, i.e., application versus data computation, and thus lays the foundation of a framework enabling a number of good security practices for app developers. Besides, we note that the guarantees offered by the above mentioned frameworks are inherently weaker than the guarantees provided by secure hardware. Thus, applications implemented using these frameworks can essentially be considered as weakly secure, based on extensible data tasks satisfying isolation and peripheral isolation (provided the user's web browser is trusted). Also, these frameworks do not provide confidentiality and attestation, two important security properties for an ES-PDMS, which further underlines the importance of secure hardware in this context.

A third aspect is related to data destruction and more generally the usage control offered by the ES-PDMS. In terms of functionalities, we focused in this paper on five main stages of the personal data life-cycle. One might object that data destruction represents also an important stage and should be considered as such. While

we agree with this observation, we argue that data destruction raises issues that are mainly outside the scope of the architectural study of this paper. For example, in the case of secondary copies of user data (e.g., email and other social interaction data, health, employment, insurance), the main issue is not deleting the copies of the data collected into the PDMS but destroying the original data managed by the external service that generated it. Thus, the problem shifts to a legal issue, e.g., see the “right to be forgotten” on the Internet in the European legislation. Also, regarding the primary copies of some user data (e.g., quantified-self data, smart home data, photos, videos and documents generated by the user), a general problem appears if the users disseminates the data to other users. In this case, a form of user control can be maintained if the data is shared with another ES-PDMS (e.g., a sticky policy is associated to the shared object and is enforced by the reference monitor module of the PDMS). However, there are major limitations in practice, e.g., how can one preclude a malicious user from making a screenshot when she visualizes a photo that has been shared with her. Hence, data destruction leads to important but very challenging problems such as enforcing usage control or making users aware of their responsibilities.

Many other important problems remain to be addressed such as personal data visualization, data integration, legal, economic or cognitive aspects, or user perception regarding security. Although quite independent of architectural issues, we argue that the extensible architecture introduced in this paper offers interesting means (through data tasks) to tackle such issues without decreasing the security level of the solution. In the following section we discuss a few important challenges from the data management perspective and from an architectural angle.

5. Research challenges towards an ES-PDMS architecture

Without claiming to be exhaustive, we believe that at least three main challenges arise from the core of our analysis of the architectural vision introduced in the paper. The first challenge directly stems from the reference architecture presented above and concerns its formal analysis. The second challenge relates to the control tools derived from the enforced position and responsibilities acquired by the PDMS owners. The third challenge concerns the concrete implementation and enforcement of distributed data processing without leaks using Trusted Execution Environments. These three challenges are described below.

Challenge 1: Design and formal definition of the Core engine.

As sketched in Section 4, the Core engine is expected to provide at least authentication and secure personal data storage and access, to enforce security and privacy policies, and to sequence the execution of data tasks installed under the user’s control. While existing work (e.g., relational algebra, iterator model and classical query rewriting to access data) can be reused to this end, a particular focus must be put on minimizing the code size and complexity such that the Core engine can be proven through formal methods. This leads to delegate non-critical operations to data tasks potentially based on existing –non proven and large – code modules or libraries, interacting with the Core without jeopardizing the global security of the computations.

A modular architecture such as the one we propose allows to construct independent definitions of the security goals, rigorous proofs for each Core component, and rely solely on the security of the Core and on the security properties of the data tasks (rather than on their implementation). While such an approach is common in cryptographic protocol design, the complexity lies in the large amounts of data and large scale of the system considered here, compared to usual cryptographic protocols. The modularity of our architecture can help in solving these issues, as designing provably secure components allows abstracting away lower level details.

A good example of such a modular design is the secure map-reduce framework VC3 [35] which builds on top of SGX’s local isolation and attestation guarantees to prove security and integrity of the whole computation. A similar approach could be adapted to the combination of components described in this paper. Solutions already exist to model the low-level properties of data tasks as described in Section 4.1 (e.g., [18,28]), which could form a solid foundation for higher-level properties.

Challenge 2: Controlled data sharing. Defining a well-calibrated data sharing policy, and enforcing it despite piracy actions, requires an expertise which is out of reach of most individuals. Of course, new data sharing and usage control techniques dedicated to personal cloud lay users should be invented. Although this issue is general and might go beyond the strict case of the PDMS context [36], a new challenge specific to the PDMS context is the one of making the PDMS owner able to define her own sharing policy without being forced to understand the underlying access control semantics. To this end, the administration of data sharing should rely on three properties: (i) *visualization*: whatever the underlying access control model, the owner should have the capacity to visualize the net effects of these rules and to easily adjust them if required; (ii) *trusted reference monitor*: the reference monitor logic enforcing the sharing policy must itself be understandable by the holder and the platform implementing this logic must be trusted by her; and (iii) *zero-knowledge grants*: side channels should be proscribed to avoid leaking personal information within authorized sets of permission. Each of these properties leads to specific challenges.

Visualization. Regarding the first point, existing works like π Box [34] promote the idea of letting users visualize the effects of their decisions before validating any sharing action. In the personal cloud context however, the amount of permissions can be huge. A first issue is thus to assist the holder in this validation task. A recent study [30] makes one step in this direction by identifying suspicious grants, thanks to ‘watchdog triggers’ comparing new authorizations with previous ones to identify outliers. A second issue is to help the owner regulate, and then visualize, the complete lifecycle of her personal data, e.g., from their capture to their dissemination, all the way to their deletion while the data may undergo transformations (e.g., aggregation, anonymization) at some steps of this lifecycle to reduce its sensitivity. Moreover, the effects of some rules are challenging to visualize (e.g., time or location-based contextual rules). Hence significant work remains to be done to define such friendly visualization tools. The challenge is important since providing a visual feedback to the individuals about the way they are exposed greatly helps them to adapt their behavior [37,38].

Trusted reference monitor. As shown in Section 4, visualization tools only make sense if the holder can trust what is actually plotted. This means designing visualization tools that can be integrated in data tasks providing peripherals isolation, and provide to the PDMS owner means to trust the reference monitor to precisely enforce the effect visualized. Therefore, the reference monitor logic must be basic to be easily understood by the holder and must be integrated in the Core and run inside a TEE. A primary solution is to materialize all permissions by means of ACLs (i.e., user u is granted access a to resource r if $(u, r, a) \in \text{ACL list}$), whatever the complexity of the underlying access control model producing these ACLs. This leads to a materialized view maintenance problem in the TEE and to an embedded data management problem to minimize the reference monitor overhead at data access time.

Zero-knowledge grants. A third issue is to proscribe by design any side channel to be created between the personal cloud and a remote party. For example, a set $\{(u, r, a)\}$ of authorizations could be considered acceptable by the PDMS owner when each triple (u, r, a) is examined individually, but may reveal unexpected sensitive information from the PDMS through a side channel when considered globally. For example, imagine a sharing model extracting

from Alice's PDMS a set of ACLs authorizing her friend Bob to access to certain photos. The set of photos authorized to a Bob could be legitimate when considered individually (i.e., a legitimate permission, granted to a legitimate subject, on a regular object), but could globally leak information about the content of Alice's PDMS (e.g., the photos being ordered on size, the values of the i th pixel of each picture may be used as a side channel and leak the credit card number of Alice). Building on the proposed architecture, specific system data tasks could be imagined to circumvent such issues. Typically, verification techniques based on the replay of the data tasks checking the resulting permissions on a 'what if' basis (e.g., replay the sharing rule on a dataset of photos with and without the banking information) may help revealing such side channels. Of course, this problem is not limited to the PDMS context, but is of great importance here as data sharing decisions are made by lay users on the basis of their entire digital assets.

Challenge 3: Secure distributed data-oriented computations using trusted execution environments. The personal cloud paradigm allows for novel big data applications on personal data (e.g., participatory sensing, epidemiological studies, personalized recommender systems, etc.). More precisely, the objective of distributed database computations in the PDMS context is being able to compute any function taking as inputs the private values of n PDMS owners without leaking any information other than the final result. This is close to the well-known secure multiparty computation (MPC) problem. While secure distributed –personal – data processing is not a new issue, the architecture promoted in this paper introduces new specificities which open up to new possibilities but also raises new challenges. First, the PDMS architecture is distributed at the individual level and is expected to scale up to nationwide populations. While MPC protocols have been widely studied in cryptography, they have not been designed with scalability in mind and large-scale solutions exist only for a very limited set of functions [39]. Second, the security of the architecture we propose relies on secure hardware (or more precisely, Trusted Execution Environment as described in Section 4.4), which exhibit specific constraints with a potential impact on data management structures and algorithms, and calls for new threat models when compared to traditional MPC. Secure and efficient data processing in this context leads to specific challenges including the following:

Integrity and confidentiality of generic hardware-based distributed computations. Relying on the properties of the proposed architecture inherited from a TEE opens up the field for generic and scalable solutions, but has major differences with traditional MPC. First, the adversary model is different. Traditional MPC considers an *honest-but-curious* adversary model, which makes sense when the risk of being blacklisted is a strong enough deterrent to force the participating entities (e.g., central servers) to not deviate from the protocol. Under the assumption that the data tasks involved in the distributed computation are isolated thanks to TEEs, even in the presence of a curious or malicious PDMS owner, the data task sticks close to a *fully honest* behavior. A first research issue is thus to propose a way to map any arbitrary distributed protocol within an understandable manifest specifying the overall orchestration of the computation (i.e., specifying the role and privileges of each data tasks involved), and ensure at execution that the computation protocol is respected as stated in a manifest. This is not an easy task for several reasons. First, the secure execution of the distributed computation protocol may be enforced on a *local correctness checking* basis, i.e., each participant involved in the computation should be given means to check the correctness of the distributed protocol from its local view and be guaranteed that, if the local check is verified, the global output is also correct. Second, in the case of many data oriented operations (e.g., hashing or sorting as part of joins, group by or duplicate removal) the data flow specified between the data tasks is not deterministic and independent of

the data content, but can lead to transferring data to a subsequent data task depending on the data value (e.g., Alice's data tasks transmits data to Bob's data task according to a hash of the data value) which may reveal personal information to an attacker able to observe the data exchanges. *Counter measures* should be thus envisioned to enrich the manifest, e.g., with pre-processing data sampling steps or introduction of dummy data exchanges into the manifest. Third, while TEEs are supposed to guarantee the confidentiality of the data task content, this security property cannot be considered in practice as unconditionally unbreakable. Indeed, even if secure hardware is used, a fraction of the participating PDMS owners may have instrumented their PDMS platform and exploit side channel attacks to retrieve the content of the data tasks they are in charge of. Such attacks should be taken into account when building the computation manifest, to avoid letting an attacker compromise a large set of users' privacy or being able to target a specific user. A new expected property should thus be introduced, called *data locality*, specifying that any information in possession of any involved *data task* should be as close as possible to the personal data contributed by the PDMS owner of that data task. This property is essential to reason about corruption, define relevant metrics quantifying the potential impact on data leakage of colluding participants with instrumented hardware, and design and integrate appropriate counter measures in the manifest to minimize the benefit-to-cost ratio of such attacks.

Performance of large scale data-oriented operations under the TEE constraints. Considering the PDMS architecture presented in Section 4, an important challenge is obviously to investigate the use of Trusted Execution Environments (TEEs) available at user side (e.g., Intel SGX available in PCs, ARM TrustZone in smartphones/tablets and home boxes). In the database area, there is an intense research agenda on data management for new and/or specialized hardware [31,40] and systems like Oracle M7 even start to provide hardware implementations of rich sets of database operations (SQL in silicon). But the focus is on secure centralized databases or on the database-as-a-service (DaaS) context. Typically, TrustedDB [41] considers a database running on a tamper proof dedicated secure hardware, Cypherbase [42] adjuncts a secure hardware to an SQL-Server database and EnclaveDB [43] considers a transactional database running inside an SGX enclave. The typical limitations of the cryptographic overhead of accessing persistent data outside the TEE enclave [44], the limited RAM amount of each TEE enclave [26], the cost of external function calls [45] and memory access overheads [46], may slow the computing by orders of magnitude compared to a regular environment, and have to be taken into account. Preliminary works in this direction try to take advantage of several Intel SGX enclaves [26,35] to parallelize the processing in a secure manner. The problem mentioned above of enforcing the *local correctness checking* and *leakage resilience* properties hence turns the decentralized challenge itself into a performance issue. A precise analysis of TEEs enclaves w.r.t. data-oriented operations in a distributed context still remains to be done and will undoubtedly raise many interesting data management challenges.

6. Conclusion

The personal cloud paradigm is coming at a rapid pace. The goal is to empower individuals with their personal data and to unlock new usages founded on the use of their personal data under their control. The recent worldwide smart disclosure initiatives and privacy regulations offer great support for the personal cloud movement. However, the journey is not without danger and many solutions have taken a path that eventually leads to empower users in a way which may exacerbate privacy risks.

A first approach was to consider that the personal cloud context could be addressed with a simple adaptation of classical corporate cloud techniques, hence the proliferation of online personal cloud services. This is unfortunately not the case. While corporate solutions address a carefully controlled set of applications and are tuned towards data management and security experts, the PDMS context sketches an open and rich ecosystem of untrusted data processing apps in interaction with an unsecure execution environment and a layman PDMS owner. Trying to answer the fear associated with online personal clouds, zero-knowledge and home cloud solutions emerged but in fact, they transformed the trust assumption in the personal cloud provider into the myth of the owner's self-capacity to administer and guarantee the security of her own environment.

Solving this issue is both a scientific, technical and societal challenge. On the scientific side, the primary challenge addressed to the data management and security research communities is to define and formally analyze a personal cloud architecture capable of combining strong expectations in terms of security with extensible data management capabilities covering the complete personal data life cycle. Our definition of an Extensive and Secure Personal Data Management System (ES-PDMS) along with a set of security properties and reference architecture is a first step towards this goal but many exciting research issues still need to be investigated. On the technical side, we have sketched different alternatives for instantiating this reference architecture into concrete platforms but a long road remains before achieving operational solutions exploiting the full capacity of advances in secure hardware. Some steps along this road are undoubtedly difficult enough to introduce new research issues on their own. Finally, the societal challenge is multidisciplinary: how to devise new economic models reconciling lucrative (or non-profit) exploitation of personal data and user's empowerment? How to make the legislation evolve in relation with the evolution of PDMS architectures, notably regarding the sharing of liability between the PDMS provider, the application providers and the owner herself? And finally, how to convince people of the importance of all the previous questions and provide them with the right choices? Architecture also matters when devising solutions to these questions. It must provide formal guarantees that the interest of each party will be preserved, that each party will get the appropriate tools to endorse her own responsibility and finally it must be as transparent and friendly to use as possible.

Hence, we believe that the personal cloud paradigm is a cornerstone of the digital evolution and that many of its challenges are rooted in architectural choices. Our hope is that this paper will provide a solid foundation for the discussions around this promising topic and lead to impactful and original research.

Acknowledgments

This research is supported by the Inria Innovation Lab 'Own-Care' and by the ANR PerSoCloud grant no ANR-16-CE39-0014.

References

- [1] W. Christl, K. Kopp, P.U. Riechert, *Corporate Surveillance in Everyday Life*, Cracked Labs, 2017.
- [2] Task Force on Smart Disclosure. Smart Disclosure and Consumer Decision Making. Report, NST Council, 2013.
- [3] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation, GDPR).
- [4] S. Abiteboul, B. André, D. Kaplan, *Managing your digital life*, CACM 58 (2015) 5.
- [5] T. Allard, N. AnCIAUX, L. Bouganim, Y. Guo, L. Le Folgoc, B. Nguyen, P. Pucheral, I. Ray, I. Ray, S. Yin, *Secure personal data servers: a vision paper*, in: VLDB, 2010.
- [6] Y.A. deMontjoye, E. Shmueli, S.S. Wang, A.S. Pentland, *OpenPDS: Protecting the privacy of metadata through safe answers*, PLoS One 9 (2014) 7.
- [7] A.P. Dalskov, C. Orlandi, *Can you trust your encrypted cloud?: An assessment of SpiderOakONE's security*, in: ACM Asia Conference on Computer and Communications Security, AsiaCCS, 2018.
- [8] A. Chaudhry, J. Crowcroft, H. Howard, A. Madhavapeddy, R. Mortier, D. Haddadi, H. McAuley, *Personal data: thinking inside the box*, in: Aarhus Conference on Critical Alternatives, 2015.
- [9] R. Mortier, H. Haddadi, T. Henderson, D. McAuley, J. Crowcroft, *Human-data interaction: the human face of the data-driven society*. Available at SSRN 2508051, 2014.
- [10] N. AnCIAUX, P. Bonnet, L. Bouganim, B. Nguyen, I.S. Popa, P. Pucheral, *Trusted cells: A sea change for personal data services*, in: CIDR, 2013.
- [11] N. AnCIAUX, L. Bouganim, P. Pucheral, Y. Guo, L. Le Folgoc, S. Yin, *MILo-DB: a personal, secure and portable database machine*, *Distrib. Parallel Databases* 32 (1) (2014).
- [12] S. Lallali, N. AnCIAUX, I.S. Popa, P. Pucheral, *Supporting secure keyword search in the personal cloud*, *Inf. Syst.* (2017) 72.
- [13] C.Q. To, B. Nguyen, P. Pucheral, *Private and scalable execution of SQL aggregates on a secure decentralized architecture*, *TODS* 41 (2016) 3.
- [14] D.H.T. That, I.S. Popa, K. Zeitouni, C. Borcea, *PAMPAS: Privacy-aware mobile participatory sensing using secure probes*, in: ACM International Conference on Scientific and Statistical Database Management (SSDBM), 2016.
- [15] M.M. Bazm, M. Lacoste, M. Südholt, J.M. Menaud, *Side Channels in the Cloud: Isolation Challenges, Attacks, and Countermeasures*. Preprint, 2017.
- [16] D.A. Fernandes, L.F. Soares, J.V. Gomes, M.M. Freire, P.R. Inácio, *Security issues in cloud environments: a survey*, *Inf. Secur.* 13 (2014) 2.
- [17] E. Brickell, J. Camenisch, L. Chen, *Direct anonymous attestation*, in: ACM International Conference on Computer and Communications Security (CCS), 2004.
- [18] M. Barbosa, B. Portela, G. Scerri, B. Warinschi, *Foundations of hardware-based attested computation and application to SGX EuroS & P*, 2016.
- [19] J. Katz, *Universally composable multi-party computation using tamper-proof hardware*, in: International Conference on the Theory and Applications of Cryptographic Techniques, 2007.
- [20] Amiri Sani, A. SchrodinText, *Strong protection of sensitive textual content of mobile applications*, in: ACM International Conference on Mobile Systems, Applications and Services (MobiSys), 2017.
- [21] M. Lentz, R. Sen, P. Druschel, B. Bhattacharjee, *SeCloak. ARM trustzone-based mobile peripheral control*, in: ACM International Conference on Mobile Systems, Applications and Services, MobiSys, 2018.
- [22] Adi Shamir, *How to share a secret*, *Commun. ACM* 22 (11) (1979).
- [23] A.S. Ibrahim, J. Hamlyn-Harris, J. Grundy, *Emerging security challenges of cloud virtual infrastructure*, arXiv preprint, arXiv:1612-09059, 2016.
- [24] J.M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, A. Perrig, *TrustVisor: Efficient TCB reduction and attestation*. S & P, 2010.
- [25] ARM. *Security Technology-Building a Secure System using TrustZone Technology*. ARM Technical White Paper, 2009.
- [26] V. Costan, S. Devadas, *Intel SGX Explained*. IACR Cryptology ePrint Archive, 2016.
- [27] M. Sabt, M. Achemlal, A. Bouabdallah, *Trusted execution environment: What it is, and what it is not*, in: Trustcom/BigDataSE/ISPA, 2015.
- [28] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A.R. Sadeghi, G. Scerri, B. Warinschi, *Secure Multiparty Computation from SGX IACR Cryptology ePrint Ar*. 2016.
- [29] N. AnCIAUX, S. Lallali, I.S. Popa, P. Pucheral, *A scalable search engine for mass storage smart objects*, in: PVLDB, 2015.
- [30] P. Tran-Van, N. AnCIAUX, P. Pucheral, *SWYSWYK: a privacy-by-design paradigm for personal information management systems*, in: Information Systems Development ISD, 2017.
- [31] W. Lehner, *The data center under your desk: how disruptive is modern hardware for DB system design?*, *Keynote at PVLDB 10 (2017) 12*.
- [32] T. Chajed, J. Gjengset, J. Van Den Hooff, M.F. Kaashoek, J. Mickens, R. Morris, N. Zeldovich, *Amber: Decoupling user data from web applications*, in: 15th Workshop on Hot Topics in Operating Systems (HotOS XV), 2015.
- [33] S. Palkar, M. Zaharia, *DIY hosting for online privacy*, in: ACM Workshop on Hot Topics in Networks, 2017.
- [34] S. Lee, E.L. Wong, D. Goel, M. Dahlin, V. Shmatikov, *Pibox: A platform for privacy-preserving apps*, in: NSDI, 2013.
- [35] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, M. Russinovich, *VC3: trustworthy data analytics in the cloud using SGX*. S & P, 2015.
- [36] P. Tran-Van, N. AnCIAUX, P. Pucheral, *A new sharing paradigm for the personal cloud*, in: Trustbus, 2017.
- [37] H. Almuhamidi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L.F. Cranor, Y. Agarwal, *Your Location has been Shared 5.398 Times!: A Field Study on Mobile App Privacy Nudging*. CHI, 2015.
- [38] H. Harkous, R. Rahman, B. Karlas, K. Aberer, *The Curious Case of the PDF Converter that Likes Mozart: Dissecting and Mitigating the Privacy Risk of Personal Cloud Apps*. arXiv preprint, 2016.

- [39] J. Saia, M. Zamani, Recent results in scalable multi-party computation, in: Int. Conf. on Current Trends in Theory and Practice of Informatics, 2015.
- [40] G. Alonso, Data Processing in Modern Hardware. Tutorial at EDBT, 2016.
- [41] S. Bajaj, R. Sion, [TrustedDB: A trusted hardware-based database with privacy and data confidentiality](#), *Trans. Knowl. Data Eng.* 26 (2014) 3.
- [42] A. Arasu, K. Eguro, M. Joglekar, R. Kaushik, D. Kossmann, R. Ramamurthy, Transaction processing on confidential data using cipherbase, in: ICDE, 2015.
- [43] C. Priebe, K. Vaswani, M. Costa, EnclaveDB: A Secure Database using SGX, in: IEEE Symposium on Security & Privacy (S & P), 2018.
- [44] Stefan Brenner, et al., SecureKeeper: Confidential ZooKeeper using Intel SGX Middleware, 2016.
- [45] ChongChong Zhao, et al., On the Performance of Intel SGX IEEE Web Inf. Systems and Applications Conf. 2016.
- [46] M. Bilal, Towards Secure Stream Processing Using Intel SGX. Report Univ. Louvain, 2017.

Appendix E

Supporting secure keyword search in the personal cloud



Supporting secure keyword search in the personal cloud



Saliha Lallali^a, Nicolas Anciaux^{a,b,*}, Iulian Sandu Popa^{b,a}, Philippe Pucheral^{b,a}

^aINRIA Saclay-Ile-de-France, Université Paris-Saclay, 1 Rue H. d'Estienne d'Orves, 91120 Palaiseau, France

^bDAVID Lab, University of Versailles Saint-Quentin-en-Yvelines, Université Paris-Saclay, 45 avenue des Etats-Unis, 78035, Versailles Cedex, France

ARTICLE INFO

Article history:

Received 31 August 2016
Revised 5 May 2017
Accepted 27 September 2017
Available online 28 September 2017

Keywords:

Embedded search engine
Indexing techniques
Conditional top-*k* queries
Flash memory
Secure token
Secure personal cloud
Smart object

ABSTRACT

The Personal Cloud paradigm has emerged as a solution that allows individuals to manage under their control the collection, usage and sharing of their data. However, by regaining the full control over their data, the users also inherit the burden of protecting it against all forms of attacks and abusive usages. The Secure Personal Cloud architecture relieves the individual from this security task by employing a secure token (i.e., a tamper-resistant hardware device) to control all the sensitive information (e.g., encryption keys, metadata, indexes) and operations (e.g., authentication, data encryption/decryption, access control, and query processing). However, secure tokens are usually equipped with extremely low RAM but have significant Flash storage capacity (Gigabytes), which raises important barriers for embedded data management. This paper presents a new embedded search engine specifically designed for secure tokens, which applies to the important use-case of managing and securing documents in the Personal Cloud context. Conventional search engines privilege either insertion or query scalability but cannot meet both requirements at the same time. Moreover, very few solutions support data deletions and updates in this context. In this paper, we introduce three design principles, namely Write-Once Partitioning, Linear Pipelining and Background Linear Merging, and show how they can be combined to produce an embedded search engine matching the hardware constraints of secure tokens and reconciling high insert/delete/update rate and query scalability. Our experimental results, obtained with a prototype running on a representative hardware platform, demonstrate the scalability of the approach on large datasets and its superiority compared to state of the art methods. Finally, we also discuss the integration of our solution in another important real use-case related to performing information retrieval in smart objects.

© 2017 Published by Elsevier Ltd.

1. Introduction

We are witnessing an exponential accumulation of personal data on central servers: data automatically gathered by administrations, companies and web sites but also data produced by the individuals themselves and deliberately stored in the cloud for convenience (e.g., photos, agendas, raw data produced by smart appliances and quantified-self devices). Unfortunately, there are many examples of privacy violations arising from abusive use or attacks, and even the most secured servers are not spared.

The Personal Cloud paradigm has recently emerged as a way to allow individuals to manage under their control the collection, usage and sharing of their data, as requested by the World Economic Forum.¹ Initiatives like Blue Button and Green Button in the US, MiData in Great Britain and MesInfos in France bring about

this paradigm by returning personal data retained by companies and administrations to individuals. Personal cloud platforms also arise in the market place (e.g., Cozy Cloud, Own Cloud, SeaFile and Younity to cite² only a few). This user-centric vision illustrates the gravity shift of information management from organizations to individuals. However, at the time individuals recover their sovereignty of their data, they also inherit the burden of organizing this personal data space and more importantly of protecting it against all forms of attacks and abusive usages, a responsibility that they cannot endorse.

The Secure Personal Cloud paradigm that we proposed in [7] relieves the individual from this security task. The corresponding architecture, illustrated in Fig. 1, combines a traditional personal cloud server (e.g., running on a plug computer or an internet gateway at home) and a *secure token* (i.e., a tamper-resistant hardware device). Heterogeneous data issued by external sources and by personal appliances are all transformed into documents.

* Corresponding author at: INRIA Saclay-Ile-de-France, Université Paris-Saclay, 1 Rue H. d'Estienne d'Orves, 91120 Palaiseau, France.

E-mail address: nicolas.anciaux@inria.fr (N. Anciaux).

¹ The World Economic Forum. Rethinking Personal Data: Strengthening Trust. May 2012.

² OwnCloud: <https://owncloud.org/>; CozyCloud: <https://www.cozycloud.cc/>; SeaFile: <http://www.seefile.com/>; Younity: <http://getyounity.com/>.

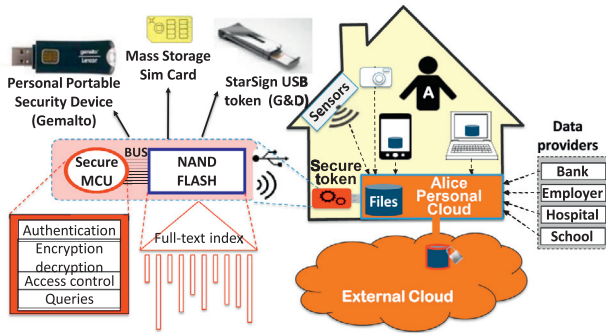


Fig. 1. Example of a secure personal cloud platform.

Document metadata (keywords extracted from the file content, date, type, authors, tags set by the user herself, etc.) is extracted at insertion time, stored in the secure token and indexed so that the secure token can act as a privacy preserving Google Desktop or Spotlight for the user's dataspace. Documents themselves are encrypted by the secure token before being stored in the Personal Cloud, locally or remotely. Thus, the secure token plays the role of a gatekeeper for the whole Personal Cloud by managing all the sensitive information (e.g., encryption keys, metadata, indexes) and operations (e.g., authentication, data encryption/decryption, access control, and query processing) [22,23].

In the context of the Personal Cloud, embedding a full-text search engine in a secure token will allow a user to securely search through her file collection in a simple way (i.e., using keywords), without exposing any metadata to the outside world. A file can be any form of document, mail, picture, music or video file, etc., that is associated with a set of terms. A query can be any form of keyword search using a ranking function (e.g., *tf-idf*) identifying the top-*k* most relevant files. Designing such embedded search engine is however very challenging. Indeed, data storage in secure tokens is usually provided by large capacity removable SD or μ SD cards or by soldered raw Flash chips while computing power is provided by microcontrollers (MCU) equipped with tiny RAM (tens of KB). This conjunction of hardware constraints raises critical issues deeply discussed in [5]. Typically, NAND Flash badly adapts to random fine-grain updates while state-of-the-art indexing techniques either consume a lot of RAM or produce a large quantity of random fine-grain updates.

We propose in this paper an efficient and scalable search engine adapted to the highly constrained architecture of secure tokens. Specifically, we make the following contributions³:

- We define two mandatory properties, namely *Bounded RAM agreement* (capturing the hardware constraints) and *Full scalability* (capturing the performance requirements) and then introduce three design principles, namely *Write-Once Partitioning*, *Linear Pipelining* and *Background Linear Merging*, to devise an inverted index complying with these properties.

³ This paper is an extended version of [6]. The new material covers three significant contributions. First, we provide all algorithms underlying our search engine, some of them being non trivial, and perform a thorough analysis of their RAM consumption to demonstrate the compliance of our design with the Bounded RAM agreement. Second, we extended the performance measurements performed in [6] with a third real dataset (ENRON), representative of the personal Cloud context and more generally of any context manipulating rich documents with random updates. We also extended the type of situations measured in order to demonstrate the compliance of our design with the Full scalability property. Taken together, Sections 8 and 9 validate the approach in the most comprehensive and complementary way. Third, Section 7 proposes an extension of our work to support *conditional top-k* queries in the personal Cloud context. Finally, note that an operational prototype of the complete design has been developed and different facets of this prototype have been demonstrated [22,23].

- Based on these principles, we propose a novel inverted index structure and related algorithms to support all the basic index operations, i.e., search, insertion, deletion and update.
- We show that our solution can be extended towards a *conditional top-k* query engine to support flexible and secure searches in a Personal Cloud context.
- Finally, we validate our design in two complementary ways. First, we do a precise analysis of the RAM consumption of each algorithm and demonstrate that each satisfies the Bounded RAM agreement. Second, we conducted a comprehensive set of experiments on a real representative secure token platform, using three real and synthetic datasets, and show that query and insertion/deletion/update performance can be met together demonstrating Full scalability compliance.

It is worth noticing that the hardware architecture of secure tokens is very similar to the hardware architecture of smart objects, which also consists in an MCU connected to a NAND Flash storage. Given the capacity of smart objects to acquire, store and process data, new services have emerged. Camera sensors tag photographs and provide search capabilities to retrieve them [40]. Smart objects maintain the description of their surrounding [41], e.g., shops like bookstores can be queried directly by customers in search of a certain product enabling the Internet of Things [1]. Smart meters record energy consumption events and GPS devices track locations and moves. This explains the growing interest for transposing traditional data management functionalities directly into smart objects. The embedded search engine proposed in this paper can be equally employed in the context of smart objects to search relevant objects in their surroundings based on their description, search pictures by using tags or perform analytic tasks (i.e., top-*k* queries) over sets of events (i.e., terms) captured during time windows (i.e., files).

The rest of the paper is organized as follows. Section 2 details the search engine requirements, the secure tokens' hardware constraints, analyses the state-of-the-art solutions and derives from this analysis a precise problem statement. Section 3 introduces our three design principles, while Sections 4 and 5 detail the proposed inverted index structure and algorithms derived from these principles. Section 6 is devoted to the tricky case of file deletions and updates. Section 7 introduces the need for conditional top-*k* queries and discusses the extension of our search engine to match such a requirement. Then, we validate our design with respect to the Bounded RAM agreement (Section 8) and Full scalability (Section 9) properties. Finally, Section 10 concludes. At a more general level, the paper is organized in four parts: (i) the problem presentation (Sections 1 and 2), (ii) the proposed solution (Sections 3, 4, 5 and 6), (iii) the extension to conditional top-*k* queries (Section 7), and (iv) the evaluation (Sections 8 and 9).

2. Problem statement

This section describes the main requirements of a full-text search engine, the hardware constraints of secure tokens, and reviews the literature addressing the problem of implementing a search engine under these constraints. Then, in the light of the existing works and their shortcomings, we precisely state the problem addressed in this paper.

2.1. Search engine requirements

As in [33], we consider that the search engine of interest in this paper has similar functionality as a Google Desktop embedded in secure tokens. Hence, we use the terminology introduced in the Information Retrieval literature for full-text search. Then, a document refers to any form of data files, terms refers to any

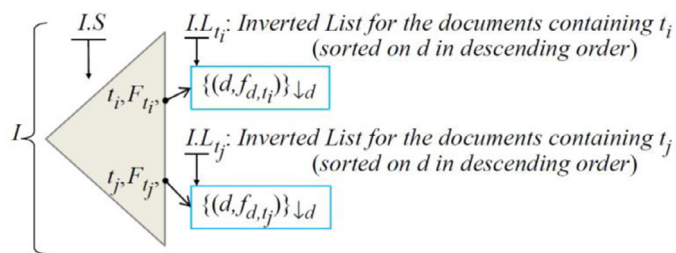


Fig. 2. Typical inverted index structure.

forms of metadata elements, term frequencies refer to metadata element weights and a query is equivalent to a full-text search.

Full-text search has been widely studied by the information retrieval community since decades (see [42] for a recent survey). The core problem is, given a collection of documents and a user query expressed as a set of terms $\{t_i\}$, to retrieve the k most relevant documents according to a ranking function. In the wide majority of the related works, the *tf-idf* score, i.e., term frequency-inverse document frequency, is used to rank the query results. A document can be of many types (e.g., text file, image, etc.) and is associated with a set of terms (or keywords) describing its content and weights indicating their respective importance in the document. For text documents, the terms are words composing the document and their weight is their frequency in the document. For images, the terms can be tags, metadata or visterms describing image subparts [40]. For a query $Q=\{t\}$, the *tf-idf* score of each indexed document d containing at least a query term can be computed as follows:

$$tf - idf(d) = \sum_{t \in Q} \log(f_{d,t} + 1) \cdot \log\left(\frac{N}{F_t}\right)$$

where $f_{d,t}$ is the frequency of term t in document d , N is the total number of indexed documents, and F_t is the number of documents that contain t . This formula is given for illustrative purpose, the weight between $f_{d,t}$ and N/F_t varying depending on the proposals.

Classically, full-text search queries are evaluated efficiently using an inverted index, named I hereafter (see Fig. 2). Given $D=\{d_i\}$ a set of documents, the inverted index I over D consists of two main components [42]: (i) a search structure $I.S$ (also called dictionary) which stores for each term t appearing in the documents the number F_t of documents containing t and a pointer to the inverted list of t ; (ii) a set of inverted lists $\{LL_t\}$ where each list stores for a term t the list of $(d, f_{d,t})$ pairs where d is a document identifier in D that contains t and $f_{d,t}$ is the weight of the term t in the document d (typically the frequency of t in d). The dictionary is constituted by all the distinct terms t of the documents in D , and is large in practice, which requires organizing it into a search-efficient structure such as a B-tree.

A query $Q=\{t\}$ is traditionally evaluated by: (i) accessing $I.S$ to retrieve for each query term t the inverted lists elements $\{LL_t\}_{t \in Q}$; (ii) allocating in RAM one container for each unique document identifier in these lists; (iii) computing the score of each of these documents using a weight function, e.g., *tf-idf*; (iv) ranking the documents according to their score and producing the k documents with the highest scores.

2.2. Secure Tokens' hardware constraints

Whatever their form factor and usage, secure tokens share strong commonalities in terms of data management architecture. Indeed, a large NAND Flash storage is used to persistently store the data and the indexes, and a microcontroller (MCU) executes the embedded code, both being connected by a bus. Hence, the

architecture inherits hardware constraints from both the MCU and the Flash memory.

The MCUs embedded in secure tokens usually have a low power CPU, a tiny RAM (few KB), and a few MB of persistent memory (ROM, NOR or EEPROM) used to store the embedded code. The NAND Flash component (either raw NAND Flash chip or SD/micro-SD card) also exhibits strong limitations. In NAND Flash, the base unit for a read and a write operation is the sector (usually 512 bytes) with raw NAND Flash chips or the page (usually 2 Kbytes or four sectors) with SD/micro-SD cards in which the access to the Flash memory is managed by a Flash Translation Layer (FTL). The sectors/pages must be erased before being rewritten but the erase operation must be performed at a block granularity (e.g., 256 pages). Erases are then costly and a block wears out after about 10^4 repeated write/erase cycles. In addition, the sectors/pages have to be written sequentially in a block. Therefore, NAND Flash badly supports random writes. We observed this same bad behavior both with raw NAND Flash chips and SD/micro-SD cards. Our own measurements shown in Table 1 (see Section 9.1) corroborate the ones published in [12] indicating that random writes are (much) more costly than sequential writes on SD cards. While high-end SSDs use large on-device RAM (e.g., 512Mbytes) to reorder random writes and optimize their performance, secure tokens equipped with very scarce RAM and basic NAND Flash storage cannot hide NAND Flash constraints and as such are exposed to large performance degradation. Hence, in the embedded context, random writes in Flash storage must be proscribed.

We also mention that in case of accessing the Flash through an FTL (e.g., with an SD card), the access can be done at a granularity smaller than the base unit (e.g., reading/writing at a sector granularity instead of a page granularity) with the inconvenient of a reduced read/write throughput of the device. However, the sequential/random write ratio remains practically the same with the ratio at the base granularity (see Table 1). The advantage in this case is the reduction of the amount of RAM memory required to process the data read from or written to the Flash storage, which is a major benefit in the context of secure tokens given their tiny RAM memory. In this work, we employ this approach since our secure tokens use a micro-SD card storage, but we prefer to access it at a sector granularity (i.e., 512 bytes) to reduce the RAM consumption of the proposed method. For simplicity, use the terms *page* and *sector* interchangeably in the rest of the paper to denote a data unit of 512 bytes in Flash or in RAM.

Finally, it is worth observing that given the strong similarity between the hardware architecture of secure tokens and of smart objects, we can consider that the secure tokens represent a specific instance of smart objects, i.e., smart objects having a tamper-resistant MCU. Hence, in this paper we use the terms *secure token* and *smart object* according to this observation.

2.3. State-of-the-Art solutions

Data management embedded in secure tokens [4,34] or more generally in smart objects is no longer a new topic. Many proposals from the database community tackle this problem in the context of the Internet of Things [11], strengthening the idea that smart objects must now be considered as first-class data sources. For instance, simple query evaluation facilities have been recently proposed for sensor nodes equipped with large Flash memory [14,16] to enable filtering operations. Relational database operations like selection, projection and join for new generations of SIM cards with large Flash storage capacities have been proposed in [5,35]. More complex treatments such as facial recognition and the related indexing techniques have been investigated also [15]. However, several works [5,25,35] consider a traditional database context and do not address the full-text search problem, leading

Table 1
Measured performance of common SD cards.

Throughput (KB/s) with sector / page granularity	Read (512B / 2KB)	Sequential Write (512B / 2KB)	Random Write (512B / 2KB)
Kingston μ SDHC	385 / 1053	79 / 270	3.1 / 11
Lexar SDMI4GB-715	625 / 1667	238 / 833	2.8 / 10
Samsung μ SDHC Plus	385 / 1111	172 / 625	1.6 / 6.1
SiliconPower SDHC	357 / 909	147 / 526	12.5 / 36.4
SanDisk μ SDHC	417 / 1176	357 / 1111	14.3 / 30.8

to different query processing techniques and indexing structures. Therefore, we focus below on works specifically addressing embedded search engines and then extend the review to a few works related to Flash-based indexing when the way they tackle the MCU and Flash constraints can enlighten the discussion.

2.3.1. Embedded search engines

A few pioneer works recently demonstrate the interest of embedding search engine techniques into smart objects equipped with extended Flash storage to manage collections of files stored locally [32,33,37,38,40]. These works rely on a similar design of the embedded inverted index as proposed in [33]. Instead of maintaining one inverted list per term in the dictionary, each term is hashed to a bucket and a single inverted list is built for each bucket. The inverted lists are stored sequentially in Flash memory, within chained pages, and only a small hash table referencing the first Flash page of each bucket is kept in RAM. The number of buckets is kept small, such that (i) the large dictionary of terms (usually tens of MB) is replaced by a small hash table stored in RAM, and (ii) the main part of the RAM can be used as an insertion buffer for the inverted lists elements, i.e., $(t, d, f_{d,t})$ triples. This approach complies with a small RAM and suits well the Flash constraints by precluding fine grain random (re)writes in Flash. However, each inverted lists corresponds to a large number of different terms, which unavoidably leads to a high query evaluation cost that increases proportionally with the size of the data collection. The less RAM available, the smaller the number of hash buckets and the more severe the problem is. In addition, these techniques do not support document deletions, but only data aging mechanisms, where old index entries automatically expire when overwritten by new ones. A similar design is proposed in [40] that builds a distributed search engine to retrieve images captured by camera sensors. A local inverted index is embedded in each sensor node to retrieve the relevant images locally, before conducting the distributed search. However, this work considers powerful sensors nodes (with tens of MB of local RAM) equipped with custom SD card boards (with specific performances). At the same time, the underlying index structure is based on inverted lists organized in a similar way as in [33]. All these methods are highly efficient for document insertions, but fail to provide scalable query processing for large collections of documents. Therefore, their usage is limited to applications that require storing only a small number (few hundreds) of documents.

2.3.2. Key-value pair indexing in NAND flash

In the key-value store context, SkimpyStash [13], LogBase [36], SILT [26] and Hyder [10] propose Flash-aware structures to store and query key-value pairs. Hyder [10] proposes a multiversion key-value database stored in Flash and shared over the network. It makes use of a single binary balanced tree index in Flash to find any version of any tuple corresponding to a given key. The binary tree is not updated in place, the path from the inserted or updated node being rewritten up to the root. Unfortunately, this technique cannot be used to implement full-text searches where each term may appear in many documents (i.e., binary trees are not adequate to index non-unique keys). SkimpyStash [13], LogBase [36] and SILT [26] organize key-value pairs in a log structure to exploit

sequential writes, but require some form of in-memory (RAM) indexing with a size proportional to the database size. Thus, the memory consumption may easily exceed the RAM size of an MCU (i.e., these methods require at least 1B per indexed record).

2.3.3. B-tree indexing in NAND flash

In the database context, adapting the B-tree to NAND Flash has received a great attention. Indeed, the B-tree is a very popular index and its standard implementation performs poorly in Flash [39]. Many recent proposals [2,24,39] tackle this problem. The key idea in these approaches is to buffer the updates in log structures that are written sequentially and to leverage the fast (random) read performance of Flash memory to compensate the loss of optimality of the lookups. When the log is large enough, the updates are committed into the B-tree in a batch mode, to amortize the Flash write cost. The log must be indexed in RAM to ensure performance. The different proposals vary in the way the log and the in-memory index are managed, and in the impact it has on the commit frequency. To amortize the write cost by a significant factor, the log must be seldom committed, which requires more RAM. Conversely, limiting the RAM size leads to increasing the commit frequency, thus generating more random writes. The RAM consumption and the random write cost are thus conflicting parameters. Under severe RAM limitations, the gain on random writes definitely vanishes.

2.3.4. Partitioned indexes

In another line of work, partitioned indexes have been extensively employed especially to improve the storage performance in environments with insert-intensive workloads and concurrent queries on magnetic disks. A prominent example is the LSM-tree (i.e., the Log-Structured Merge-tree) [30] and its many variants (e.g., the Stepped Merge Method [17], the Y-tree [18], the Partitioned Exponential file [19], and the bLSM-tree [31] to name but a few). The LSM-tree consists in one in-memory B-tree component to buffer the updates and one on-disk B⁺-tree component that indexes the disk resident data. Periodically, the two components are merged to integrate the in-memory data and free the memory. The benefit of such an approach is twofold. First the updates are integrated in batch, which amortizes the write cost per update. Second, the merge operation uses sequential I/Os, which reduces the disk arm movements and thus, highly increases the throughput. If the indexed dataset becomes too large, the index disk component can be divided into several disk components of exponentially increasing size to reduce the write amplification of merges. Many works have proposed optimized versions of the LSM-tree. For instance, bLSM [31] fixes several limitations of the LSM-tree. Among the improvements, the main contribution is an advanced merge scheduler that bounds the index write latency without impacting its throughput. Also, the FD-tree [24] proposes a similar structure with the LSM-tree to optimize the data indexing on SSDs. Furthermore, the storage systems of the major web service provider, e.g., Google's Bigtable and Facebook's Cassandra, employ a similar partitioning approach to implement key-value stores. The idea is to buffer large amounts of updates in RAM and then flush them in block on disk as a new partition. Periodically, the small partitions are merged into a large partition.

The proposed search engine shares the general idea of index partitioning and merging with the above mentioned works. However, the similarity stops at the general level since the specific hardware constrains and type of queries in our context cannot be satisfied by the existing solutions. In particular, the small amount of RAM requires frequent flushes of the buffered updates. This leads to a specific organization of the partitions and merge scheduling in our structure. The type of query in our context (i.e., top-k keyword search) represents another major difference with the existing partitioning methods that only consider the classical key-value search. To be able to evaluate full text search queries in the presence of deletions and limited amount of RAM, our search engine proposes a novel index organization with a specific query processing.

In general, designing access methods is often a matter of tradeoff between minimizing read times, update cost and memory/storage overhead as recently observed in [8]. Given the specific architecture of secure tokens, this tradeoff translates to a tension between memory and Flash storage in the embedded context [3]. Specifically, tiny RAM and NAND Flash persistent storage introduce conflicting constraints and lead to split state of the art solutions in two families. The *insert-optimized family* reaches insertion scalability thanks to a small indexed structure buffered in RAM and sequentially flushed in Flash, thereby precluding costly random writes in Flash. This good insertion behavior is however obtained to the detriment of query scalability, the performance of searches being roughly linear with the index size in Flash. Conversely, the *query-optimized family* reaches query scalability by adapting traditional indexing structures to Flash storage, to the detriment of insertion scalability, the number of random (re)writes in Flash (linked to the log commit frequency) being roughly inversely proportional to the RAM capacity. In addition, we are not aware of works addressing the crucial problem of random document deletions in the context of an embedded search engine.

2.4. Problem formulation

In the light of the preceding sections, the problem addressed in this paper can be formulated as *designing an embedded full-text search engine that has the following two properties*:

- **Bounded RAM agreement:** the proposed engine must be able to respect a predefined RAM consumption bound (RAM_Bound), precluding any solution where this consumption depends on the size of the document set.
- **Full scalability:** the proposed engine must be scalable for queries and updates (insertion, deletion of documents) without distinction.

The Bounded RAM agreement is required to comply with the widest population of secure tokens. The consequence is that the full-text search engine must remain functional even when very little RAM (a few KB) is made available to it. Note that the RAM_Bound size is a subpart of the total physical RAM capacity of a secure token considering that the RAM resource is shared by all software components running in parallel on the platform, including the operating system. The RAM_Bound property is also mandatory in a co-design perspective where the hardware resources of a given platform must be precisely calibrated to match the requirements of a particular application domain.

The Full scalability property guarantees the generality of the approach. By avoiding to privilege a particular workload, the index can comply with most applications and data sets. To achieve update scalability, the index maintenance needs to be processed without generating random writes, which are badly supported by the Flash memory. At the same time, achieving query scalability

means obtaining query execution costs in the same order of magnitude with the ideal query costs provided by a classical inverted index I .

3. Design principles

Satisfying the Bounded RAM agreement and Full scalability properties simultaneously is challenging, considering the conflicting MCU and Flash constraints mentioned above. To tackle this challenge, we propose in this paper an indexing method that relies on the following three design principles.

P1. Write-Once Partitioning: *Split the inverted index structure I in successive partitions such that a partition is flushed only once in Flash and is never updated.*

By precluding random writes in Flash, Write-Once Partitioning aims at satisfying update scalability. Considering the Bounded RAM agreement, the consequence of this principle is to parse documents and maintain I in a streaming way. Conceptually, each partition can be seen as the result of indexing a window of the document input flow, the size of which is limited by the RAM_Bound. Therefore, I is split in an infinite sequence of partitions $\langle I_0, I_1, \dots, I_p \rangle$, each partition I_i having the same internal structure as I . When the size of the current I_i partition stored in RAM reaches RAM_Bound, I_i is flushed in Flash and a new partition I_{i+1} is initialized in RAM for the next window.

A second consequence of this design principle is that document deletions have to be processed similar to document insertions since the partitions cannot be modified once they are written. This means adding compensating information in each partition that will be considered by the query process to produce correct results.

P2. Linear Pipelining: *Compute each query Q with respect to the Bounded RAM agreement in such a way that the execution cost of Q over $\langle I_0, I_1, \dots, I_p \rangle$ is in the same order of magnitude as the execution cost of Q over I .*

Linear Pipelining aims at satisfying query scalability under the Bounded RAM agreement. A unique structure I as the one pictured in Fig. 2 is assumed to satisfy query scalability by nature and is considered hereafter as providing a lower bound in terms of query execution time. Hence, the objective of Linear pipelining is to keep the performance gap between Q over $\langle I_0, I_1, \dots, I_p \rangle$ and Q over I , both small and predictable (bounded by a given tuning parameter). Computing Q as a set-oriented composition of a set of Q_i over I_i , (with $i=0, \dots, p$) would unavoidably violate the Bounded RAM agreement as p increases, since it will require to store all Q_i 's intermediate results in RAM. Hence the necessity to organize the processing in pipeline such that the RAM consumption remains independent of p , and therefore of the number of indexed documents. Also, the term *linear* pipelining conveys the idea that the query processing must preclude any iteration (i.e., repeated accesses) over the same data structure to reach the expected level of performance. This disqualifies brute-force pipeline solutions where the *tf-idf* scores of documents are computed one after the other, at the price of reading the same inverted lists as many times as the number of documents they contain.

However, Linear Pipelining alone cannot prevent the performance gap between Q over $\langle I_0, I_1, \dots, I_p \rangle$ and Q over I to increase with the increase of p as (i) multiple searches in several small I_p 's are more costly than a single search in a large I and (ii) the inverted lists in $\langle I_0, I_1, \dots, I_p \rangle$ are likely to occupy only fractions of Flash pages, multiplying the number of Flash I/Os to access the same amount of data. A third design principle is then required.

P3. Background Linear Merging: *To limit the total number of partitions, periodically merge partitions in a way compliant with the Bounded RAM agreement and without hurting update scalability.*

The objective of partition merging is therefore to obtain a lower number of larger partitions to avoid the drawbacks mentioned

above. Partition merging must meet three requirements. First the merge must be performed in pipeline to comply with the Bounded RAM agreement. Second, since its cost can be significant (i.e., proportional to the total size of the merged partitions), the merge must be processed in background to avoid locking the index structure for unbounded periods of time. Since multi-threading is not supported by the targeted platforms, background processing can simply be understood as the capacity to interrupt and recover the merging process at any time. Third, update scalability requires that the total cost of a merge run be always smaller than the time to fill out the next bunch of partitions to be merged.

Taken together, principles P1 to P3 reconcile the Bounded RAM agreement and Full scalability index properties. The technical solutions to implement these three principles are presented in the next sections. To ease the presentation, we introduce first the foundation of our solution considering only document insertions and queries. The trickier case of document deletions is postponed to Section 6.

4. Write-once partitioning and linear pipelining

These two design principles are discussed together because the complexity comes from their combination. Indeed, Write-Once Partitioning is straightforward on its own. It simply consists in splitting I in a sequence $\langle I_0, I_1, \dots, I_p \rangle$ of small indexes called partitions, each one having a size bounded by RAM_Bound . The difficulty is to implement a linear pipeline execution of any query Q on this sequence of partial indexes.

Executing Q over I would lead to evaluate:

$$\text{Top}_k \left[\sum_{t \in Q} W \left(f_{d,t}, \frac{N}{F_t} \right) \right], \text{ with } d \in D$$

where Top_k selects the k documents $d \in D$ having the largest $tf\text{-idf}$ scores, each score being computed as the sum, for all terms $t \in Q$, of a given weight function W taking as parameter the frequency $f_{d,t}$ of t in d and the inverse document frequency N/F_t . Our objective is to remain agnostic regarding W and then let the precise form of this function open. Let us now consider how each term of this expression can be evaluated by a linear pipelining process on a sequence $\langle I_0, I_1, \dots, I_p \rangle$.

Computing N . We assume that the number of documents is a global metadata maintained at insertion/deletion time and needs not be recomputed for each Q .

Computing F_t . F_t should be computed only once for each term t since F_t is constant for Q . This is why F_t is usually materialized in the dictionary part of the index ($\{t, F_t\} \subset I.S$), as shown in Fig. 2. When I is split in $\langle I_0, I_1, \dots, I_p \rangle$, the global value of F_t should be computed as the sum of the local F_t of all partitions. The complexity comes from the fact that the same document d may cross several partitions with the consequence of contributing several times to the global F_t if a simple sum is performed. The Bounded RAM agreement precludes maintaining in RAM a history of all the terms already encountered for a given document d across the parsing windows, the size of this history being unbounded. Accessing the inverted lists $\{I_i.L_t\}$ of successive partitions to check whether they intersect for a given d would also violate the Linear Pipelining principle since these same lists will be accessed again when computing the $tf\text{-idf}$ score of each document.

The solution is then to store in the dictionary of each partition the boundary of that partition, namely the identifiers of the first and last documents considered in the parsing window. Then, two bits *firstd* and *lastd* are added in the dictionary for each inverted list to register whether this list contains one (or both) of these documents, i.e., $\{t, F_t, \text{firstd}, \text{lastd}\} \subset I.S$. As illustrated in Fig. 3, this is sufficient to detect the intersection between the inverted lists of a

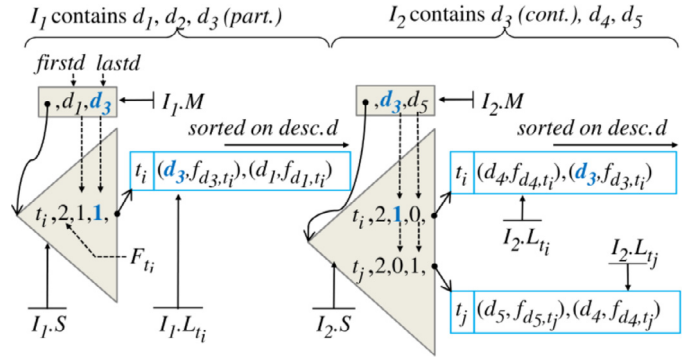


Fig. 3. Consecutive index partitions with overlapping documents.

same term t in two successive partitions. Whether an intersection between two lists is detected, the sum of their respective F_t must be decremented by 1. Hence, the correct global value of F_t can easily be computed without physically accessing the inverted lists.

During the F_t computation phase, the dictionary of each partition is read only once and the RAM consumption sums up to one buffer to read each dictionary, page by page, and one RAM variable to store the current value of each F_t .

Computing $f_{d,t}$. If a document d overlaps two consecutive partitions I_i and I_{i+1} , the inverted list L_t of a queried term $t \in Q$ may also overlap these two partitions. In this case the $f_{d,t}$ score of d is simply the sum of the (last) $f_{d,t}$ value in $I_i.L_t$ and the (first) $f_{d,t}$ value in $I_{i+1}.L_t$. To get the $f_{d,t}$ values, the inverted lists $I_i.L_t$ have to be accessed. The pointers referencing these lists are actually stored in the dictionary which has already been read while computing F_t . According to the Linear pipelining principle, we avoid reading again the dictionary by storing these pointers in RAM during the F_t computation. The extra RAM consumption is minimal and bounded by the fact that the number of partitions is itself bounded thanks to the merging process (see Section 5).

Computing Top_k . Traditionally, a RAM variable is allocated to each document d to compute its $tf\text{-idf}$ score by summing the results of $W(f_{d,t}, N/F_t)$ for all terms $t \in Q$ [42]. Then, the k best scores are selected. Unfortunately, this approach conflicts with the Bounded RAM agreement since the size of the document set is likely to be much larger than the available RAM. Hence, we organize the query processing in a pure pipeline way, allocating a RAM variable only to the k documents having currently the best scores. This forces the complete computation of $tf\text{-idf}(d)$ to be done for each d , one after the other. To meet this requirement while precluding any iteration on the inverted lists, these lists are maintained sorted on the document id. Note that if document ids reflect the insertion ordering, the inverted lists are naturally sorted. Hence, the $tf\text{-idf}$ computation sums up to a simple linear pipeline merging process of the inverted lists for all terms $t \in Q$ in each partition (see Fig. 4). The RAM consumption for this phase is therefore restricted to one variable for each of the current k best $tf\text{-idf}$ scores and to one buffer (i.e., a RAM page) per query term t to read the corresponding inverted lists $I_i.L_t$ (i.e., $I_i.L_t$ are read in parallel for all t , the inverted lists for the same t being read in sequence). Fig. 4 summarizes the data structures maintained in RAM and in Flash to handle this computation.

5. Background linear merging

The background merging process aims at achieving scalable query costs by timely merging several small indexes into a larger index structure. As mentioned in Section 3, the merge must be a pipeline process in order to comply with the Bounded RAM agreement while keeping a cost compatible with the update rate.

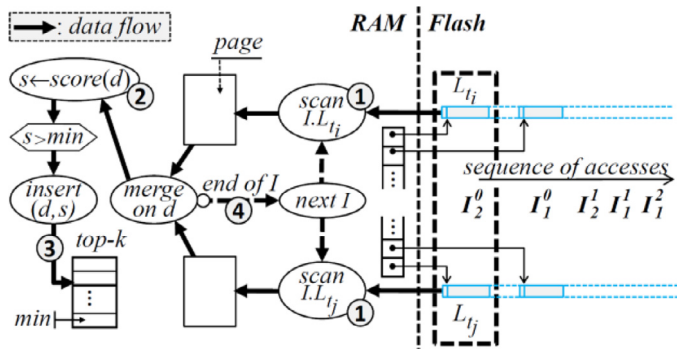


Fig. 4. Linear pipeline computation of Q over terms t_i and t_j .

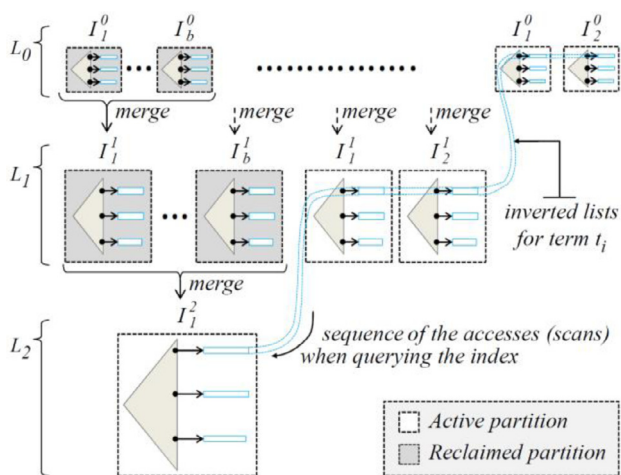


Fig. 5. The scalable and sequential flash structure.

Moreover, the query processing should continue to be executed in Linear Pipelining (see Section 4) on the structure resulting from the successive merges. Therefore, the merges have to preserve the global ordering of the document ids within the index structures.

To meet these requirements, we introduce a Sequential and Scalable Flash structure, called SSF, pictured in Fig. 5. The SSF consists in a hierarchy of partitions of exponentially increasing size. Specifically, each new index partition is flushed from RAM into the first level of the SSF, i.e., L_0 . The merge operation is triggered automatically when the number of partitions in a level becomes b , the branching factor of SSF, which is a predefined index parameter. The merge combines the b partitions at level L_i of SSF, denoted by I_1^i, \dots, I_b^i , into a new partition at level L_{i+1} , denoted by I_1^{i+1} and then reclaims all partitions at level L_i .

The merge is directly processed in pipeline as a multi-way merge of all partitions at the same level. This is possible since the dictionaries of all the partitions are already sorted on terms, while the inverted lists in each partition are also sorted on document ids. So are the dictionary and the inverted lists of the resulting partition at the upper level. More precisely, the algorithm works in two steps. In the first step, the IL part of the output partition is produced. Given b partitions in the index level L_i , $b+1$ RAM pages are necessary to process the merge in linear pipeline: b pages to merge the inverted lists in IL of all b partitions and one page to produce the output. The indexed terms are treated one after the other in alphabetic order. For each term t , the head of its inverted lists in each partition is loaded in RAM. These lists are then consumed in pipeline by a multi-way merge. Document ids are encountered in descending order in each list and the output list resulting from the merge is produced in the same order. A

particular case must be distinguished when two pairs $(d, f1_{d,t})$ and $(d, f2_{d,t})$ are encountered in separate lists for the same d ; this means that document d overlaps two partitions and these two pairs are aggregated in a single $(d, f1_{d,t} + f2_{d,t})$ before being added to IL . In the second step, the metadata IM is produced (see Fig. 3), by setting the value of $firstd$ (resp. $lastd$) with the $firstd$ (resp. $lastd$) value of the $first$ (resp. $last$) partition to be merged, and the IS structure is constructed sequentially, with an additional scan of IL . The IS tree is built from the leaves to the root. This step requires one RAM page to scan IL , plus one RAM page per IS tree level. For each list encountered in IL , a new entry $(t, F_t, presence_flags)$ is appended to the lowest level of IS ; the value F_t is obtained by summing the $f_{d,t}$ fields of all $(d, f_{d,t})$ pairs in this list; the presence flag reflects the presence in the list of the $firstd$ or $lastd$ document. Upper levels of IS are then trivially filled sequentially. This Background Merging process generates only sequential writes in Flash and previous partitions are reclaimed in large blocks after the merge. This pipeline process sequentially scans each partition only once and produces the resulting partition also sequentially. Hence, assuming $b+1$ is strictly lower than RAM_bound , one RAM buffer (of one page) can be allocated to read each partition and the merge is I/O optimal. If b is larger than RAM_bound , the algorithm remains unchanged but its I/O cost increases since each partition will be read by page fragments rather than by full pages.

Search queries can be evaluated in linear pipeline by accessing the partitions one after the other from partitions b to 1 in level 1 up to level n . In this way, the inverted lists are scanned in descending order of the document ids, from the most recently inserted document to the oldest one, and the query processing remains exactly the same as the one presented in Section 4, with the same RAM consumption. The merging and the querying processes could be organized in opposite order (i.e., in ascending order of the document ids) with no impact. However, order matters as soon as deletions are considered (see Section 6). SSF provides scalable query costs since the amount of indexed documents grows exponentially with the number of levels, while the number of partitions increases only linearly with the number of levels.

Note that merges in the upper levels are exponentially rare (one merge in level L_i for b^i merges in L_0) but also exponentially costly. To mitigate this problem, we perform the merge operations in background (i.e., in a non-blocking manner). Since the merge may consume up to b pages of RAM, we launch/resume it each time after a new partition is flushed in L_0 of the SSF, the RAM being empty at this time. A small quantum of time (a few hundred milliseconds in practice) is allocated to the merging process. Each time this quantum expires, the merge is interrupted and its execution status (i.e., a cursor indicating the current Flash page position in each partition) is memorized. The quantum of time is chosen so that the merge of a given SSF level ends before the next merge of the same level need to be triggered. In this way, the cost of a merge operation is spread among the flush operations and remains almost transparent. This basic strategy is simple and does not make any assumption regarding the index workload. However, it could be improved in certain contexts, by taking advantage of the idle time of the platform.

6. Document deletions

To the best of our knowledge, our proposal is the first embedded search index to implement document deletions. This problem is actually of primary importance because deletions are required in many practical scenarios. Unfortunately, index updating increases significantly the complexity of the index maintenance by reintroducing the need for random updates in the index structure. In this section we extend the index structure to support the deletions of documents without generating any random write in Flash.

6.1. Solution outline

Implementing the delete operation is challenging, mainly because of the Flash memory constraints which proscribe the straightforward approach of updating in-place the inverted index. The alternative to updating in-place is *compensation*, i.e., the deleted documents' identifiers (*DDIs*) are stored in an appropriate way and used as a filter to eliminate the ghost documents retrieved by the query evaluation process.

A basic solution could be to organize the *DDIs* as a sorted list in Flash and to intersect this list at query execution time with the inverted lists in the *SSF* corresponding to the query terms. However, this solution raises several problems. First, the documents are deleted in random order, according to users' and application decisions. Hence, maintaining a sorted list of *DDIs* in Flash would violate the Write-Once Partitioning principle since the list has to be rewritten each time a set (e.g., a page) of new *DDIs* is flushed from RAM. Second, the computation of the F_t for each query term t during the first step of the query processing cannot longer be achieved without an additional merge operation to subtract the sorted list of *DDIs* from the inverted lists of the *SSF*. Third, the full *DDI* list has to be scanned for each query regardless of the query selectivity. These two last elements make the query cost dependent of the total number of deleted documents and then conflict with the Linear pipelining principle.

Therefore, instead of compensating the query evaluation process, we propose a solution based on compensating the indexing structure itself. In particular, a document deletion is treated similarly to a document insertion, i.e., by re-inserting the metadata (terms and frequencies) of all deleted documents in the *SSF*. The objective is threefold: (i) to be able to compute, as presented in Section 4, the F_t for each term t of a query based on the metadata only (of both existing and deleted documents), (ii) to have a query performance that depends on the query selectivity (i.e., number of inserted and deleted documents relevant to the query) and not on the total number of deleted documents and (iii) to effectively purge the indexing structure from the largest part of the deleted documents at Background Merging time, while remaining compliant with the Linear Pipelining principle. We present in the following the required modifications of the index structure to integrate this form of compensation.

6.2. Impact on write-once partitioning

As indicated above, a document deletion is treated similarly to a document insertion. Assuming a document d is deleted in the time window corresponding to a partition I_i , a pair $(d, -f_{d,t})$ is inserted in each list $I_i.L_t$ for the terms t present in d and the F_t value associated to t is decremented by 1 to compensate the prior insertion of that document. To distinguish between an insertion and a deletion, the frequency value $f_{d,t}$ for the deleted document id is simply stored as a negative value, i.e., $-f_{d,t}$.

6.3. Impact on linear pipelining

Executing a query Q over our compensated index structure sums up to evaluate:

$$Top_k \left[\sum_{t \in Q} W \left(|f_{d,t}|, \frac{N}{F_t} \right) \right], \text{ with } d \in (D^+ - D^-)$$

where D^+ (resp. D^-) represents the set of inserted (resp. deleted) documents.

Computing N . As presented earlier, N is a global metadata maintained at update time and then already integrates all insert and delete operations.

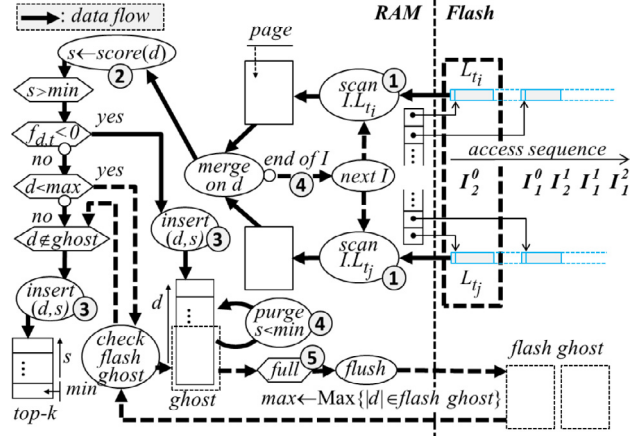


Fig. 6. Linear pipeline computation of Q in the presence of deletions.

Computing F_t . The global F_t value for a query term t is computed as usual since the local F_t values are compensated at deletion time (see above). The case of deleted documents that overlap with several consecutive partitions is equally treated as with the inserted documents.

Computing $f_{d,t}$. The $f_{d,t}$ of a document d for a term t is computed as usual, with the salient difference that a document which has been deleted appears twice: with the value $(d, f_{d,t})$ (resp. $(d, -f_{d,t})$) in the inverted lists of the partition I_i (resp. partition I_j) where it has been inserted (resp. deleted). By construction $i < j$ since a document cannot be deleted before being inserted.

Computing Top_k . Integrating deleted documents makes the computation of Top_k more subtle. Following the Linear Pipelining principle, the *tf-idf* scores of all documents are computed one after the other, in descending order of the document ids, thanks to a linear pipeline merging of the insert lists associated to the queried terms. To this end, the algorithm introduced in Section 4 uses k RAM variables to maintain the current k best *tf-idf* scores and one buffer (i.e., a RAM page) per query term t to read the corresponding inverted lists. Some elements present in the inverted lists correspond actually to deleted documents and must be filtered out. The problem comes from the fact that documents are deleted in random order. Hence, while inverted lists are sorted with respect to the insertion order of documents, a pair of the form $(d, -f_{d,t})$ may appear anywhere in the lists. In case a document d has been deleted, the unique guarantee is to encounter the pair $(d, -f_{d,t})$ before the pair $(d, f_{d,t})$ if the traversal of the lists follows a descending order of the document ids. However, maintaining in RAM the list of all encountered deleted documents in order to filter them out during the follow-up of the query processing would violate the Bounded RAM agreement.

The proposed solution works as follows. The *tf-idf* score of each document d is computed by considering the modulus of the frequencies values $|\pm f_{d,t}|$ in the *tf-idf* score computation, regardless of whether d is a deleted document or not. Two lists are maintained in RAM: $Top_k = \{(d, score(d))\}$ contains the current k best *tf-idf* scores of documents which exist with certainty (no deletion has been encountered for these documents); $Ghost = \{(d, score(d))\}$ contains the list of documents which have been deleted (a pair $(d, -f_{d,t})$ has been encountered while scanning the inverted lists) and have a score better than the smallest score in Top_k . Top_k and $Ghost$ lists are managed as follows. If the score of the current document d is worse than the smallest score in Top_k , it is simply discarded and the next document is considered (step 2 in Fig. 6). Otherwise, two cases must be distinguished. If d is a deleted document (a pair $(d, -f_{d,t})$ is encountered), then it enters the $Ghost$ list (step 3); else it enters the Top_k list unless its id is already present

in the *Ghost* list (step 4). Note that this latter case may occur only if the id of d is smaller than the largest id in *Ghost*, making the search in *Ghost* useless in many cases. An important remark is that the *Ghost* list has to register only the deleted documents which may compete with the k best documents, to filter them out when these documents are later encountered, which makes this list very small in practice.

While simple in its principle, this algorithm deserves a deeper discussion in order to evaluate its real cost. This cost actually depends on whether the *Ghost* list can entirely reside in RAM or not. Let us compute the nominal size of this list in the case where the deletions are evenly distributed among the document set. For illustration purpose, let us assume $k=10$ and the percentage of deleted documents $\delta=10\%$. Among the first 11 documents encountered during the query processing, 10 will enter the Top_k list and 1 is likely to enter the *Ghost* list. Among the next 11 documents, 1 is likely to be deleted but the probability that its score is in the 10 best scores is roughly 1/2. Among the next 11 ones, this probability falls to about 1/3 and so on and so forth. Hence, the nominal size of the *Ghost* list is:

$$\delta \cdot k \cdot \sum_{i=1}^n 1/i$$

This value can be approximated by $\delta \cdot k \cdot (\ln(n) + \varepsilon)$. For 10,000 queried documents, $n=1000$ and the size of the *Ghost* list is only $\delta \cdot k \cdot (\ln(n) + \varepsilon) \approx 10$ elements, far beyond the RAM size. In addition, the probability that the score of a *Ghost* list element competes with the Top_k ones decreases over time, giving the opportunity to continuously purge the *Ghost* list (step 5 in Fig. 6). In the very improbable case where the *Ghost* list overflows (step 6 in Fig. 6), it is sorted in descending order of the document ids, and the entries corresponding to low document ids are flushed. This situation remains however highly improbable and will concern rather unusual queries (none of the 300 queries we evaluated in our experiment produced this situation, while allocating a single RAM page for the *Ghost* list).

6.4. Impact on background pipeline merging

The main purpose of the Background Merging principle, as presented in Section 5, is to keep the query processing scalable with the indexed collection size. The introduction of deletions has actually a marginal impact on the merge operation, which continues to be efficiently processed in linear pipeline as before. Moreover, given the way the deletions are processed in our structure, i.e., by storing couples $(d, -f_{d,t})$ for the deleted documents, the merge acquires a second function which is to absorb the part of the deletions that concern the documents present in the partitions that are merged. Indeed, let us come back to the Background Merging process described in Section 5. The main difference when deletes are considered is the following. When inverted lists are merged during step 1 of the algorithm, a new particular case may occur, that is when two pairs $(d, f_{d,t})$ and $(d, -f_{d,t})$ are encountered in separate lists for the same d ; this means that document d has actually been deleted; d is then purged (the document deletion is absorbed) and will not appear in the output partition. Hence, the more frequent the Background Merging, the smaller the number of deleted entries in the index.

Taking into account the supplementary function of the merge, i.e., to absorb the data deletions, we can adjust the absorption rate of deletions by tuning the branching factor of the last index level since most of the data is stored in this index level. By setting a smaller value to the branching factor b' of the last level, the merge frequency in this level increases and consequently the absorption rate also increases. Therefore, in our implementation we use a smaller value for the branching factor of the last index level (i.e.,

$b'=3$ for the last level and $b=10$ for the other levels). Typically, about half of the total number of deletions will be absorbed for $b'=3$ if we consider that the deletions are uniformly distributed over the data insertions.

7. Towards conditional top- k queries

The core of the proposed solution has been described in the previous sections. In this section, we discuss the extension of the proposed search engine to support *conditional top- k queries*, i.e., top- k queries combined with conditions expressed over documents' metadata. Specifically, we present two major use-cases in the context of the Personal Cloud, i.e., keyword search combined with file metadata and tag-based access control. Then, we show that these two use-cases require conditional top- k queries and explain how the SSF data structure and algorithms can be extended to support it in a natural way.

7.1. The need for conditional top- k queries in the personal cloud

7.1.1. Combining keyword search and file metadata search

When a new file is added to the user's collection, both the keywords extracted from the file content (if any) and the file metadata (e.g., creation date, filename, file type and extension, tags set by the user herself, etc.) can be indexed by the search engine and then used to evaluate the score of the documents relevant for the user queries. Similar to the classical inverted index, the base implementation of our search engine (see Section 4) does not make any distinction between the terms extracted from the file content and the file metadata terms. Hence, the score of a document is computed using the classical tf-idf formula (see Section 2.1) regardless if the query terms are content keywords or metadata terms in the document.

Nonetheless, from a semantic point of view, it makes sense to separate the *content terms* from the *metadata terms* in the query evaluation like in the existing file search engine implementations (e.g., Google desktop or Spotlight). The idea is that the query terms are matched only with the content terms of the indexed documents, while users can specify additional constraints regarding the document metadata. For instance, a user can combine the query terms "research meeting Bordeaux" with the constraints "file type = pptx or (file type = mail and sender = Bob)". In this case, the query results will only consist of documents having the extension "pptx" or documents of type "mail" sent by "Bob", which contain at least one term among "research meeting Bordeaux" and ranked based on the weight of these three words in the documents. In general, the metadata search consists in one or several metadata terms that are combined by AND/OR to form a logical expression, i.e., disjunctions of conjunctions of metadata terms. Formally, a metadata search rule is a *logical expression*: $R_{metadata} = \bigvee_j (\bigwedge_i t_{i,j}^{pred})$, where each $t_{i,j}^{pred}$ is a metadata term predicate which for a given document d is evaluated true only if the metadata term $t_{i,j}$ is associated with d . For each document matching the query content terms, the logical expression on the metadata terms is evaluated and the document is considered in the query results only if the metadata expression is evaluated true.

7.1.2. Tag-based access control

The owner of a personal cloud might want to share some of her documents with other users or applications. To this end, she needs to customize the sharing of her documents by adding a personalized access control (AC) policy for each user/application accessing her personal cloud. She also wants to be sure that the personal cloud is able to securely enforce the defined policies. The latter is achieved by integrating the access control engine

within the secure token together with the search engine as depicted in Fig. 1. The definition of the AC policies depends on the employed AC model. Access control is a well-established topic in the databases field. However, traditional AC models like MAC (Mandatory Access Control), DAC (Discretionary Access Control) or R-BAC (Role-Based Access Control) are less suitable for the Personal Cloud paradigm than Tag-Based Access Control (TBAC) models [9,21,27,28,29]. Several works consider TBAC models, due to their simplicity, for non-technical computer users that require to define access control policies over their personal data. For example, [9] proposes a semantic access control for online photo albums based on descriptive tags found in social networks such as Flickr, Delicious and linked data. [29] goes in the same direction and shows that tags related to data organization can be easily used by home users to express coherent policies for access control. In [21,27,28], access control models to share files based on the TBAC model are proposed and in vivo studies involving non-specialist users show the usability of such models. A TBAC model is thus considered as a good candidate for the personal cloud context.

TBAC can be easily integrated with our inverted index if we consider that (1) an appropriate set of *access control terms* can be inserted to tag the documents at insertion, and (2) these access terms can be used at query processing time to evaluate *logical expressions* leading to discard or not each document from the query results. For illustration purpose, we define hereafter a simple TBAC model. Let D be the set documents referred by the inverted index. Each document is associated with set of terms (regular terms extracted from the documents content and access terms derived from the document content or metadata). All terms are extracted automatically at insertion time. For each document $d \in D$, let T_d be the set of access terms that was assigned to d at insertion time. Let U be set of all users/applications that can access the Personal Cloud. Any user/application has to be authenticated before gaining access to the Personal Cloud. We assume that by default everything is private and that applications are allowed to query (i.e., read) only the subset of the documents which match the access control rule defined for that application. For each user $u \in U$, the Cloud owner can define an access control rule R_u as disjunctions of conjunctions of access control terms, (i.e., $R_u = \vee_j (\wedge_i t_{i,j}^{pred})$), where each $t_{i,j}^{pred}$ is a term predicate which for a given document d is evaluated true only if the term $t_{i,j}$ appears in the access term list T_d . Therefore, at query time, the embedded access control engine evaluates for any document d in the Personal Cloud if u has the right to access d . This is realized internally by the Boolean function $Filter(u, d)$, which returns true iff R_u is true on d .

Let us consider a simple example. Bob is a friend of Alice with whom she shares the passion for country music. Bob asks Alice to share with him her country music collection. A collection rule has been settled such that any music file inserted in the personal cloud of Alice is associated with the access terms “music” and “<music_style>” where the value of music style is derived from the domain of music styles (e.g., “country”, “variety”, “jazz”, ...). Alice may define a permission rule for user Bob in her Personal Cloud: “Bob: music \wedge country”. Thus, Bob can query all the documents in the Alice’s server that contain both the access term “music” and “country”. Alice herself needs to access her email archive over the last two years from her smartphone, but does not want that the rest of her personal data space to be accessible from her smartphone. Therefore, assuming a collection rule associates any email file with the access terms “email” and “<date>”, Alice may define the rule “Alice_smartphone: (email \wedge 2014) \vee (email \wedge 2015)”.

Note that the logical expression used to define the AC policies has the same format (i.e., disjunctions of conjunctions) as the document metadata logical expression that can be used in the

scenario that combines keyword search with file metadata search describe above. Hence, these two scenarios require the same modifications to be integrated in the proposed search engine. The transformed keyword search queries are called *conditional top-k queries* in the sequel.

7.2. Search engine adaptation to conditional Top-k queries

To support conditional top-k queries, the modifications in our search engine have to be done at three levels: (i) update the top-k scoring function; (ii) adjust the insertion process and the inverted index structure; (iii) modify accordingly the query evaluation process. We detail here below these three modifications.

7.2.1. Modification of the top-k scoring function

Conditional top-k queries require to evaluate a logical expression for all the candidate documents for the query results, i.e., documents having a tf-idf score high enough to enter the top-k results. Executing a query Q in this context is equivalent to evaluate the following function:

$$Top_k \left[Filter(le, T_d) \cdot \sum_{t \in Q} W \left(f_{d,t}, \frac{N}{F_t} \right) \right], \text{ with } d \in D$$

Compared with the initial top-k scoring function defined in Section 4, the only difference is the appearance of the $Filter()$ element. $Filter$ is a function that takes as input a logical expression le (e.g., a TBAC policy or a file metadata expression) and the list of metadata terms T_d of a document. The function returns 1 if le is true on T_d and 0 otherwise. In other words, the tf-idf score of the documents whose metadata verify the logical expression remains unchanged, whereas the tf-idf score is set to 0 for the rest of the documents, which are thus eliminated from the query results.

7.2.2. Impact on the data insertion and the inverted index structure

The conditional top-k functionality requires separating semantically the content terms from the metadata terms of a document in the index structure. We use hereafter the notion of metadata term to refer to any kind of metadata (i.e., access control terms, file metadata terms or other types of metadata). To obtain this separation, we prefix all the metadata terms with a reserved tag, which will lexically distinguish the two types of terms without having any repercussion on the search structure (i.e., IS) of the inverted index in a partition. Therefore, the search of a content term is done normally, while to search for a metadata term one has to prefix the term value with the metadata tag. Finally, the inverted lists of metadata terms have the same structure (i.e., pairs $(d, f_{d,t})$) and organization (i.e., sorted on descendant value of d) as the inverted lists of content terms. The only difference is that the $f_{d,t}$ can only have two values, i.e., +1 for a document insertion and -1 for a document deletion.

7.2.3. Impact on the query evaluation

The enriched query execution is depicted in Fig. 7. The query process starts as before and considers only the content terms of the query. Whenever the score of a document d is within the k best current scores, the identifier of d is searched in the inverted lists of the metadata terms involved in the conditional expression. If $Filter$ evaluates this condition to *false*, the document d is discarded. Otherwise, d it is kept and inserted in the top_k buffer. The search within the inverted lists of the access terms *requires only one additional RAM page*. This page is used to search the occurrence of d in the inverted of each metadata term of the logical expression. To avoid reading the complete inverted list of a metadata term, the search of document d is performed using a dichotomy. Also, to evaluate the conjunctive parts of the logical

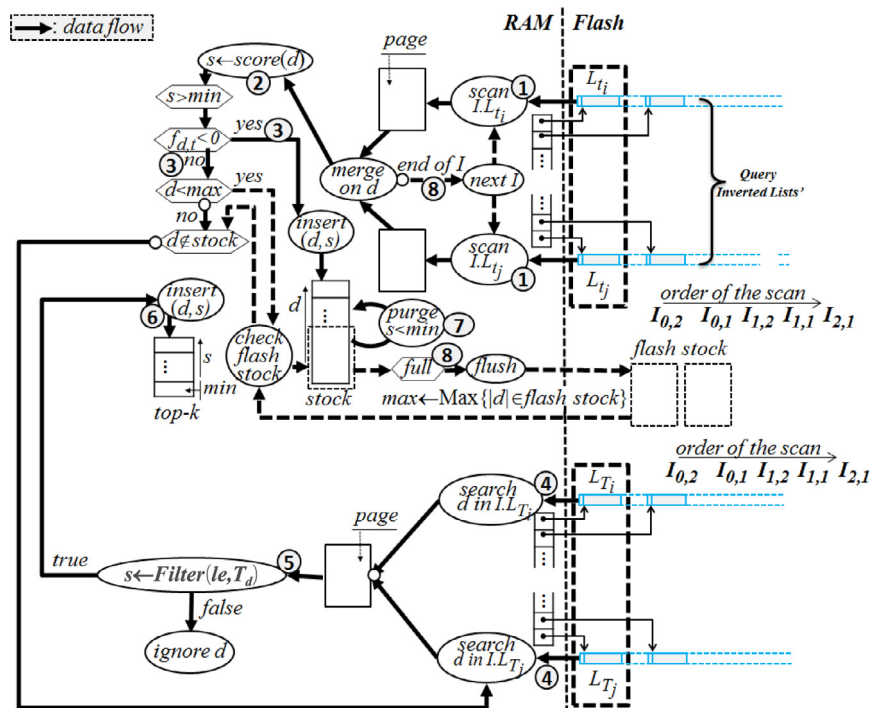


Fig. 7. Linear pipeline computation of a conditional top- k query over terms t_i and t_j and metadata terms $L_{\bar{n}}$ and $L_{\bar{j}}$.

expression, metadata terms are accessed from the less frequent term (with the smallest inverted list) to the most frequent term, to potentially stop the search as soon as d is absent from a list. The overhead in terms of memory consumption for the evaluation of the logical expression is kept minimal (i.e., one RAM page) and the execution time overhead is low since: (i) only the documents entering within the k best current scores trigger *Filter*; (ii) deleted documents can be inserted in the *Ghost* list without checking the logical expression to minimize IO cost; (iii) the search is performed using dichotomy if the inverted list is larger than one Flash page; and (iv) a subset of the access terms (and corresponding inverted lists) have to be accessed when the expression turns to be *false* (typically, for conjunctive expressions). Our experiments (see Section 9) demonstrate that the execution time overhead due to logical expression evaluation is very acceptable.

We should note that file metadata search and TBAC functionalities can be implemented together in the search engine. In this case, the metadata terms have to be grouped in two classes corresponding to the two functionalities. Then, *Filter* has to evaluate both the logical expression on the file metadata terms and the logical expression on the access control terms and return 1 only if both expressions are true.

8. Bounded RAM agreement conformance

Intuitively, the three design principles introduced in Section 3 conduct to algorithmic solutions complying by construction with the *Bounded RAM agreement* and *Full scalability* properties. This section concentrates on the RAM consumption while the performance and scalability analyses are postponed to Section 9. The primary objective of the Bounded RAM agreement is to ensure that the RAM demand does not depend on the size of the document set, thereby guaranteeing the generality of the studied solution. A rough analysis of the algorithms is sufficient to assess this behavior. However, the second objective of this agreement is to guarantee that the corresponding RAM bound is kept small enough to comply with the widest population of secure tokens. This requires a more subtle analysis of each algorithm,

the devil being often hidden in algorithmic details, especially when algorithms are non-trivial. Thus, Section 8.1 presents the detailed version of the SSF algorithms and Section 8.2 analyses their respective RAM consumption. As described in Section 2, we recall that the physical architecture of a secure token is mainly composed of a secure MCU (having a very limited amount of RAM) to execute the embedded code and a large NAND Flash storage (typically a micro-SD card) to store the data. To minimize the RAM consumption, all the I/Os are executed at a sector granularity (i.e., 512B I/Os) in our algorithms. The detailed configuration of our test platform is given in Section 9.1.

8.1. SSF algorithms

Algorithm 1 presents the pseudocode of the insertion and deletion operations. Since a deletion is treated as an insertion by the SSF, the algorithm is the same for both operations. The only difference is that for an insertion the frequency f_t of each document term is a positive value, whereas for a deletion the term frequencies f_t are negative values. An inserted/deleted document is represented in the index by a set of triples $\{(t, f_t, d)\}$ with t a (distinct) term and f_t its frequency in a document and d the identifier of that document. Each triple is initially in RAM (line 21 in Algorithm 1) until the buffered RAM partition reaches the *RAM_Bound* limit. Then the RAM partition is written in Flash and its inverted lists are indexed on the terms with a non-dense B⁺-tree (see line 8 in Algorithm 1 and also Algorithm 4). After the RAM flush, a merge (see Algorithm 2) is triggered if the number of partitions in the level 0 of the SSF reaches the branching factor b . The algorithm also checks if other merges are required in the upper SSF levels (lines 12 to 18 in Algorithm 1), since a merge in an SSF level creates a new partition in the subsequent level.

Algorithm 2 presents the pseudocode for the merge operation. The *merge* is directly processed in pipeline as a multi-way merge of all partitions at the same level. This is possible since the dictionaries of all the partitions are already sorted on terms, while the inverted lists in each partition are also sorted on document ids. So are the dictionary and the inverted lists of the resulting partition

Algorithm 1 SSF insertion / deletion.

Input: A input stream D of triples $\{(t, f_t, d)\}$ with t a (distinct) term and f_t its frequency in a document and d the identifier of that document.

Output: \emptyset .

```

1.  $P_{RAM}$  the RAM partition made of  $\{ \langle t, f_t, \{(d, f_{d,t})_{\downarrow d} \rangle \}_{\downarrow t} \}$ ;
2. for each triple  $e \in D$  do /* fetch the next triple  $e$  from the input stream */
3.   if  $\text{sizeof}(P_{RAM}) = \text{RAM\_Bound}$  then /* Flush the RAM partition in Flash */
4.     write  $\{t, f_t, \{(d, f_{d,t})_{\downarrow d}\}_{\downarrow t}$  from  $P_{RAM}$  to a chained list of Flash sectors;
5.      $\text{ptr}$  = address of the first written Flash sector of the new partition;
6.      $i$  = smallest index  $i \mid P[0][i] = \emptyset$ ; /* this smallest  $i$  always exists here and  $i \in [1, b]$  */
7.     /* Build the hierarchical index and return the address of the root sector */
8.      $P_{RAM} = \emptyset$ ; /* free  $P_{RAM}$  */
9.      $\text{root} = \text{Build\_hierarchical\_index}(\text{ptr})$ ; /* see Algorithm 4 – NB:  $P_{RAM}$  is released at this stage */
10.     $P[0][i] = \text{root}$ ;
11.    if  $(i = b)$  /* Merge the partitions if nb of partitions in the level reaches the branching factor  $b$  */
12.       $l = 0$ ;
13.      while  $P[l][p] \neq \emptyset, \forall p \in [1, b]$  do /* while level  $l$  contains  $b$  partitions */
14.        Merge( $l$ ); /* see Algorithm 2 */
15.        for  $p = 1$  to  $b$  do /* free all partitions in level  $l$  */
16.           $P[l][p] = \emptyset$ ;
17.        end
18.         $l++$ ;
19.      end
20.    end
21.    insert the triple  $e$  into  $P_{RAM}$ ;
22. End

```

Algorithm 2 SSF merge.

Input: l level of the SSF with the partitions to be merged.

Output: \emptyset .

```

1.  $ln[b]$  an array of  $b$  sectors allocated in RAM; /*  $b$  RAM pages to concomitantly read the  $b$  partitions in level  $l$  */
2.  $Out$  a sector allocated in RAM; /* one RAM page to temporarily buffer the partial result of the merge */
3.  $Ptr$  the address of a Flash sector;
4.  $Ptr[b]$  an array of  $b$  addresses of Flash sectors;
5.  $\text{memset}(Out, \emptyset)$ ; /* initialize  $Out$  */
6.  $\forall x \in [1, b], Ptr[x] = \text{Access\_hierarchical\_index}(P[l][x], -1)$  /* lowest term */;
7. /* initialize  $Ptr$  with the first Flash sector of each partition */
8. /* Perform in pipeline the Multi-way merge */
9. while  $\exists Ptr[x] \neq \emptyset \mid x \in [1, b]$  do
10.   for  $x = 1$  to  $b$  do
11.     if  $Ptr[x] \neq \emptyset$  and  $Ram[x]$  is empty then
12.        $ln[x] = \text{load Flash sector } Ptr[x]$ ; /*  $ln[x] \supset$  elements of  $\{t, f_t, \{(d, f_{d,t})_{\downarrow d}\}_{\downarrow t}\}$  */;
13.        $Ptr[x] = Ptr[x] + \text{sizeof}(\text{Flash\_sector})$ ; /* address of the next partition sector in Flash */
14.     end
15.   end
16.    $t_{\text{frontier}} = \min(\{\forall x \in [1, b], \max(\{t \in ln[x]\})\})$ ; /* find the border term for all the terms in RAM, i.e., all the terms inferior to the border term can be safely merged */
17.   while  $\exists x \in [1, b]$  and  $t \in ln[x] \mid t \leq t_{\text{min}}$  do
18.     if  $Out$  is full then
19.       write  $Out$  in the next free Flash sector;
20.        $\text{memset}(Out, \emptyset)$ ; /* Delete content of  $Out$  */
21.     end
22.      $t_{\text{next}} = \min(\{t \in ln[x] \mid x \in [1, b]\})$ ;
23.      $E = \{\text{elements } e_x \text{ of type } \langle t, f_t, \{(d, f_{d,t})_{\downarrow d}\}_{\downarrow t} \} \mid e_x \in ln[x], e_x.t = t_{\text{next}}, x \in [1, b]\}$ ;
24.     write in  $Out$  the element  $\langle t_{\text{next}}, \sum_{1 \leq x \leq b} e_x.f_t, \{e_1.\{(d, f_{d,t})_{\downarrow d}\}_{\downarrow t} + \dots + e_b.\{(d, f_{d,t})_{\downarrow d}\}_{\downarrow t}\} \rangle$ ;
25.     /* the resulted list of  $t_{\text{next}}$  the concatenation of all the partial lists of  $t_{\text{next}}$  in the merged partitions */
26.   remove all the elements  $e_x \in E$  from  $ln$ ;
27. end
28. write  $Out$  in the next free Flash sector;
29.  $\text{memset}(Out, \emptyset)$ ;
30. end
31. free( $ln$ ); free( $Out$ );
32.  $\text{root} = \text{Build\_hierarchical\_index}(Ptr)$ ; /* index the terms of the newly created partition */
33.  $p$  = smallest index  $p \mid P[l+1][p] = \emptyset$ ;
34.  $P[l+1][p] = \text{root}$ ; /* store in the index metadata the root of the index of the new partition */
35. end

```

Algorithm 3 SSF search.

Input: $Q = \{q_i\}$ a set of q query terms; k the requested number of results (top- k).
Output: $R[k]$ an array of k couples $(d, tfidf_score)$ of document identifiers and their $tfidf$ score.

1. $Ft[q]$ an array of q values to store the Ft frequency for each query term initialized at 0;
2. $Ptr[q][l][b]$ a set of pointers to the start of the inverted lists of each query term in each partition in each index level;
3. $Ptr = \text{Compute_Ft}(Q, Ft)$; /* compute the F_t for each query term, see Algorithm 5 */
4. $R = \text{Compute_Top_k}(Q, Ft, Ptr, k)$; /* compute the top- k results (in pipeline), see Algorithm 6 */
5. **return** R ;

at the upper level. More precisely, the algorithm works in two steps. In the first step, the IL part (i.e., the set of inverted lists) of the output partition is produced (lines 6–27 in Algorithm 2). Given b partitions in the index level L_i , $b + 1$ RAM pages are necessary to process the merge in linear pipeline: b pages to merge the inverted lists in IL of all b partitions (line 1 in Algorithm 2) and one page to produce the output (line 2 in Algorithm 2). The indexed terms are treated one after the other in alphabetic order (line 6 in Algorithm 2). For each term t , the head of its inverted lists in each partition is loaded in RAM (line 10 in Algorithm 2). These lists are then consumed in pipeline by a multi-way merge (lines 21 and 22 in Algorithm 2). Document ids are encountered in descending order in each list and the output list resulting from the merge is produced in the same order. For the b partition pages loaded in RAM, a border term is computed (line 14 in Algorithm 2). Then, all the terms inferior to the border term can be safely merged and their inverted lists written in the new partition. The border term is updated whenever a new partition page is loaded in RAM (line 10 in Algorithm 2). In the second step (line 29 in Algorithm 2), the IS structure is constructed sequentially, with an additional scan of IL (see Algorithm 4). This Background Merging process generates only sequential writes in Flash and previous partitions are reclaimed in large blocks after the merge. This pipeline process sequentially scans each partition only once and produces the resulting partition also sequentially. Hence, assuming $b + 1$ is strictly lower than RAM_bound , one RAM buffer (of one page) can be allocated to read each partition and the merge is I/O optimal. If b is larger than RAM_bound , the algorithm remains unchanged but its I/O cost increases since each partition will be read by page fragments rather than by full pages. Hence, the memory consumption of the merge operation will be lower than the RAM_Bound in all cases. The merge algorithm also requires storing $b + 1$ pointers in RAM. However, the RAM consumption for these variables represents only a fraction of a RAM page and is negligible compared to the $b + 1$ RAM pages required by the multi-way merge.

Algorithm 3 presents the query processing algorithm in the SSF. Given a set of query terms an integer value k , the algorithm returns an array of k couples $(d, tfidf_score)$ of document identifiers and their $tfidf$ score. The search algorithm consists in two steps. First, the Ft value of each query term is computed (line 3 in Algorithm 3). This part is described in detail in Algorithm 5. Second, the list of top- k documents with the highest scores is obtained (line 4 in Algorithm 3). This part is described in detail in Algorithm 6.

Algorithm 4 presents the pseudocode of the construction of the hierarchical index (i.e., the IS structure) on top of the set of inverted lists in each partition. The algorithm is invoked in Algorithm 1 (i.e., after the creation of a new partition in the first SSF level) and in Algorithm 2 (i.e., after the creation of a new partition by merging the partitions of an SSF level). The IS structure is constructed sequentially and requires a single full scan of IL previously created. The IS tree is built from the leaves to the root. This requires one RAM page to scan IL (line 1 in Algorithm 4), plus one RAM page to write the IS (line 2 in Algorithm 4). For each Flash page of the IL containing at least one head-list (line

8 in Algorithm 4), the maximum term and its Flash address are indexed in the index leaves (lines 10 and 13 in Algorithm 4). Once the bottom index level is created, the upper levels of IS are trivially filled sequentially in the same manner through a recursive call (line 24 in Algorithm 4).

Algorithm 5 presents the computation of the F_t values for the query terms, which represents the first phase of the query processing. F_t is computed only once for each term t since F_t is constant for Q . This is why F_t is materialized in the dictionary part of the index $(\{t, F_t\} \subset IS)$, as shown in Fig. 2. Since I is split in (I_0, I_1, \dots, I_p) , the global value of F_t is computed as the sum of the local F_t of all partitions (lines 2 to 10 in Algorithm 5). The algorithm visits all the SSF partitions (lines 2 and 3 in Algorithm 5) and in each partition it uses the IS structure to access the inverted lists corresponding to the query terms (lines 4 and 5 in Algorithm 5). If a query term is found, its global F_t value is increased with the local f_t value (line 10 in Algorithm 5). For the sake of simplicity, we do not consider in Algorithm 5 the case of the documents overlapping between consecutive partitions. The overlapping documents are detected by checking the two bits (i.e., $firstd$ and $lastd$) in IS (see Fig. 3). Whether an intersection between two lists is detected, the sum of their respective F_t must be decremented by 1. Hence, the correct global value of F_t can easily be computed without physically accessing the inverted lists. During the F_t computation phase, the dictionary of each partition is read only once and the RAM consumption sums up to one buffer to read each dictionary, page by page, and one RAM variable to store the current value of each F_t . In addition, a set of pointers to the start Flash addresses of the inverted list for each query term in each SSF partition is also stored in RAM to avoid re-accessing the IS of each partition in the second phase of the query processing.

Algorithm 6 presents the pseudocode to compute the top- k document identifiers and their scores for a set of query terms, which represents the second phase of the query processing in the SSF. The algorithm takes as input the F_t values of the query terms and set of pointers to the start Flash addresses of the inverted list for each query term in each SSF partition, priorly computed by Algorithm 5. The proposed algorithm works as follows. For each SSF level from the lowest to the highest one (line 5 in Algorithm 6), all the partitions of the SSF level are accessed from the most recent to the oldest one (line 6 in Algorithm 6). For each partition, the $tfidf$ computation sums up to a simple linear pipeline merging process of the inverted lists for all terms $t \in Q$ (lines 8 to 32 in Algorithm 5). The RAM consumption (line 1 in Algorithm 6) for this phase is therefore restricted to one buffer (i.e., a RAM page) per query term t to read the corresponding inverted lists $I_i.L_t$ (i.e., $I_i.L_t$ are read in parallel for all t , the inverted lists for the same t being read in sequence). In addition, two lists are maintained in RAM (lines 2 and 3 in Algorithm 6): $R[k] = \{(d, score(d))\}$ contains the current k best $tfidf$ scores of documents which exist with certainty (no deletion has been encountered for these documents); $Ghost = \{(d, score(d))\}$ contains the list of documents which have been deleted (a pair $(d, -f_{d,t})$ has been encountered while scanning the inverted lists) and have a score better than the smallest score in $R[k]$. The $tfidf$ score of each

Algorithm 4 Build hierarchical index.**Input:** *ptr* the beginning address of the sorted inverted lists in a partition.**Output:** *root* the address of the root Flash sector of the index.

```

1. ramin = Alloc_RAM (sizeof(Flash_sector));
2. ramout = Alloc_RAM (sizeof(Flash_sector));
3. ptrin = ptr; /* pointer to the head Flash sector of the list { < t, ft, {(d, fd,t)d > }t */
4. ptrout = address of the next free sector in Flash;
5. no_nodes = 0; /* number of nodes in the currently built index level */
6. while ptrin ≠ ∅ do
7.   load in ramin the Flash sector at address ptrin;
8.   get max(t) in ramin; /* find the last vocabulary term in this page */
9.   if t ≠ ∅ then
10.    append (t, ptrin) to ramout;
11.   end
12.   if ramout is full then
13.    write ramout at address ptrout;
14.    no_nodes ++;
15.    if no_nodes = 1 then
16.     ptr = ptrout; /* keep the address of first index node in the current index level for recursive call */
17.    end
18.    ptrout = ptrout + sizeof(Flash_sector); /* address of the next free sector in Flash */
19.   end
20.   ptrin = ptrin + sizeof(Flash_sector); /* get the address of the next Flash sector */
21. end
22. free (ramin); free (ramout);
23. if no_nodes > 1 /* if the current index level has more than one node then recursively index this level */
24.   ptr = Build_hierarchical_index(ptr);
25. end
26. return ptr;

```

Algorithm 5 Compute *F_t*.**Input:** a set $Q = \{q_i\}$ of q query terms, $F_t[q]$ an array of F_t frequency values with $F_t[i]$ the value for q_i .**Output:** $Ptr[q][l][b]$ a set of pointers with $Ptr[q_i][l_j][p]$ having the start Flash address of the inverted list for query term q_i of partition p in level l_j .

```

1. ptr a pointer to store the address of a Flash sector;
2. for  $l=0$  to max ( $\{i \mid P[i][0] \neq \emptyset\}$ ) do /* for each level starting from the first one */
3.   for  $p=1$  to max ( $\{j \mid P[l][j] \neq \emptyset\}$ ) do /* for each partition of that level */
4.     for  $i=1$  to  $q$  do /* for each query term */
5.       ptr = Access_hierarchical_index( $P[l][p]$ ,  $q_i$ );
        /* find the Flash sector containing the inverted list of  $q_i$  in this partition */
6.       load in RAM the Flash sector at address ptr;
7.       search in RAM the entry of the element  $e = \{q_i, f_i, \{(d, f_{d,t})_{d,t}\}$ ;
8.       if  $e$  exists then
9.          $F_t[i] = F_t[i] + e.f_i$ ;
10.        compute  $Ptr[i-1][l][p-1]$  as the address in Flash of the start of the list  $e.\{(d, f_{d,t})_{d,t}\}$ ;
11.       end
12.     end
13.   end
14. end
15. return Ptr;

```

document d is computed (line 15 in Algorithm 6) by considering the modulus of the frequencies values $|\pm f_{d,t}|$ in the *tf-idf* score computation, regardless of whether d is a deleted document or not. $R[k]$ and *Ghost* lists are managed as follows. If the score of the current document d is worse than the smallest score in $R[k]$, it is simply discarded and the next document is considered (line 16 in Algorithm 6). Otherwise, two cases must be distinguished. If d is a deleted document (a pair $(d, -f_{d,t})$ is encountered), then it enters the *Ghost* list (line 18 in Algorithm 6); else it enters the $R[k]$ list unless its id is already present in the *Ghost* list (lines 20 to 22 in Algorithm 6). Note that this latter case may occur only if the id of d is smaller than the largest id in *Ghost*, making the search in *Ghost* useless in many cases. An important remark is that the *Ghost* list has to register only the deleted documents which may compete with the k best documents (line 23 in Algorithm 6), to filter them out when these documents are later encountered, which makes this list very small in practice. In addition, the probability that the score of a *Ghost* list element competes with the $R[k]$ ones decreases over time, giving the opportunity to continuously purge the *Ghost* list (line 23 in Algorithm 6). If the

Ghost list overflows (i.e., exceeds the size of a RAM page), it is sorted in descending order of the document ids, and the entries corresponding to low document ids are flushed. For simplicity, we omitted the flushing process from Algorithm 6. Note also that this situation highly improbable as explained in Section 6.3. None of the queries we evaluated in our experiments has produced an overflow of the RAM page allocated to the *Ghost* list.

8.2. Memory consumption analysis and bounds

We show in this section that the RAM consumption of the algorithms presented above, which implement all the SSF operations (index maintenance and search), never exceeds a predefined and small bound. As indicated in Section 2.2, we consider that a Flash sector has 512 bytes here after. The extension of the analysis to a page granularity is straightforward.

8.2.1. Insertion/deletion

These operations are managed by Algorithm 1 (SSF insertion/deletion), which may subsequently call Algorithms 2 (SSF

Algorithm 6 Compute top- k .

Input: $Q = \{q_i\}$ a set of q query terms; $Ft[q]$ an array of size q with $Ft[i]$ the Ft values of the query term q_i ; $Ptr[q][l][b]$ a set of pointers with $Ptr[q_i][l][p]$ storing the address in Flash of the inverted list element $\langle t, f_i, \langle d, f_{d,t} \rangle_{\downarrow d} \rangle$ for partition p , level l , and a term q_i ; k the number of documents identifiers requested in result.

Output: $R[k]$ an array of couples $(d, tfidf_score)$ of document identifiers and their $tfidf$ score.

```

1.  $Ram[]$  an array of  $q$  sectors allocated in RAM;
2.  $\forall x \in [0, k-1], R[x].tfidf\_score = 0$ ; /* initialize the tfidf scores in the result at 0 */
3.  $Ghost$  one RAM page to temporarily maintain the current best ranked deleted documents still appearing in the index
4. /* Multi-way merge of the inverted lists of the query terms in each index partition */
5. for  $l = 0$  to  $\max(l \mid \exists Ptr[q_i][l][p] \in Ptr, p \in [1, b], i \in [1, q])$  do /* for each level containing query terms */
6.   for  $p = \max(p \mid \exists Ptr[q_i][l][p] \in Ptr, i \in [1, q])$  to 1 do /* for each partition with query terms */
7.     for each  $i \mid \exists Ptr[q_i][l][p] \in Ptr$  do /* for each query term */
8.       load in  $Ram[i]$  the first sector of the inverted list  $\langle d, f_{d,t} \rangle_{\downarrow d}$  of  $q_i$  from address  $Ptr[i][l][p-1]$ ;
9.        $Ptr[i][l][p-1] = Ptr[i][l][p-1] + \text{sizeof}(\text{Flash\_sector})$ ;
10.      /* compute the next sector address of the current inverted list of  $q_i$  */
11.    end
12.     $d_{frontier} = \min(\{\forall i \in [1, q], \max(\{d \in Ram[i]\})\})$ ; /* find the border document id for all the
13.      documents in RAM, ie. all the doc ids inferior
14.      to the border doc id can be safely scored */
15.    while  $(\exists a \text{ couple } (d, f_{d,t}) \in Ram \mid d \leq d_{frontier})$  do /* compute tfidf for the next docs */
16.       $d_{next} = \min(\{d \in Ram\})$ ;
17.       $E = \{\text{couples } (d, f_{d,t}) \in Ram \mid d = d_{next}\}$ ;
18.       $tfidf\_score = \text{Compute\_TFIDF}(d_{next}, E, Ft)$ ; /* compute TF-IDF for  $d_{next}$  */
19.      if  $tfidf\_score > \min(\{tfidf\_score \in R\})$  then /* if the score of  $d_{next}$  enters current best  $k$  scores */
20.        if  $d_{next}$  is a deleted document then
21.          insert  $(d_{next}, tfidf\_score)$  into  $Ghost$ 
22.        else
23.          if  $d_{next} \notin Ghost$  then
24.            delete from  $R$  the entry with minimum  $tfidf\_score$ ;
25.            insert in  $R$   $(d_{next}, tfidf\_score)$ ;
26.            delete from  $Ghost$  all entries  $d_{ghost}$  having  $\text{score}(d_{ghost}) < \min\_score(R)$ ;
27.            remove from  $Ram$  all the couples  $e \in E$ ;
28.          end
29.        end
30.      end
31.      for each empty sector  $Ram[i] \in Ram \mid Ptr[i][l][p-1] \neq \emptyset$  do /* scan next sectors */
32.        load in  $Ram[i]$  the inverted list  $\langle d, f_{d,t} \rangle_{\downarrow d}$  of  $q_i$  from address  $Ptr[i][l][p-1]$ ;
33.         $Ptr[i][l][p-1] = Ptr[i][l][p-1] + \text{sizeof}(\text{Flash\_sector})$ ;
34.      end
35.       $d_{frontier} = \min(\{\forall i \in [1, q], \text{Max}(\{d \in Ram[i]\})\})$ ;
36.    end
37.  end
38.  end
39.   $\text{free}(Ram)$ ;  $\text{free}(Ptr)$ ;
40.  return  $R$ ;

```

Merge) and 4 (Build hierarchical index). The memory consumption of Algorithm 1 is limited to the current triple e to be inserted or deleted (line 2 of Algorithm 1) and the RAM partition of the SSF called P_{RAM} (line 1 in Algorithm 1), which is systematically flushed to the Flash storage and then freed from the RAM (line 7) whenever it reaches a predefined fixed size, i.e., RAM_Bound . The *Build_hierarchical_index* algorithm is always called after P_{RAM} has been released from the RAM (line 8) and consumes the size of two Flash sectors allocated in RAM (lines 1 and 2 of Algorithm 2) and a few local variables (i.e., two pointers and one integer). Although *Build_hierarchical_index* can be called recursively (line 24 in Algorithm 4), a recursive call always happens after the two sectors allocated in RAM are freed (line 22). The *Merge* algorithm is called at line 13 of Algorithm 1 after P_{RAM} is freed and after *Build_hierarchical_index* has returned and released the memory (at line 22 of Algorithm 4). *Merge* consumes a RAM size of $b + 1$ Flash sectors (line 1 and 2 of Algorithm 2), b pointers (line 4) and a few local variables. Note that the value of b (the branching factor) is fixed and is chosen such that the RAM consumption of *Merge* never exceeds RAM_Bound . Omitting the few RAM variables needed in the algorithms, the RAM consumption of Algorithms 1, 2 and 4 is equal to: $\text{MAX}(\text{sizeof}(\text{inserted triple } e) + \text{sizeof}(\text{RAM partition } P_{RAM}); 2 * \text{sizeof}(\text{Flash_Sector}); (b + 1) * \text{sizeof}(\text{Flash_Sector}))$.

8.2.2. Search

The search operation is managed by Algorithm 3 (SSF search), which calls Algorithms 5 (Compute Ft) and 6 (Compute Top- k). For a search on q query terms, the RAM consumption of Algorithm 3 is the size of the q query terms, the k couples $(d, tfidf_score)$ being the result of the query, an array of q numeric values (line 1) used to store the frequency of each query term, and the size of an array of $q * l * b$ pointers to the inverted lists of each of the q query term in the b partitions of the l index levels (line 2). Algorithm 5 only consumes few local variables and Algorithm 6 consumes an array of q sectors allocated in RAM to scan the SSF partitions. Omitting the few RAM variables needed in these algorithms, the RAM consumption of Algorithms 3, 5 and 6 is equal to: $q * \text{sizeof}(\text{query_terms}) + k * \text{sizeof}(d, tfidf_score) + q * l * b * \text{sizeof}(\text{pointer}) + q * \text{sizeof}(\text{Flash_sector})$.

The curves in Fig. 8 show the RAM consumption of our algorithms with an increasing number of indexed documents in the database. We fix the maximum number of terms q in the search queries to 5, the size of the RAM partition P_{RAM} to 1024 bytes, the number of results k to 10, the branching factor b to 4 (left curve) and 8 (right curve), and we increase the database size up to 4GB. These settings are similar with the ones we use in the experimental evaluation here below, except the index size which is much larger than the indexes we obtained with our test datasets. We intentionally increase the index size to a very large value to

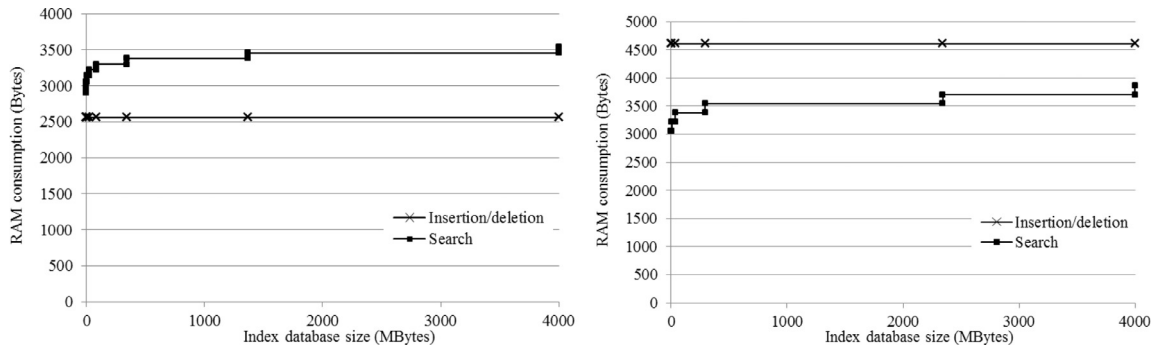


Fig. 8. (a) RAM consumption varying the index database size with $b=4$; (b) RAM consumption varying the index database size with $b=8$.

show that the database size has only negligible impact on the RAM consumption after a certain volume of data (e.g., 500MB) has been indexed.

These curves illustrate the compliance of the proposed structure and algorithms with the *Bounded RAM agreement*. With $b=4$ (respectively $b=8$), the RAM requirement never exceeds 3500 bytes (resp. 4600 bytes). Increasing the value of the branching factor b has a negative impact on the RAM consumption required to insert/delete documents. Typically, the *Merge* algorithm requires more memory to comply with the linear pipelining rule. Oppositely, increasing b has a slightly positive impact on the RAM consumption required to evaluate search queries since with less number of levels in the SSF, less pointers are allocated in RAM (line 2 in Algorithm 2). Overall, with these settings, the RAM consumption is lower than 5KB even for an index size of 4GB. Note that the background implementation of the *Merge* does not require additional RAM, since the successive incremental (partial) merge operations are triggered in Algorithm 1, after the Flush of P_{RAM} . We show in the evaluation section below that the background merge process performed incrementally at each P_{RAM} flush terminates far before the next merge operation starts.

We also note that in case a page granularity access (i.e., 2KB I/Os) is used instead of a sector granularity access (i.e., 512B I/Os), the RAM consumption increases proportionally with the size of the I/O since it mainly depends on the I/O size. For instance, the maximum RAM consumption will augment from 5KB to 20KB in the configuration having the branching factor $b=8$.

9. Full scalability conformance

In this section we present an extensive performance evaluation of the proposed search engine to assess whether it complies with the Full scalability property. We introduce the testing hardware platform, the used datasets and the related use-cases in Section 9.1. In Section 9.2, we discuss the index maintenance, i.e., insertion/merge cost and the frequency of the merges. The query performance of the search engine is presented in Section 9.3. The index search performance for advanced keyword-based functionalities is analyzed in Section 9.4. The impact of the deletion rate on the index size and the query performance is discussed in Section 9.5. We also compare both the search and the insert performance of our method with two representative search engines in Section 9.6. Finally, in the light of the obtained experimental results, we discuss the limitations of our approach in Section 9.7.

9.1. Experimental setup

All the experiments have been conducted on a development board ST3221G-EVAL (see Fig. 9) equipped with the MCU STM32F2171G connected to a MicroSD card slot. This hardware configuration is representative of typical secure tokens [5,6,23,34] or

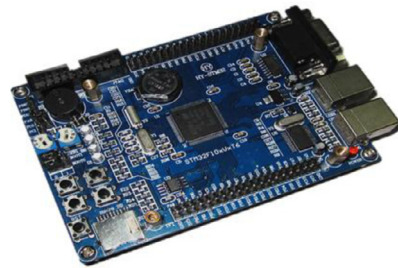


Fig. 9. The development board ST3221G-EVAL used in the experiments.

smart objects [33,35]. The board runs the embedded operating system RTOS 7.0.1 (see <https://sourceforge.net/p/freertos/code/HEAD/tree/tags/V7.1.0/>). The search engine code is stored on the internal NOR Flash memory of the MCU, while the inverted index is stored on a MicroSD NAND Flash card. We tested our index structure with several commercial MicroSD cards (see Table 1). For the complete set of experimental results presented in this section, we selected two representative MicroSD cards (i.e., Kingston MicroSDHC Class 10 4GB and Silicon Power SDHC Class 10 4GB) exhibiting different performance as measured on our development board (see lines 1 and 4 in Table 1). The MCU has 128 KB of available RAM. However, the search engine only uses a maximum amount of 5 KB of RAM, to validate our design whatever the available RAM of existing secure tokens and whatever the fragment of this RAM allocated to the search engine running in competition with other embedded software. To achieve this low RAM_Bound, we access the NAND Flash at a sector granularity (i.e., 512 bytes) as indicated in Section 8.2.

We implemented the proposed method and used Microsearch [33] and the classical inverted index [42] for comparison. All the tested methods have been implemented in the C language. The code is compiled and then flushed and executed in the MCU of the development board. The SSF is the most complex method to implement. The inverted index can be seen as a simplified version of the SSF since it does not use partitioning (and therefore there are no merge operations). However, the insertions/deletions are directly applied to the index structure, generating thus a large number of costly in-place updates. Microsearch is also fairly simple to implement since its index structure mainly consists in a set of reversed linked lists with an in-memory table containing a pointer to the Flash address of the last added page of each list. A list corresponds to a hash bucket and consequently contains all the index terms associated with the bucket. Additional details about the implementation of the competing methods are given in Section 9.6.

9.1.1. Datasets and queries

Selecting a representative data and query set to evaluate our solution is challenging considering the diversity and quick evolu-

tion of secure token usages, explaining the absence of recognized benchmarks in this area. We then consider two use-cases where an embedded keyword-based search engine is called to play a central role and which exhibit different requirements in terms of document indexing, with the objective to assess the versatility of the solution.

The first use-case is in the Personal Cloud context and considers the use of a secure token embedding a Personal Data Server [4,5,34] to securely store, query and share personal files (documents, photos, emails) as presented in Section 1. This use-case is representative of situations where the indexing documents have a rich content (tens to hundreds of thousands of terms) and documents updates and deletes can be performed randomly. Protecting and querying the content captured by a set-top-box registering watched TV programs and videos with their related metadata thanks to a secure chip integrated in the set-top-box is no more utopia and is another example of this context. To capture the behavior of our solution in such context, we use two referenced, representative data sets (i.e., ENRON and a pseudo-desktop document collection) and their associated query sets as described below. The interest of using these datasets is that they are large enough to test the index scalability and are well recognized in the IR community.

The second use-case targets the smart sensor context and the case where documents with a poor content are integrated in a Personal Cloud. For instance, home gateways capture a variety of events issued by a growing number of smart appliances, car trackers register our locations and driving habits to compute insurance fees and carbon tax [4]. Here, the documents are time windows, the terms are events occurring during this time window, and top-*k* queries are useful for analytic tasks. Executing the queries at the sensor side helps reducing the cost, energy consumption and risk of private information leakage. This use-case is representative of situations where the indexing documents have a poor content (hundreds to thousands of terms/event types). Similarly, in the Personal Cloud, poor content documents are typically the binary files (e.g., photo, music or video files) that contain some terms describing the file content (e.g., for a music file the terms may indicate the artist, album, song title, release date, genre, etc.). Therefore, a poor document collection in this case corresponds to a user who mainly stores and indexes binary files in her Personal Cloud. We are not aware of publicly available representative datasets for this context and therefore generate a synthetic dataset for this use-case.

Hence, our evaluation is based on three datasets (see Table 2), which, by their diversity, cover a significant part of the possible use-cases related to smart objects. To evaluate the scalability and efficiency of the search engine, we use the ENRON dataset (available at <https://www.cs.cmu.edu/~enron/>) composed of 0.5 million emails and a query set of 300 representative queries prepared for this dataset (available at http://www.prism.uvsq.fr/~isap/files/ENRON_queries.zip) built for this dataset. The statistics of the dataset and the queries are given in Table 2. The total number of terms extracted from the ENRON dataset is 565,343 terms, among which 30,624 are frequent. We did not do any text pre-processing (e.g., stemming, feature selection, stop word removal, correction of typos, etc.) of the dataset, which partially explains the very large number of unique terms. However, the text processing is orthogonal to this work since our main concern is the search engine efficiency and not its accuracy. Also, the high number of terms is useful to show the scalability of the proposed search engine.

The second dataset is represented by the pseudo-desktop collection of documents presented in [20]. As ENRON, this dataset applies to the Personal Cloud use-case. The prominent difference from ENRON is that this dataset contains five representative types of personal files (i.e., email, html, pdf, doc and ppt) and not only emails as in ENRON. The desktop search is an important topic in

Table 2
Statistics of the datasets and the query sets.

ENRON data set and query set	
Number of documents	500,000
Total Raw Text	946 MB
Total Unique Words	565,343
Total Word Occurrences	52,410,653
Average Occurrences per Word	92
Frequent Words	30,624
Infrequent Words	534,719
Frequent Word Occurrences	5.41%
Infrequent Word Occurrences	94.58%
Size of documents in bytes (avg, max)	1 KB, 874 KB
Size of documents in words (avg, max)	180, 108,026
Total number of queries	300
Number of queries with 1, 2, 3, 4 and 5 terms	51, 179, 48, 13, 9
Pseudo-desktop dataset and query set	
Number of documents	27,000
Total Raw Text	252 MB
Total Unique Words	337,952
Total Word Occurrences	35,624,875
Average Occurrences per Word	26
Frequent Words	20,752
Infrequent Words	317,210
Frequent Word Occurrences	6.14%
Infrequent Word Occurrences	93.85%
Size of documents in bytes (avg, max)	8 KB, 647 KB
Size of documents in words (avg, max)	1304, 105,162
Total number of queries	837
Number of queries with 1, 2, 3, 4 and 5 terms	85, 255, 272, 172, 82
Synthetic dataset and query set	
Number of documents	100,000
Total Raw Text	129 MB
Total Unique Words	10,000
Total Word Occurrences	10,000,000
Average Occurrences per Word	988
Frequent Words	1968
Infrequent Words	8032
Frequent Word Occurrences	19.68%
Infrequent Word Occurrences	80.32%
Size of documents in bytes (avg, max)	1.3 KB, 1.3 KB
Size of documents in words (avg, max)	100, 100
Total number of queries	1000
Number of queries with 1, 2, 3, 4 and 5 terms	200, 200, 200, 200, 200

the IR community. However, real personal collections of desktop files cannot be published because this raises evident privacy issues. Instead, the authors in [20] propose a method to generate synthetic (pseudo) desktop collections and show that such collections have the same properties as real desktop collections. We use in our experiments the pseudo-desktop collection provided in [20]. The statistics of this collection are given in Table 2. As recommended in [20], we preprocess the files in this collection by removing the stop words and stemming the remaining terms using the Krovetz stemmer. This explains the smaller number of terms in the vocabulary (i.e., 337,952) compared to ENRON (565,343). Nevertheless, the vocabulary is still rich and contains a high number of terms. In addition, the average size of the files is about 8 times larger in the pseudo-desktop collection than in ENRON since the desktop collection contains not only emails but also larger html, pdf, doc and ppt files. In our experiments, we use a set of 837 queries prepared for this dataset and provided in [20].

Finally, the third dataset is a synthetic dataset that we generated to consider the second use-case introduced above. More precisely, we consider the case of a smart meter deployed at home to enable a new generation of energy services. The smart meter records events reported by any smart object in the house (e.g., a specific washing program of a washing machine, played or recorded TV channel for a set-up box, triggering lights or air

Table 3
Statistics of the flush and merge operations with the ENRON dataset.

	Flush [$RAM \rightarrow L_0$]	Merge [$L_0 \rightarrow L_1$]	Merge [$L_1 \rightarrow L_2$]	Merge [$L_2 \rightarrow L_3$]	Merge [$L_3 \rightarrow L_4$]	Merge [$L_4 \rightarrow L_5$]	Merge [$L_5 \rightarrow L_6$]
Number of Read IOs	1 (1)*	67 (72)	585 (738)	3510 (4210)	21,034 (23,229)	129,818 (14,550)	404,578 (404,578)
Number of Write IOs	9 (9)	82 (102)	463 (707)	2738 (3448)	17,703 (19,771)	119,357 (137,789)	389,774 (389,774)
Exec. time on Kingston (seconds)	0.008 (0.0084)	0.67 (0.82)	3.8 (5.6)	22.3 (29.7)	141.3 (158)	944 (1077)	3003 (3003)
Exec. time on Silicon Power (seconds)	0.0044 (0.0045)	0.41 (0.57)	2.54 (3.56)	15.2 (17.7)	92 (102)	604 (688)	1907 (1907)
Total number of occurrences	286,265	35,783	4473	559	70	9	1
No. of inserted docs between consecutive flushes/merges	2 (27)	17 (134)	132 (964)	1057 (4306)	8410 (20,061)	61,556 (106,601)	150,237 (150,237)

* The numbers given in brackets are maximum values, other values are average values.

Table 4
Statistics of the flush and merge operations the pseudo-desktop dataset.

	Flush [$RAM \rightarrow L_0$]	Merge [$L_0 \rightarrow L_1$]	Merge [$L_1 \rightarrow L_2$]	Merge [$L_2 \rightarrow L_3$]	Merge [$L_3 \rightarrow L_4$]	Merge [$L_4 \rightarrow L_5$]
Number of Read IOs	1 (1)*	90 (92)	503 (617)	2027 (2570)	11,010 (15,211)	50,997 (73,026)
Number of Write IOs	9 (9)	71 (100)	339 (548)	1485 (2085)	9409 (14,027)	47,270 (66,335)
Exec. time on Kingston (seconds)	0.008 (0.0084)	0.58 (0.77)	2.9 (4.44)	13.2 (19.1)	84.6 (124.4)	436 (615)
Exec. time on Silicon Power (seconds)	0.004 (0.0045)	0.38 (0.48)	1.94 (2.84)	8.67 (11.7)	54.5 (79.3)	278 (393)
Total number of occurrences	73,277	9160	1145	143	18	2
No. of inserted docs between consecutive flushes/merges	0.42 (16)	3 (42)	24 (232)	189 (1193)	1453 (6496)	8906 (10,547)

* The numbers given in brackets are maximum values, other values are average values.

conditioner in a certain room, etc.). A document in this case corresponds to a time window (e.g., of 1 h). The document terms are the event identifiers for the events that occur during the time window and their frequency. We generated a synthetic dataset (see Table 2) having a vocabulary of 10,000 terms. The synthetic collection contains 100 thousand files, which corresponds to a history of events of about 10 years considering that each file covers a one hour window. On average, each file contains 100 terms. Compared to the previous two datasets, this synthetic dataset covers the use-cases in which the documents have poor content (i.e., small vocabulary and small to average document size). We also generated a set of 1000 random queries to test the index query performance with this dataset.

9.2. Index maintenance

According to the algorithms presented earlier, the insertions and deletions of documents produce a sequence of index partitions which are subsequently merged in the SSF. Given the RAM_Bound of 5 KB, we set in all the experiments the branching factor b of the intermediate levels in the SSF to 8, to decrease the merge frequency, and the branching factor b' of the last level in the SSF to 3, to absorb faster the document deletions since the partitions of the last level are the largest.

The insertion or deletion of a new document is very efficient, since the document metadata is preliminarily inserted in RAM. Also, given the small size of the RAM_Bound, flushing the RAM content into the level L_0 of the SSF is fast; it takes on average around 6 ms to write a partition in L_0 in all our experiments (see Tables 3, 4 and 5). Given the low cost of the metadata insertion, we focus next on the SSF merge cost, which is periodically triggered (i.e., each time the number of flushed partitions in L_0 reaches the branching factor b). The merge of the partitions in L_0 generates a new partition in L_1 , which may generate a subsequent merge from L_1 to L_2 and in subsequent levels.

Tables 3, 4 and 5 present the number of IOs for the flush and merge operations performed in the different SSF levels, and their execution times for the three datasets on the two tested SD cards,

after the complete insertion of all the documents in the datasets and the random deletion of 10% of documents. In our experiments, the deletions are uniformly distributed over the inserted documents and uniformly interleaved with the insertions. All these operations lead to an SSF with 7 levels for the ENRON dataset (see Table 3), with 6 levels for the desktop dataset (see Table 4) and for the synthetic dataset (see Table 5). The number of levels of the index structure grows with the number of inserted documents and the average size of a document. As expected, the merge time grows exponentially from L_0 to L_6 , since the size of the partitions also increases by (nearly) a factor of b . It requires a few seconds to merge the partitions in the levels L_0 to L_3 and up to several minutes in L_4 to L_6 . The merge time is basically linear with the size of the merged partitions in the number of reads and writes. The merge time can vary especially in the first three levels of the SSF, depending on the distribution of the terms in the indexed documents. However, the partitions begin to contain most of the term dictionary in L_3 and the variation of the merge time in the upper levels is less significant. Note that with the pseudo-desktop collection, only fragments of documents are inserted in the first SSF level since the documents are large. Nevertheless, the document fragments are united in the subsequent SSF levels once the partitions are merged. This fragment condensation also explains the smaller partition sizes of the intermediate levels of the SSF with the desktop dataset compared with the other two datasets.

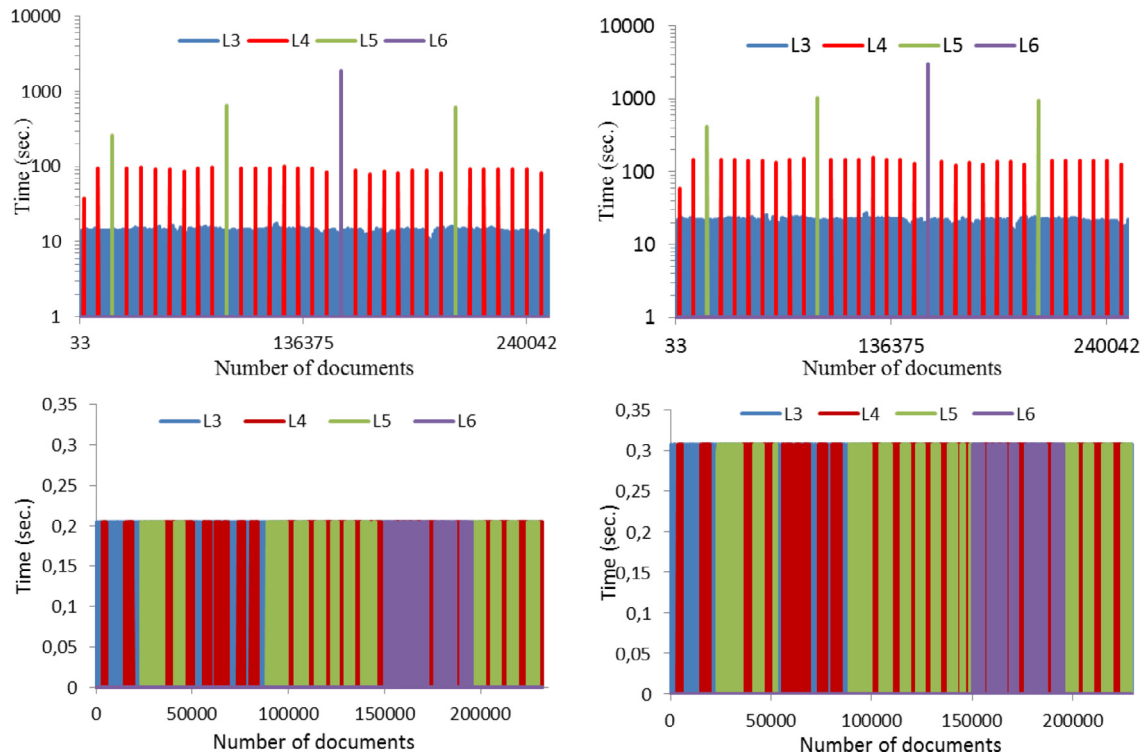
Tables 3, 4 and 5 indicate that the more costly a merge is, the less frequent it is. Typically, the merges in L_5 , which require many minutes to complete, occur after the insertion and deletion of about 61,000 ENRON documents or 9000 desktop documents. Only 80 merges costing more than 20 s are triggered while inserting the complete set of documents and deleting 10% of the ENRON collection. Even if the costly merges are rare, blocking the index when a merge is triggered for a long duration may be problematic for some applications. The merge operation is thus implemented in a non-blocking manner as explained in Section 5. After every RAM flush an additional time window of 340 ms for Kingston SD card and 210 ms for Silicon Power SD card is allocated to resume the current merge operation (if any). The time window is

Table 5

Statistics of the flush and merge operations the synthetic dataset.

	Flush [RAM→L ₀]	Merge [L ₀ →L ₁]	Merge [L ₁ →L ₂]	Merge [L ₂ →L ₃]	Merge [L ₃ →L ₄]	Merge [L ₄ →L ₅]
Number of Read IOs	1 (1)*	91 (93)	643 (652)	3873 (3934)	20,446 (21,877)	58,985 (58,985)
Number of Write IOs	9 (9)	88 (91)	511 (517)	2788 (2817)	17,910 (19,192)	56,037 (56,037)
Exec. time on Kingston (seconds)	0.008 (0.0084)	0.69 (0.71)	4.3 (4.3)	24.9 (25.1)	160.3 (172)	516 (516)
Exec. time on Silicon Power (seconds)	0.004 (0.0045)	0.44 (0.45)	2.77 (2.8)	16.4 (16.6)	103 (110)	328 (328)
Total number of occurrences	40,195	5025	628	79	10	1
No. of inserted docs between consecutive flushes/merges	2.43 (3)	19 (20)	155 (157)	1238 (1250)	9280 (9995)	22,846 (22,846)

* The numbers given in brackets are maximum values, other values are average values.

**Fig. 10.** Insert performances with blocking (up) and non-blocking (down) merge with silicon power storage (left column) and Kingston storage (right column) for the ENRON dataset.

chosen as the minimum time limit (in practice, we increase the minimum time with 10% to avoid any risk of merge overlapping) to guarantee that the merge of a given SSF level will end before the next merge of the same level is triggered. After this time delay, the merge is interrupted and its execution is memorized again. In this way, the potentially high cost of a merge operation is spread among a certain number of flush operations.

Fig. 10 compares the time to execute the merge operations in a blocking and non-blocking manner in the index levels from 3 to 6 with the ENRON dataset (similar results were obtained with the other two datasets). The merges in levels 0, 1 and 2 could not be represented because of their very low cost and high frequency. Also, we only represented the insertion of approximately half of the ENRON dataset in Fig. 10, since this is sufficient to capture the overall index update performance, while allowing for reasonable image clarity. We can observe that the cost of the merge in L₆ of the SSF is 1907 s (32 min) with the SP storage, if the merge is performed in a blocking manner. However, this cost will be spread among the next 11,483 insert/delete operations (each time the RAM is flushed) using non-blocking merges. Also, a merge triggered in a lower SSF level preempts the current merge in a higher level (if any).

Table 6

Blocking vs. non-blocking merge performance with ENRON.

	SD Card	Max. cost (sec.)	Avg. cost (sec.)
Blocking merge	Kingston	3003	0.23
	SP	1907	0.15
Non-blocking merge	Kingston	0.31	0.29
	SP	0.20	0.18

Table 6 compares the maximum and the average insertion/deletion time in the index with the blocking and the background merge implementation. The time is measured as the RAM flush time plus the merge time, if a merge is triggered (for the blocking merge) or is currently in progress (for the non-blocking merge). With the blocking merge, there is large gap between the maximum and the average insertion time, since the maximum insertion time corresponds to the merge time to create a partition in the sixth index level. With the background merge, this gap is much lower, since the cost of the merge operations is amortized over a large number of insertions/deletions. The insertion/deletion time never exceeds 0.31 s for Kingston and 0.20 s for Silicon Power (see Table 6). Note also that the non-blocking implementation of

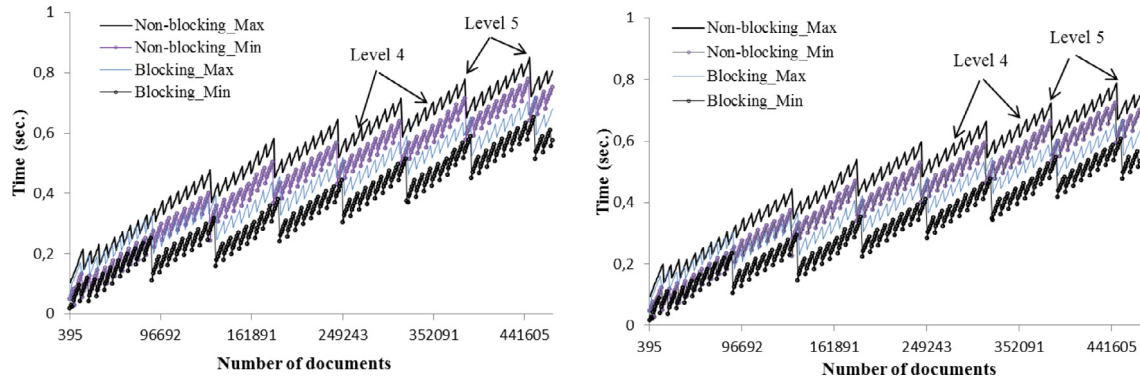


Fig. 11. Query performances with blocking and non-blocking merge with silicon power (left) and Kingston (right) storage for the ENRON dataset.

merges has a slight impact on the query cost since the number of lower level partitions can temporarily exceed the value of b (see next section).

9.3. Index search performance

We evaluated the search performance of the proposed index on our test board and the two SD cards, with both the blocking and non-blocking merge implementations. Because of the similarity of the results, we present in the figures hereafter the results obtained with both storages only with the ENRON dataset. For the other two datasets, we present the results with a single SD card, i.e., Silicon Power. Fig. 11 shows the average query time for the 300 search queries in our test query set, as a function of the number of indexed documents in the ENRON collection. The query set mixes queries consisting of 1 and up to 5 terms (see Table 2). For simplicity, the curves in Fig. 11 present the query cost before (i.e., the “max” curves) and after (i.e., the “min” curves) each merge occurring in the higher index levels, i.e., from the level 3 to the level 5. We can observe that locally, the query cost increases linearly with the number of partitions in the lower levels, and then decreases significantly after every merge operation. The large variations in the query cost correspond to the creation of a new partition in the fifth level of the SSF (see the arrows in Fig. 11), while the intermediary peaks correspond to the creation of a partition in level 4 of the SSF.

Globally, the query time also increases linearly with the number of indexed documents, but the linear increase is very slow. For example, we see that after inserting 500K documents and deleting 10% of them, our search engine is able to answer queries with an average execution time of only 0.45 s and a maximum time of 0.79 s for Kingston and an average of 0.49 s and a maximum of 0.89 s for Silicon Power (with the non-blocking merge implementation). The query times are lower with a blocking merge, i.e., an average of 0.35 s and a maximum time of 0.67 s for Kingston and an average of 0.38 s and a maximum of 0.72 s for Silicon Power. The non-blocking merge leads to an increase of the query cost because the number of lower level partitions can temporarily exceed b , so that more partitions have to be visited. In our setting, the increase is on average of about 28% compared with the query time with a blocking merge. The query time increase is approximately 0.1 s for Kingston and 0.11 s for Silicon Power and appears to be a fair trade-off for applications that cannot accept unpredictable or unbounded update index latencies. Note that the increase of the query cost with a non-blocking merge can be decreased by enlarging the time window allocated to periodically process a merge.

Fig. 12 presents the evolution of the query time with the number of indexed documents for the pseudo-desktop and synthetic datasets. The deletion rate in this figure is 10% as with ENRON.

For better readability of the figures, we present the minimum and the maximum query times only with the non-blocking merge implementation. As with the ENRON dataset, the query times with a blocking merge are about 25% lower than with the non-blocking merge. With the pseudo-desktop dataset, our index exhibits an average execution time of only 0.17 s and a maximum time of 0.32 s for Kingston and an average of 0.18 s and a maximum of 0.35 s for Silicon Power. With the synthetic dataset, the average and the maximum execution times are 0.19 s and 0.39 s respectively for Kingston, while for Silicon Power the average is 0.21 s and the maximum time is 0.42 s. As with the ENRON dataset, globally the query cost increases linearly with the index size. The overall index size is lower for the pseudo-desktop and synthetic datasets compared with ENRON (see Section 9.5), which explains the smaller values of the average query times with these two datasets.

9.4. Index search performance for advanced Keyword-based functionalities

In this section, we present the impact of the advanced keyword search implementation (see Section 7) on the query performance. For the sake of simplicity, we limit the scope of the evaluation to conjunctive logical expressions. To generate queries with logical expressions, we select part of the indexed terms as metadata terms and associate to each query a set of such metadata terms. Regarding the metadata terms, there are two major factors that impact the query performance: the popularity of the term (i.e., number of documents that contain the metadata) and the number of terms in the logical expression. Hence, to test the impact of logical expressions on the search engine performance, we vary the frequency of the involved metadata terms and also the number of metadata terms involved in each logical expression. To be able to compare the results with our previous measures, we consider the same datasets as above, namely the ENRON and the Pseudo-desktop data sets, and the Silicon Power microSD card. In the next experiments, we have fixed the top- k threshold to 10 (i.e., $k = 10$).

9.4.1. Impact of the popularity of the metadata terms on the query performance

To test the impact of the frequency of the metadata terms on the query performance, we select sets of terms appearing with different frequencies in the considered data sets. We build three groups of 100 terms. For Enron, the first group contains 100 terms appearing in the document collections with low frequency (i.e., an average of 27 appearances), the second group contains 100 terms with medium frequency (i.e., an average of 325 appearances) and the third group contains 100 terms with very large frequency (i.e., an average of 43,765 appearances). For Pseudo-desktop, the first group of terms has an average of 11 appearances, the second group

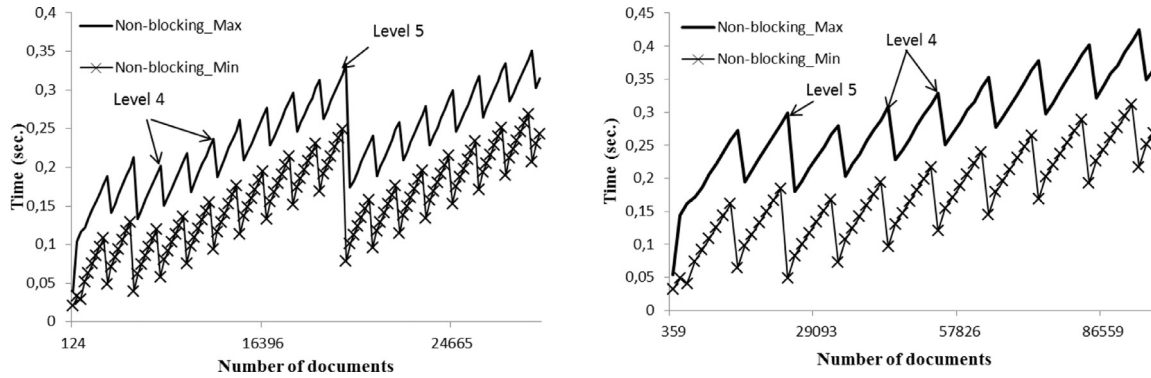


Fig. 12. Query performances with blocking and non-blocking merge with Silicon Power storage for the Pseudo-desktop (left) and the synthetic (right) datasets.

Table 7

Impact of metadata term popularity on the query performance (in seconds) with Silicon Power storage.

	No metadata	Group 1 metadata	Group 2 metadata	Group 3 metadata
ENRON	0,77	0,82	0,82	1,34
Pseudo-desktop	0,27	0,29	0,29	0,43

Table 8

Impact of number of metadata terms on the query performance (in seconds) with Silicon power storage.

	No metadata	Group 1 metadata	Group 2 metadata	Group 3 metadata
ENRON	0,77	1	0,97	0,93
Pseudo-desktop	0,27	0,35	0,34	0,33

an average of 76 appearances and the third group an average of 4964 appearances. We then randomly select 100 queries from the query sets, and we associate each query with one metadata term from each group. We measure the average query time without any metadata term, with a logical expression made of a single term taken from the first group of metadata terms, then taken from the second group and lastly taken from the third group.

Table 7 shows the average query performance after the insertion of the whole data set depending on the query group. We can observe that the impact of the metadata term popularity on the query performance is very low for groups 1 and 2, but higher for group 3. With the ENRON data set, the search engine is able to answer queries with an average execution time of 0,77 s without any metadata, 0,82 s for the first and second groups of metadata terms, and 1,34 s for the third group. With the pseudo-desktop data set, the average execution time of queries without metadata is 0,27 s, for metadata terms in groups 1 and 2 is 0,29 s, and for group 3 is 0,43 s. Globally, the query time of groups 1 and 2 increases of about 8% compared to queries without metadata and about 68% for the queries in group 3. However, as indicated by the statistics of the data sets in Table 2, only 5% of the terms are frequent in the ENRON data set and 6% in the Pseudo-desktop data set. The frequent terms are not the common case in a data set dictionary and we expect to have a similar distribution for the metadata terms. Therefore, the query performance with one metadata term will be marginally impacted in general as demonstrated by the results of the first two groups in Table 7.

9.4.2. Impact of the number of metadata terms on the query performance

To meet the different needs of personal cloud advanced keyword searches, logical expression can contain several metadata terms. To measure the impact of the number of metadata terms on the query performance, we consider logical expressions which contain 3, 5 and 7 terms randomly chosen from the data set. These expressions are then combined with the workloads containing 100 queries for each data set. Table 8 shows the average query time of

the queries without logical expressions, and in the presence of a logical expression involving 3 metadata terms (group 1), 5 metadata terms (group 2) and 7 metadata terms (group 3). The queries are evaluated after inserting the complete respective data set. We can see that the search engine is able to answer queries with an average execution time of 1 s for queries of group 1, 0,97 s for queries of group 2 and 0,93 s for queries of group 3 for the ENRON data set. A similar behavior is observed with the pseudo-desktop data set, where the average execution time of queries is 0,35 s for queries of group1, 0,34 s for group 2 and 0,33 s for group 3.

These experimental results show that the query performance is only marginally impacted by the number of terms in logical expression with a maximum increase of 30% compared to basic queries. The better performance for the queries in group 3 compared to the queries in group 1 and 2 can be explained by our evaluation strategy applied to conjunctive expressions. Indeed, we evaluate the logical expression by starting with the least frequent access term, which avoids accessing the subsequent access terms if the condition on the current term is not satisfied. Thus, many inverted lists corresponding to the most frequent access terms do not need to be accessed, which happens with a higher probability if the expression has more terms.

9.5. Index performance with various deletion rates

In this section we discuss the impact of the deletion rate on the index search performance and index size. Since the deletions are executed as insertions in our search engine, the performance of delete operations is the same with the insert performance. However, the deleted documents are temporarily stored in the index structure (i.e., until the deletions are absorbed during the index merges), which leads to an increase of the index size compared to the optimal index size (i.e., where deleted entries are directly purged). The increased index size can degrade the query performance. Nevertheless, the experimental results show that the query performance is only marginally impacted by deletions even for high deletion rates.

Table 9

Average query performance (in seconds) with different deletion rates and Kingston storage.

	0%	10%	30%	50%
Avg. query time (ENRON: 500k docs)	0.49	0.45	0.41	0.37
Avg. query time (ENRON: 250k docs)	0.33	0.34	0.35	0.37
Avg. query time (Desktop: 27k docs)	0.18	0.17	0.16	0.15
Avg. query time (Desktop: 13k docs)	0.12	0.13	0.15	0.16
Avg. query time (Synthetic: 100k docs)	0.13	0.13	0.12	0.11
Avg. query time (Synthetic: 50k docs)	0.10	0.10	0.11	0.11

Table 9 shows the average query performance for different deletion rates with the Kingston storage (we obtained similar results with the SP storage). Here, we considered two cases. First, we inserted the whole dataset while deleting a number of documents corresponding to the deletion rate (lines 1, 3 and 5 in Table 9). In this case, the higher the deletion rate is, the lower the query time is since a good part of the deleted documents (app. 50%) will be purged from the index and decrease the query processing time. In the second case, the total number of active documents in the index is the same regardless the deletion rate (lines 2, 4 and 6 in Table 9). Hence, the higher the deletion rate is, the more documents we insert to compensate the deletions. In this case, a higher deletion rate leads to larger query times since part of the deletions are present in the index and have to be processed by the queries. However, the increase of the query times is relatively small compared to the case with no deletions, i.e., less than 12% for deletion rates up to 50%. Globally, the index is robust with the number of deletions in both cases.

Table 10 shows the index size for the three datasets after the insertion of all the documents in the collection and the uniform deletion of a certain percentage of the indexed documents. In each table cell, the first number indicates the cumulated size in MB of all the *IL* parts of the SSF (i.e., the global size of the inverted lists), while the second number gives the cumulated size of all the *IS* parts of the SSF (i.e., the global size of the search structures). Also, we give in Table 10 for each dataset and deletion rate the index size of the classical inverted index used as reference. Without deletions, the SSF index size is practically the same with the inverted index size. The SSF requires a little bit more storage for the *IS* since each partition has its own search structure to index the terms in the partition. Nevertheless, the storage overhead of the SSF is negligible since the search structure represents less than 1% of the global index size (i.e., the inverted lists require much more storage than the search structures). With deletions, the size of the SSF index is larger than the size of the inverted index. Also, the size difference increases with the delete ratio. The explanation is that the deleted documents are reinserted in the SSF, which temporarily increase the index size. However, when a merge is triggered in an index level, a part of the deletions are absorbed and the deleted documents are purged from the index. Therefore, at any given time only a part of the deleted documents are still present in the index. Typically, in our tests we observed that about 45% to 55% of deletions are not absorbed after a high number of document insertions and deletions. This makes that the SSF index size to be at most 40% larger than the inverted index size even for high deletion rates, which we believe is quite acceptable.

9.6. Comparison with the state-of-the-art search engine methods

The existing solutions face important limitations when used to index large collections of documents in secure tokens. The classical inverted index [42] was designed for magnetic disks and therefore does not comply with the flash storage constraints. Besides, the few proposed methods that consider the constraints of secure

tokens process the insertions efficiently, but do not scale with large datasets and cannot support index updates.

In this section we compare our proposed search engine (called SSF here) with the representative indexing method of each of the aforementioned approaches. Hence, we choose the typical inverted index to represent the query-optimized index family (see Fig. 2) and Microsearch [33] for the insert-optimized index family. Note that the other embedded search engines presented in Section 2.3 rely on similar index structures with Microsearch. We used the same test conditions as above for two competing methods, i.e., by using a *RAM_Bound* equal to 5KB. The data insertions in the inverted index are processed in a similar way as in our search engine. The insertions are first buffered in RAM until the *RAM_Bound* is reached. Then, the buffered updates are applied in batch to the index structure. However, different from our approach, the index structure is modified in-place, which requires costly random writes. Also, to be able to evaluate the queries under the RAM constraint with the inverted index, the inverted lists of this structure have to be maintained sorted on the document ids, which permits applying a linear pipeline query processing similar to the SSF. In the case of Microsearch [33], we used a hash function with 8 buckets, since this value leads to the most balanced query-insert performance given the 5KB of RAM. Besides, we only considered data insertions and queries in the tests below, since Microsearch does not support deletions. We note also that in [33] Microsearch was implemented using NOR Flash storage, which allows the elements of the linked lists to have different sizes. In our case, the elements of the lists have always the size of a sector (i.e., 512 bytes) but this modification has a limited impact on the global performance of the method.

9.6.1. Insertion performance

Figs. 13 and 14 show the average insertion time for the three methods (i.e., SSF, Microsearch and the Inverted Index) with the three datasets. Both Microsearch and SSF exhibit good insert performance. On average, for ENRON, a document insertion in Microsearch takes about 0.094 s with Kingston and 0.059 s with Silicon Power, and 0.14 s with Kingston and 0.09 s with Silicon Power in SSF. In comparison, the document insertion time in the Inverted Index is much larger because of the costly random writes in Flash memory. On average, an insertion requires 30 s with Kingston and 7.6 s with Silicon Power, which is nearly two orders of magnitude higher than with the embedded search engines, and clearly dismisses this method in the context of secure tokens. We obtained similar results with the two other datasets as presented in Fig. 14. On average, with Silicon Power storage, for the synthetic collection, a document insertion in Microsearch takes about 0.02, 0.07 s in SSF and 15 s in the inverted index. For the pseudo-desktop collection, a document insertion in Microsearch takes about 0.08 s, 0.33 s in SSF and 30 s in the inverted index. For all the methods, the insertion times are proportional to document size. Therefore, the insertion times are the highest with the pseudo-desktop collection.

The insertion in the SSF and Microsearch relies on sequential writes in Flash to comply with the Flash memory constraints. The higher insertion cost of the SSF compared to Microsearch is generated by the SSF merges. Although the SSF insertions are less efficient than with Microsearch, the insertion time remains reasonable and will probably satisfy most of the applications especially if they require indexing large collections of documents. Indeed, the slight increase of the insert cost is outweighed by the query performance and scalability of the SSF.

9.6.2. Query performance

Figs. 15 and 16 show the query execution time for the three methods in function of the number of indexed documents. The Inverted Index has the best query times and can be considered as the

Table 10
Index (inverted lists/search structures) size (MB) with different deletion rates (from 0% to 50%).

	0%	10%	30%	50%
SSF I.L./L.S size (ENRON: 500k docs)	439 / 3.5	402 / 3	350 / 2.4	310 / 2.2
Inverted index I.L./L.S size (ENRON: 500k docs)	439 / 0.6	397 / 0.6	305 / 0.6	220 / 0.6
SSF I.L./L.S size (Desktop: 27k docs)	81 / 1.24	76 / 1.14	60 / 0.82	55 / 0.73
Inverted index I.L./L.S size (Desktop: 27k docs)	81 / 0.4	73 / 0.4	57 / 0.4	40 / 0.4
SSF I.L./L.S size (Synthetic: 100k docs)	78 / 0.97	74 / 0.88	66 / 0.78	58 / 0.69
Inverted index I.L./L.S size (Synthetic: 100k docs)	78 / 0.13	70 / 0.13	55 / 0.13	40 / 0.13

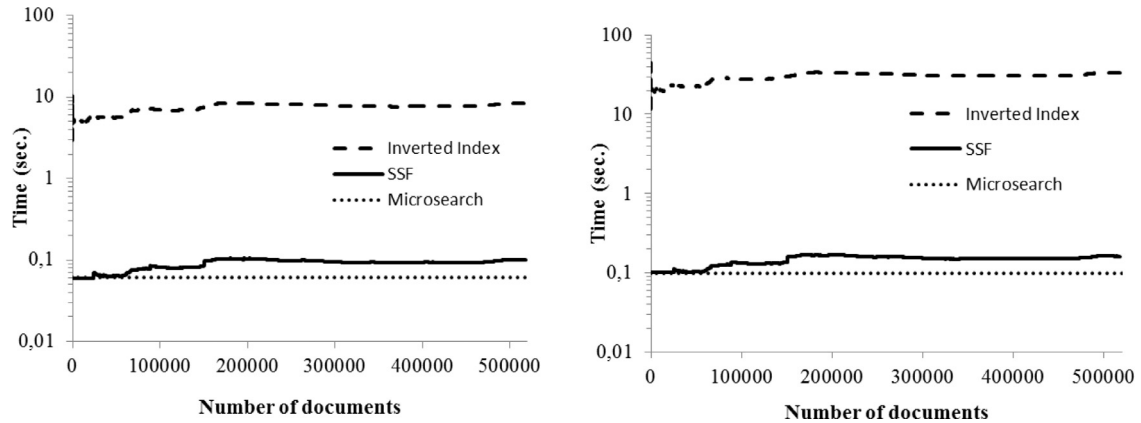


Fig. 13. Average document insertion times of Microsearch, SSF and the Inverted Index with Silicon Power (left) and Kingston (right) storage for the ENRON dataset.

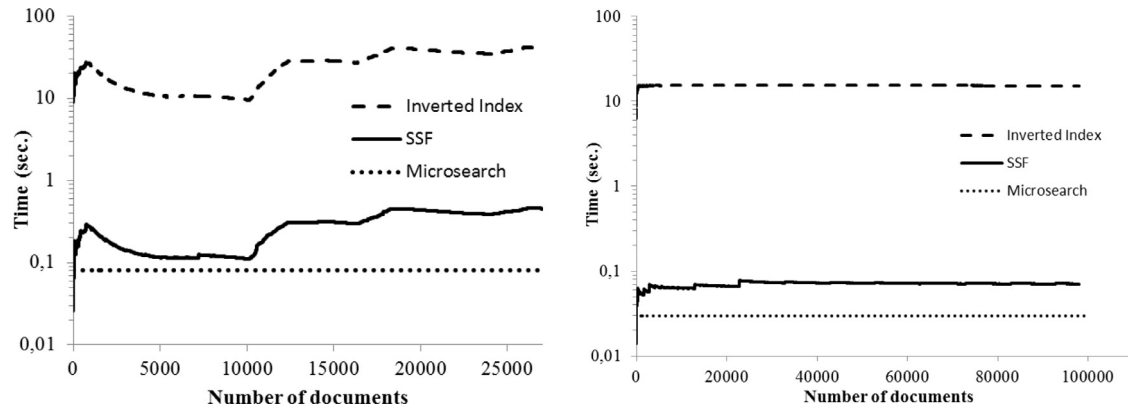


Fig. 14. Average document insertion times of Microsearch, SSF and the Inverted Index with Silicon Power storage for the pseudo-desktop (left) and synthetic (right) datasets.

most efficient structure for processing full-text search queries. We can see that query times of the SSF are very close to the Inverted Index times. On average, for ENRON, it takes 0.49 s with Kingston and 0.53 s of Silicon Power to process a query in the SSF, while for the Inverted Index it takes 0.16 s with Kingston and 0.17 s with Silicon Power. Also, the average query times with Silicon Power are 0.18 s and 0.05 s in the SSF, 0.07 s and 0.02 s in the inverted index, and 880 s and 355 s in Microsearch, for the pseudo-desktop and the synthetic respectively. The difference in performance between the SSF and the Inverted Index is explained by the fragmentation of the index structure of the SSF. However, the index partitioning in the SSF is largely outweighed when we take into account the insert performance of these two index structures.

Microsearch has the worse query performance, which clearly is not scalable to a large number of documents. On average, for ENRON, it takes 1728.80 s (28 min) with Kingston and 1861.78 s (31 min) with Silicon Power to process a query with Microsearch. Given the very large query times, we measured the real query time only up to 10,000 indexed documents and estimated the query times above this number of documents, which is fairly simple since the query time linearly increases with the number of

documents in Microsearch. Note also that even for a low number of documents, the query times of Microsearch are larger than the query times of SSF. For instance, for 1000 documents it takes 3.1 s with Kingston in Microsearch and 0.1 s in SSF. The first reason is that in Microsearch an inverted list corresponds to a large number of terms (i.e., all the terms are distributed in the 8 hash buckets). Therefore, a large part of the index data is retrieved by the query. Second, Microsearch requires two passes over the chained list containing a query term. The first pass is done to compute the global F_t value of the term, while the $tf-idf$ score of the documents containing the term is computed only in the second pass.

9.6.3. Overall performance

The tiny RAM and the NAND Flash storage of sensors introduce conflicting constraints on the index structure. Fig. 17 summarizes the update and the query performance of proposed index structure and the two representative competitors. Our solution is the only one to offer both query and update scalability under these constraints by balancing the query and the update costs for any kind of document collection. At the same time, the loss in both the query and the update performance remains reasonable compared

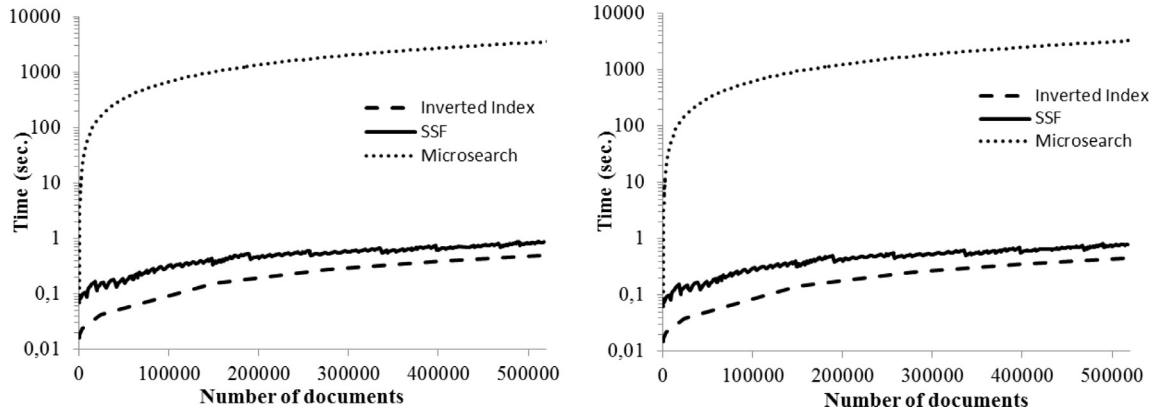


Fig. 15. Query execution times with the Inverted Index, SSF and Microsearch with Silicon Power (left) and Kingston (right) storage on the ENRON dataset.

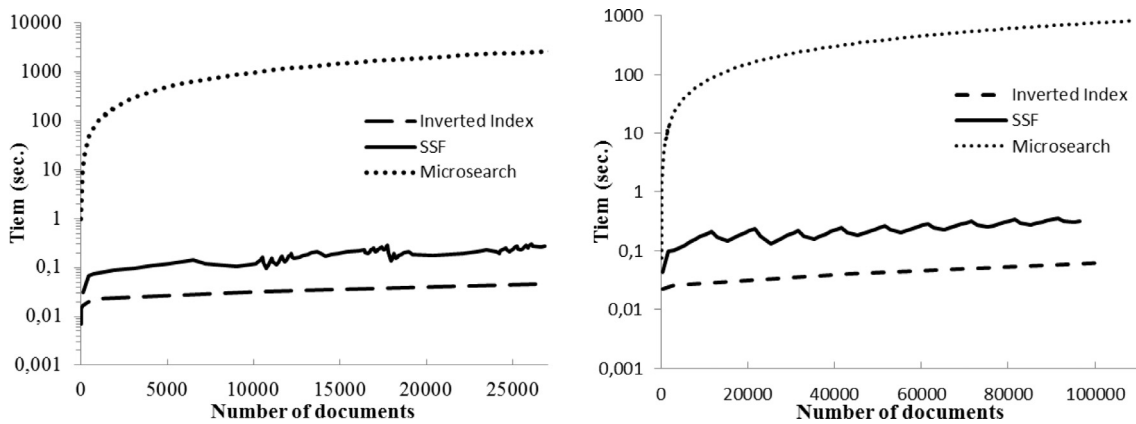


Fig. 16. Query execution times with the Inverted Index, SSF and Microsearch with Silicon Power storage for the pseudo-desktop (left) and synthetic (right) datasets.

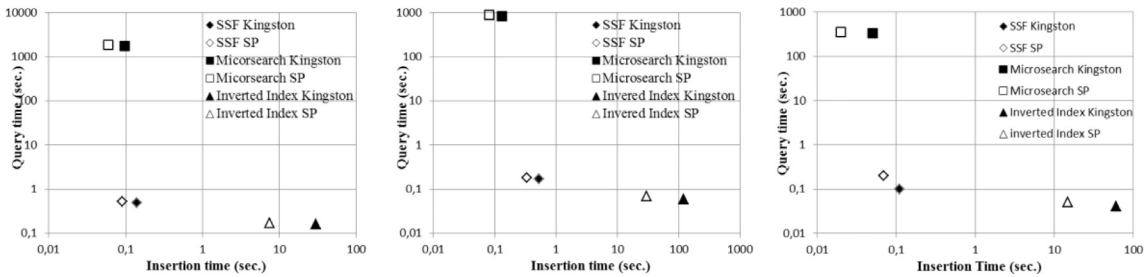


Fig. 17. Overall performance comparison for ENRON (left), pseudo-desktop (middle) and synthetic (right) datasets.

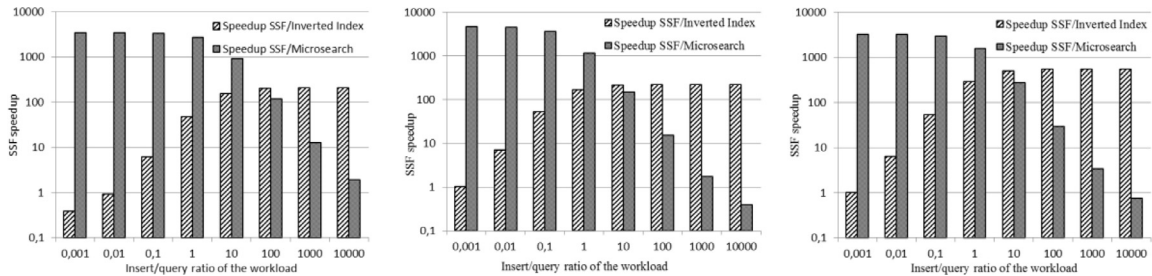


Fig. 18. Overall speedup comparison for ENRON (left), pseudo-desktop (middle) and synthetic (right) datasets with Kingston storage.

with the query optimized index (i.e., the Inverted Index) and the insert optimized index (i.e., Microsearch). Fig. 18 shows the speedup of SSF (i.e., the ratio between the throughput of SSF and of the competitors) with Kingston for workloads containing insertions and queries in different ratios. In most cases, SSF has (much) better throughput with both insert- and query-oriented workloads, while being the sole versatile method. Practically, SSF will be the

preferred index method unless the expected workload contains in an overwhelming proportion either insertions or queries.

A straightforward way to improve the efficiency of all the tested methods would be to increase the I/O granularity, e.g., passing from 512 bytes I/Os to 2 Kbytes I/Os. Specifically, the insertion and query performance of all the methods is expected to improve in this case by taking advantage of the throughput increase of

the storage device (see Table 1). However, this leads to increased RAM consumption as discussed in Section 8.2, a thorny problem for secure tokens. Note that by increasing the I/O granularity, the relative performance of the tested methods is expected to remain approximately the same, i.e., we expect to have the similar speedup values of the SSF compared with Microsearch or the inverted index since the throughput augments approximately in the same proportion for all types of I/Os.

9.7. Discussion

In the light of the above presented experimental results, we discuss in this section the limitations of the proposed approach. As above, we mainly analyze the limitations of the SSF in comparison with the classical inverted index, which can be considered as the ideal structure to index text documents at least from the query performance and index size points of view. Hence, the shortcomings of the SSF originate from the major differences between the SSF and the inverted index structures, i.e., the index partitioning and the deletion processing.

The partitioning of the SSF influences both the query and the insert performance. The query performance is degraded because of the multiple searches in several small *IS* (i.e., the search index dictionary of each partition) are more costly than a single search in a large *IS*. However, given the exponentially increasing size of the partitions in the SSF levels, the loss in the query performance is limited compared with the inverted index (see Figs. 15 and 16). Also, the query performance loss is diminishing with the increase of the number of indexed documents, suggesting that our approach is particularly appropriated for indexing large datasets. We should also note that the partitioning introduces some variability in the query cost (see the stairway-like curves in Figs. 11 and 12). The partitioning has an impact on the insert performance since already inserted documents have to be rewritten at an index merge. Yet, the insertions only generate sequential writes in Flash and the insert cost is much more scalable than for the inverted index (see Figs. 13 and 14). Finally, the partitioning has also an impact on the index size since the search structure that indexes the terms in each partition is redundant. However, the increase in the index size is negligible (see Table 10) as the search structures only take a small fraction from the global index size.

The specific way of processing deletions in the SSF also has an impact on the query performance and the index size. Since a deleted document is first reinserted in the index, deletions lead to a temporal increase of the index size and consequently, of the query cost. Nevertheless, this negative effect is limited by the merge operations that permit to purge some of the deleted documents. The experimental results show that, even for high deletion rates of up to 50%, the increase in the index size is lower than 40%, while the increase in the query cost is lower than 12% (see Tables 9 and 10).

Finally, another limitation of the proposed search engine is that we do not consider the problem of concurrent access, i.e., multiple processes that query/update the index at the same time. In our workloads combining queries/inserts/deletes, we analyze the cost of each separately, one operation at a time. Indeed secure tokens, contrary to central servers, rarely support parallel or multi-task processing. Moreover, the RAM consumption increases linearly with the number of operation executed in parallel, a serious constraint in our context.

10. Conclusion

In this paper, we present the design of an embedded search engine for secure tokens equipped with low RAM and large Flash

storage capacity. The proposed method contributes to the development of the Personal Cloud paradigm by allowing users to securely store, query and share their document collections. In addition, this work contributes to the current trend to endow smart objects with more and more powerful data management techniques. Our proposal is founded on three design principles, which are combined to produce an embedded search engine reconciling high insert/delete rate and query scalability for very large datasets. By satisfying a Bounded RAM agreement, our search engine can accommodate a wide population of secure tokens, including those having only a few KBs of RAM. Satisfying this agreement is also a mean to fulfill co-design perspectives, i.e., calibrating a new hardware platform with the hardware resources strictly necessary to meet a given performance requirement. The proposed search engine has been implemented on a hardware platform having a configuration representative of secure tokens. The experimental evaluation validates its efficiency and scalability over three real and synthetic large datasets representative of different smart object scenarios and also demonstrates the superiority of this approach compared to state of the art methods. Finally, we show that, at the price of a direct extension, our search engine can cope with conditional top-*k* queries, broadening its application scope.

The next step is to study how our three design principles can be generalized for building other kinds of embedded query engines (e.g., NoSQL like), considering that indexing any form of data streams in mass storage smart objects will encounter similar hardware constraints and then lead to similar requirements. Also, in the context of smart objects the energy efficiency of the search engine can be important (e.g., for battery powered smart objects). Another direction of work is to study the energy consumption of our approach in comparison with other existing approaches.

Acknowledgments

This work was partially supported by the ANR KISS project [grant no. ANR-11-INSE-0005] and ANR PerSoCloud project [grant no. ANR-16-CE39-0014]. The authors thank the anonymous reviewers for their insightful comments that helped to significantly improve the presentation of the paper.

References

- [1] C.C. Aggarwal, N. Ashish, A. Sheth, The internet of things: a survey from the data-centric perspective, in: *Managing and Mining Sensor Data*, Springer US, 2013, pp. 383–428.
- [2] D. Agrawal, D. Ganesan, R. Sitaraman, Y. Diao, S. Singh, Lazy-adaptive tree: an optimized index structure for flash devices, *Proc. VLDB 2* (1) (2009) 361–372.
- [3] D. Agrawal, B. Li, Z. Cao, D. Ganesan, Y. Diao, P.J. Shenoy, Exploiting the interplay between memory and flash storage in embedded sensor devices, in: *RTCSA*, 2010, 2010, pp. 227–236.
- [4] N. Ancaux, P. Bonnet, L. Bouganim, B. Nguyen, I. Sandu Popa, P. Pucheral, Trusted cells: a sea change for personal data services, *CIDR*, 2013.
- [5] N. Ancaux, L. Bouganim, P. Pucheral, Y. Guo, L.L. Folgoc, S. Yin, Milo-db: a personal, secure and portable database machine, in: *Distributed and Parallel Databases*, 2014, pp. 37–63.
- [6] N. Ancaux, S. Lallali, I. Sandu Popa, P. Pucheral, A scalable search engine for mass storage smart objects, *PVLDB 8* (9) (2015) 910–921.
- [7] B. André, N. Ancaux, P. Pucheral, P. Tran Van, A Root of Trust for the Personal Cloud, 2016, 2016 ERCIM News 106.
- [8] M. Athanassoulis, M.S. Kester, L.M. Maas, R. Stoica, S. Idreos, A. Ailamaki, M. Callaghan, Designing access methods: the RUM conjecture, in: *EDBT*, 2016, 2016, pp. 461–466.
- [9] C.-m. Au Yeung, L. Kagal, N. Gibbins, N. Shadbolt, Providing access control to online photo albums based on tags and linked data, in: *AAAI Spring Symposium on Social Semantic Web: Where Web 2.0 Meets Web 3*, 2009, pp. 9–14. conf A*, 33 citations.
- [10] P.A. Bernstein, C.W. Reid, S. Das, Hyder - a transactional record manager for shared flash, in: *CIDR*, 2011, pp. 9–20.
- [11] E. Bertino, Data security and privacy in the IoT, in: *EDBT*, 2016, 2016, pp. 1–3.
- [12] M. Bjorling, P. Bonnet, L. Bouganim, B.T. Jonsson, uflip: Understanding the energy consumption of flash devices, *IEEE Data Eng. Bull.* 33 (4) (2010) 48–54.
- [13] B. Debnath, S. Sengupta, J. Li, Skimpystash: ram space skimpy key-value store on flash-based storage, in: *Proceeding of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, 2011, pp. 25–36.

- [14] Y. Diao, D. Ganesan, G. Mathur, P.J. Shenoy, Rethinking data management for storage-centric sensor networks, in: CIDR, 2007, pp. 22–31.
- [15] J.H. Ehlers, S.A. Jassim, Wavelet library for constrained devices, in: Proc. SPIE 6579, Mobile Multimedia/Image Processing for Military and Security Applications 2007, 2007, p. 65790P, doi:10.1117/12.720713. (May 02, 2007).
- [16] Y.-M. Huang, Y.-X. Lai, Distributed energy management system within residential sensor-based heterogeneous network structure, *Wireless Sens. Netw. Ecol. Monit.* 3 (2013) 35–60.
- [17] H.V. Jagadish, P.P.S. Narayan, S. Seshadri, S. Sudarshan, R. Kanneganti, Incremental organization for data recording and warehousing, *Proc. VLDB* (1997) 16–25.
- [18] C. Jermaine, A. Datta, E. Omiecinski, A novel index supporting high volume data warehouse insertion, *Proc. VLDB* (1999) 235–246.
- [19] C. Jermaine, E. Omiecinski, W.G. Yee, The partitioned exponential file for database storage management, *VLDB J* 16 (4) (2007) 417–437.
- [20] J.Y. Kim, W.B. Croft, Retrieval experiments using pseudo-desktop collections, *CIKM*, 2009.
- [21] P. Klemperer, Y. Liang, M. Mazurek, M. Sleeper, B. Ur, L. Bauer, L.F. Cranor, N. Gupta, M. Reiter, Tag, you can see it!: using tags for access control in photo sharing, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'12), 2012, pp. 377–386. conf A*, 61 citations.
- [22] S. Lallali, N. Anciaux, I. Sandu Popa, P. Pucheral, A secure search engine for the personal cloud, in: SIGMOD Conference, 2015, pp. 1445–1450.
- [23] T. Le, N. Anciaux, S. Guilloton, S. Lallali, P. Pucheral, I. Sandu Popa, C. Chen, Distributed secure search in the personal cloud, in: EDBT, 2016, pp. 652–655.
- [24] Y. Li, B. He, R.J. Yang, Q. Luo, K. Yi, Tree indexing on solid state drives, *VLDB J* (2010) 3 (2010) 1195–1206.
- [25] T. Li, Y. Liu, Y. Tian, S. Shen, W. Mao, A storage solution for massive iot data based on nosql, in: Green Computing and Communications (GreenCom), 2012 IEEE International Conference on, November, 2012, pp. 50–57.
- [26] H. Lim, B. Fan, D.G. Andersen, M. Kaminsky, Silt: a memory-efficient, high-performance key-value store, in: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, SOSP '11, ACM, New York, NY, USA, 2011, pp. 1–13.
- [27] M.L. Mazurek, J.P. Arsenault, J. Bresee, N. Gupta, I. Ion, C. Johns, D. Lee, Y. Liang, J. Olsen, B. Salmon, R. Shay, K. Vaniea, L. Bauer, L.F. Cranor, G.R. Ganger, M.K. Reiter, Access control for home data sharing: attitudes, needs and practices, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10), 2010, pp. 645–654. conf A*, 87 citations.
- [28] M.L. Mazurek, P.F. Klemperer, R. Shay, H. Takabi, L. Bauer, L.F. Cranor, Exploring reactive access control, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11), 2011, pp. 2085–2094. cité 22 fois, A*.
- [29] M.L. Mazurek, Y. Liang, W. Melicher, M. Sleeper, L. Bauer, G.R. Ganger, N. Gupta, M.K. Reiter, Toward strong, usable access control for shared distributed data, in: Proceedings of the 12th USENIX conference on File and Storage Technologies, FAST'14, 2014, pp. 89–103. conf A, 8 citations.
- [30] P.E. O'Neil, E. Cheng, D. Gawlick, E.J. O'Neil, The Log-Structured Merge-Tree (LSM-Tree), *Acta Inf.* 33 (4) (1996) 351–385.
- [31] R. Sears, R. Ramakrishnan, bLSM: a general purpose log structured merge tree, in: SIGMOD Conference, 2012, pp. 217–228.
- [32] C.C. Tan, B. Sheng, H. Wang, Q. Li, Microsearch: when search engines meet small devices, in: Pervasive Computing, 2008, pp. 93–110.
- [33] C.C. Tan, B. Sheng, H. Wang, Q. Li, Microsearch: a search engine for embedded devices used in pervasive computing, *ACM Trans. Embed. Comput. Syst.* 9 (4) (2010) 1–29 43.
- [34] Q.-C. To, B. Nguyen, P. Pucheral, Privacy-preserving query execution using a decentralized architecture and tamper resistant hardware, in: EDBT, 2014, pp. 487–498.
- [35] N. Tsiftes, A. Dunkels, A database in every sensor, in: Proc. of the 9th ACM Conf. on Embedded Networked Sensor Systems, SenSys'11, 2011, pp. 316–332.
- [36] H.T. Vo, S. Wang, D. Agrawal, G. Chen, B.C. Ooi, Logbase: a scalable log-structured database system in the cloud, *Proc. VLDB Endowment* 5 (10) (2012) 1004–1015.
- [37] B. Wang, J.S. Baras, Hybridstore: an efficient data management system for hybrid flash-based sensor devices, in: Proceedings of the 10th European Conference on Wireless Sensor Networks, EWSN'13, Berlin, Heidelberg, Springer-Verlag, 2013, pp. 50–66.
- [38] H. Wang, C.C. Tan, Q. Li, Snoogle: a search engine for pervasive environments, *IEEE Trans. Parallel Distrib. Syst.* 21 (8) (2010) 1188–1202.
- [39] C.-H. Wu, T.-W. Kuo, L.-P. Chang, An efficient b-tree layer implementation for flash-memory storage systems, *ACM Trans. Embed. Comput. Syst.* 6 (3) (2007).
- [40] T. Yan, D. Ganesan, R. Manmatha, Distributed image search in camera sensor networks, in: Proc. of the 6th ACM Conference on Embedded Network Sensor Systems, SenSys'08, 2008, pp. 155–168.
- [41] K.-K. Yap, V. Srinivasan, M. Motani, Max: wide area human-centric search of the physical world, *ACM Trans. Sens. Netw.* 4 (4) (2008) 1–34 26.
- [42] J. Zobel, A. Moffat, Inverted files for text search engines, *ACM Comput. Surv.* 38 (2) (2006).

Appendix F

SEP2P: Secure and Efficient P2P Personal Data Processing

SEP2P: Secure and Efficient P2P Personal Data Processing

Julien Loudet^{1,2,3}
¹Cozy Cloud, France
julien@cozycloud.cc

Iulian Sandu-Popa^{3,2}
²INRIA Saclay, France
<fname.lname>@inria.fr

Luc Bouganim^{2,3}
³University of Versailles, France
<fname.lname>@uvsq.fr

ABSTRACT

Personal Data Management Systems are flourishing allowing an individual to integrate all her personal data in a single place and use it for her benefit and for the benefit of the community. This leads to a significant paradigm shift since personal data become massively distributed. In this context, an important issue needed to be addressed is: how can users/applications execute queries and computations over this massively distributed data in a secure and efficient way, relying exclusively on peer-to-peer (P2P) interactions? In this paper, we motivate and study the feasibility of such a pure P2P personal data management system and provide efficient and scalable mechanisms to reduce the data leakage to its minimum with covert adversaries. In particular, we show that data processing tasks can be assigned to nodes in a verifiable random way, which cannot be influenced by malicious colluding nodes. Then, we propose a generic solution which largely minimizes the verification cost. Our experimental evaluation shows that the proposed protocols lead to minimal private information leakage, while the cost of the security mechanisms remains very low even with a large number of colluding corrupted nodes. Finally, we illustrate our generic protocol proposal on three data-oriented use-cases, namely, participatory sensing, targeted data diffusion and more general distributed aggregative queries.

1 INTRODUCTION

The time of individualized management and control over one's personal data is upon us. Thanks to smart disclosure initiatives (e.g., BlueButton [9] and GreenButton in US, MesInfos [16] in France, Midata [25] in UK) and new regulations (e.g., the Europe's new General Data Protection Regulation [27]), users can access their personal data from the companies or government agencies that collected them. Concurrently, Personal Data Management System (PDMS) solutions are flourishing [4] both in the academic (e.g., Personal Data Servers [1], Personal Information Management Systems, Personal Data Stores [14], Personal Clouds [20]) and industry [12, 26, 33]. Their goal is to offer a data platform allowing users to easily store and manage into a single place data directly generated by user devices (e.g., quantified-self data, smart home data, photos, etc.) and data resulting from user interactions (e.g., user preferences, social interaction data, health, bank, telecom, etc.). Users can then leverage the power of their PDMS to benefit from their personal data for their own good and in the interest of the community. Thus, the PDMS paradigm holds the promise of unlocking new innovative usages.

Let us consider three emblematic distributed applications based on large user communities which could greatly benefit from the PDMS paradigm: (1) mobile participatory sensing apps [36], in which mobile users produce sensed geo-localized data (e.g., traffic,

air quality, noise, health conditions) to compute spatially aggregated statistics benefiting the whole community; (2) subscription-based or profile-based data diffusion apps [38], in which PDMS users provide preferences or exhibit profiles in order to selectively receive pertinent information; and (3) distributed query processing over the personal data of large sets of individuals [37], in which users contribute with their personal data and issue queries over the globally contributed data (e.g., computing recommendations, participative studies).

However, these exciting perspectives should not eclipse the security issues raised by the PDMS paradigm. Indeed, each PDMS can store potentially the entire digital life of its owner, thereby proportionally increasing the impact of a leakage. Hence, centralizing all users' data into powerful servers is risky since these data servers become highly desirable targets for attackers: huge amounts of personal data belonging to millions of individuals could be leaked or lost as illustrated by the recent massive attacks (e.g., Facebook, Yahoo or Equifax). Besides, such a centralized solution makes little sense in the PDMS context in which data is naturally distributed at the users' side [19].

Alternatively, recent works [4, 14, 20, 33] propose to let the user data distributed on personal trustworthy platforms under users' control. Such platforms can be built thanks to the combination of (1) a Trusted Execution Environment (TEE) (i.e., secure hardware such as smart cards [1] or secure micro-controllers [4, 5, 20], ARM TrustZone [18], or Intel SGX [29]) and (2) specific software (e.g., minimal Trusted Computing Base and information flow control [22, 29]). In this paper, we follow this approach and consider that a PDMS is a dedicated personal device that the user possesses and is secured thanks to TEE hardware.

In addition, as in many academic and commercial approaches [33], we assume that the PDMS personal device offers a rather good connectivity and availability like, for instance, home-cloud solutions [4, 12, 26, 33] (e.g., a set-top box or a plug computer [4]). Thus, PDMSs can establish peer-to-peer (P2P) connections with other PDMSs, and can be used as data processor in order to provide part of the processing required in distributed applications. Hence, our objective is to study solutions based on a full distribution of PDMSs (called nodes interchangeably) which can act as data sources and data processors and communicate in a peer-to-peer fashion. We discard solutions requiring recentralizing the distributed personal data during its processing, since this would dynamically create a personal data concentration leading to a similar risk as with centralized servers.

Incorporating TEEs greatly increases the protection against malicious PDMS owners. However, since no security measure can be considered as unbreakable, we cannot exclude having some corrupted nodes in the system and, even worse, those corrupted nodes can collude and might very well be undistinguishable from honest nodes, acting as covert adversaries [7]. Also, since data processing relies exclusively on PDMS nodes, and given the very high scale of the distribution which disqualifies secure multi-party computation (MPC) protocols [31], sensitive data leaks are unavoidable in the presence of corrupted nodes, i.e., some data

might be disclosed whenever a corrupted node is selected as a data processor.

The goal of this paper is to assess the feasibility of building a secure and efficient data processing system over a fully distributed network of PDMS housing covert adversaries. To achieve it we provide mechanisms to reduce the data leakage to its minimum, and make the following contributions:

(1) We propose a P2P architecture of PDMSs, called SEP2P (for Secure and Efficient P2P), based on classical Distributed Hash Tables (DHT) and analyze potential data leakages of data sources and data processors. We show that (i) data tasks should be assigned to nodes in a *verifiable random* way, i.e., the assignment cannot be influenced by malicious colluding nodes; and (ii) any data-oriented task, whether it is storage or computation, should be *atomic*, i.e., reduced to a maximum such that it minimizes the quantity of sensitive data accessible by the task.

(2) We focus on the verifiable random assignment problem and propose a generic solution (i.e., independent of the distributed computation tasks) which largely minimizes the verification cost (e.g., 8 asymmetric crypto-operations with a SEP2P network of 1M nodes of which 10K are colluding corrupted nodes).

(3) We experimentally evaluate the quality and efficiency of the proposed protocols. The verifiable random assignment protocol leads to minimal private information leakage, i.e., linear with the number of corrupted nodes, while the cost of the security mechanisms remains very low even with a large number of colluding corrupted nodes.

(4) We address the task atomicity subproblem by providing sketches of solutions for the three classes of applications indicated above. We do not propose full solutions since task atomicity is dependent on the considered class of distributed computation and as such needs to be studied in detail.

Sections 2 to 5 present these four contributions respectively. We finally discuss the related work in Section 6 and conclude the paper in Section 7.

2 SEP2P ARCHITECTURAL DESIGN

2.1 Base System Architecture

SEP2P is a peer-to-peer system and only relies on the PDMS nodes to enable the aforementioned applications. Consequently, each node may play several roles for SEP2P applications:

Node role 1. Each node is a potential **data source**. For instance, producing sensed geo-localized data about the local traffic speed, or sharing grades used to compute recommendations.

Node role 2. Given the fully-decentralized nature of SEP2P, each node is a potential **data processor**, also called **actor**, providing part of the required processing.

Node role 3. The initiator of a distributed processing is called the **triggering node** (T). T could be any node with participatory sensing applications, or the query issuer in distributed query or data diffusion applications.

2.2 Efficient P2P Data Processing

Relying on a fully-distributed system induces several problems, e.g., integrating new nodes, maintaining a coherent global state, making nodes that do not know each other interact, handling churn, maintaining some metadata. It thus requires a communication overlay allowing for efficient node discovery, data indexing

and search. Fortunately, these problems have already been extensively studied in the literature and the *Distributed Hash Tables* (DHTs) appear to be the solution reaching consensus.

Background 1. A **distributed hash table (DHT)** [23, 30, 34] in a P2P network offers an optimized solution to the problem of locating the node(s) storing a specific data item. The DHT offers a basic interface allowing any node of the network to store data, i.e., $store(key, value)$, or to search for certain data, i.e., $lookup(key) \rightarrow value$. DHTs proposals [23, 30, 34] share the concepts of keyspace or **DHT virtual space** (e.g., a 224 bits string obtained by hashing the key or the node ID), space partitioning (mapping space partitions to nodes, using generally a distance function), and overlay network (set of routing tables and strategies allowing reaching a node, given its node ID). For instance, the virtual space is represented as a multi-dimensional space in CAN [30], as a ring in Chord [34] or as a binary tree in Kademlia [23] and is uniformly divided among the nodes in the network. Thus, each node is responsible for the indexing of all the $(key, value)$ pairs where the key falls in the subspace it manages. Both the data storage and the lookup operations are thus fully distributed in a DHT. DHTs have interesting properties: uniform repartition of the data, scalability, fault tolerance and do not require any central coordination.

Hence, SEP2P leverages the classical DHT techniques as a basis for communication efficiency and scalability.

2.3 Security Considerations

In this paper, we use the terminology of ARM [35] to designate the three attack levels on a PDMS node, i.e., *hack*, *shack* and *lab attacks*. A *hack attack* is a software attack in which the attacker (the PDMS owner or remote attacker) downloads code on the device to control it. A *shack attack* is a low-budget hardware attack, i.e., using basic equipment and knowledge. Finally, a *lab attack* is the most advanced, comprehensive and invasive hardware attack for which the attacker has access to laboratory equipment, can perform reverse engineering of a device and monitor analog signals. Note that shack and lab attacks require a physical access to the device and that TEEs are designed to at least resist hack and shack attacks.

Our threat model considers three security assumptions:

Assumption 1. Each PDMS is locally secured by using TEE-like technology flourishing nowadays (e.g., [18, 20, 29]). This assumption is reasonable considering that a PDMS is supposed to store the entire digital life of its owner. A major security feature of TEE technology is to provide isolation, i.e., strong guarantees that the local computation inside the TEE cannot be spied upon, even in the presence of an untrusted computational environment. Hence, to break to confidentiality barrier of a TEE, a lab attack is mandatory. This has an important consequence: *an attacker cannot conduct a successful attack on a remote node, i.e., not under her possession.*

Assumption 2. Each PDMS device is supplied with a trustworthy certificate attesting that it is a genuine PDMS. Without this assumption, an attacker can easily emulate nodes in the network, and conduct a Sybil attack [11], mastering a large proportion of nodes (e.g., playing the role of data processor nodes), thus defeating any countermeasure. Note that this does not require an online PKI (the certificate can be attached to the hardware device and not to the device owner).

Assumption 3. *Corrupted nodes by a lab attack behave like covert adversaries*, i.e., they derive from the protocol to obtain private information only if they cannot be detected [7], as detected malicious behavior leads to an exclusion from the system.

2.4 Threat Model

The above considered assumptions already offer a certain level of security at the node and system levels. Yet, no hardware security can be described as unbreakable. Therefore, our threat model considers that an attacker (e.g., one or several colluding malicious users) can possess several PDMSs and conduct lab attacks on these devices, thus mastering several corrupted nodes which can collude. For simplicity, we will call them *colluding nodes*.

It is important to notice that the worst-case attack is represented by the maximum number of colluding nodes in the system (i.e., controlled by a single entity). Corrupting few nodes can lead to some private data disclosure, but this will be very limited in a well-designed system with a large number of nodes. Therefore, an attacker needs to increase the collusion range to fully benefit from the attack (i.e., access a significant amount of private data).

Thereby, the remaining question is: how many colluding nodes could an attacker control in the system? The main difficulty for an attacker is that colluding nodes must remain indistinguishable from honest nodes (see Assumption 3). Since PDMSs are associated to “real” individuals (e.g., by delivering the device only to real users proving their identity), collusions between individuals remains possible (hidden groups) but such collusions cannot scale without being minimally advertised, hence breaking the indistinguishability mentioned above. Thus, wide collusions are extremely difficult to build since it requires significant organization between a very large number of users, which in practice requires an extremely powerful attacker as well as extreme discretion, and are thus the equivalent of a state-size attack. Finally, note that considering a large proportion of colluding nodes (e.g., 10%) is vain as it would inexorably lead to large disclosure whatever the protocol having a reasonable overhead (e.g., outside the MPC scope). Hence, in this paper we consider that a very powerful attacker could control up to a small percentage (e.g., 1%) of the nodes, which corresponds to a wide collusion requiring a lab attack on these nodes as well as a highly organized collusion between the owners of those nodes.

What does the system protect? The objective of SEP2P is to offer the maximum possible confidentiality protection of the user private data under the above considered threat model. Many other issues related to statistical databases (e.g., inferences from results, determining the authorized queries, query replay, fake data injection, etc.) or to network security (e.g., message drop/delay, routing table poisoning [39]) are complementary to this work and fall outside of the scope of this paper. Similarly, we leave aside the problems related to the attestation and integrity of the code executing distributed computations (e.g., against corrupted nodes that maliciously modify the computation results).

2.5 SEP2P Requirements

Given the considered threat model, we derive in this section the requirements that a SEP2P must address to protect the data privacy of the users. Since we cannot exclude having colluding nodes in the system and since the colluding nodes behave like covert adversaries, private information leakage is unavoidable. Under these conditions, the best countermeasures one can take are: (i) *minimize the risk* of a data leakage, i.e., reduce at most the

probability of a leakage to happen; and (ii) *minimize the impact* of a data leakage, i.e., reduce at most the leakage size. Obviously, these countermeasures should not generate overheads that render the system unpractical. This leads to:

Requirement 1 (security). Random actor selection. Ensure that colluding nodes cannot influence the selection of the data processor nodes.

Requirement 2 (security). Task atomicity. Data tasks should be atomic, i.e., reduced to a maximum such that it minimizes the required sensitive data to execute the task.

Requirement 3 (efficiency). Security overheads. Minimize the number of costly operations, e.g., cryptographic signature verifications or communication overhead, and ensure system scalability with an increasing number of nodes or colluding nodes.

The task atomicity requirement is similar to the principle of compartmentalization in information security, which consists in limiting the information access to the minimum amount allowing an entity to execute a certain task. Typically, a node can execute a subtask without knowing the purpose or the scope of the global task. Dividing a given distributed computation in atomic tasks obviously depends on the precise definition of that computation. Hence, we restrict our analysis in Section 5 to sketches of solutions for the three application classes considered in this paper.

Independently of the distributed protocol chosen to implement some given application, the system must delegate the data-oriented tasks to randomly selected nodes. Therefore, the random selection protocol is generic and constitutes the security basis of any distributed protocol in our system. However, given the considered threat model, it is challenging to design an actor selection protocol that is both secure and efficient. Section 3 addresses this problem while section 4 evaluates the proposed solution.

3 SECURE ACTOR SELECTION

Let us first detail some useful classical cryptographic tools focusing on the properties used in our protocol.

Background 2. A cryptographic hash function [24] is a one-way function that maps a data of arbitrary size to a fixed size bit string (e.g., 224 bits) and is resistant to collision. An interesting property of hash functions is that output distribution is uniform. In the following, $hash()$ refers to cryptographic hash.

Background 3. A cryptographic signature [24] can be used by a node n to prove that a data d was produced by n (authentication) and has not been altered (integrity). The signature is produced by encrypting $hash(d)$ using the private key of n . Any node can verify the signature by decrypting it using the public key of n and comparing the result with $hash(d)$. The signature includes the signer public key certificate, $cert_n$ (see Assumption 2).

We consider a system of N nodes, in which we want to randomly select A actors, despite wide collusion attacks from C colluding nodes. The main notations are summarized in Table 1.

3.1 Effectiveness, Cost and Optimal Bounds

Ideally, we would want to ensure that all A actors are honest, but this is impossible, since colluding nodes are indistinguishable from honest nodes. Therefore, the best achievable protection is obtained when actors are randomly selected and the selection cannot be influenced by C colluding nodes, i.e., the average number of corrupted selected actors in the ideal case is $A_{ideal_C} = A \times C / N$

N	Total number of nodes in the SEP2P system
A	Number of actor nodes (data processors)
C	Maximum number of colluding nodes ($C \geq 1$)
A_C	Average number of corrupted actors for a given protocol
A_{ideal_C}	Average number of corrupted actors for an ideal protocol
T	Triggering node (starting the execution)
k	Security degree
α	Security threshold
S	Execution Setter node, computing actor list
R_i, rs_i	DHT region R_i of size rs_i

Table 1: Main notations for Sections 3.1 and 3.2

($A_{ideal_C} > 0$). Thus, the impact of a collusion attack remains proportional with the number of colluding nodes, which is the best situation given our context. This guarantees that the attacker cannot obtain more private information than what she can passively get from observing the information randomly reaching its colluding nodes.

The following definitions quantify the security effectiveness and security cost of an actor selection protocol.

Definition 1. The **security effectiveness** of an actor selection protocol is defined as the ratio between A_{ideal_C} and the average number of corrupted selected actors for the measured protocol (A_C), i.e., security effectiveness = A_{ideal_C}/A_C . The security effectiveness has maximum value (i.e., 1) when $A_C = A_{ideal_C}$ and minimum value (i.e. C/N) when all the actors are corrupted.

Definition 2. A **verifier node** is a node who needs verifying the actor list before delivering sensitive data, e.g., a data source.

Definition 3. The **security cost** of an actor selection protocol is defined as the number of asymmetric cryptographic operations, e.g., signature verification, required by verifier nodes to check the selected actor list.

Note that the security cost considers only the verification of the actor list and not the cost of building the list. The rationale is that the verification cost has a larger impact on the overall performance since the number of verifier nodes can be high in a large distributed system: data sources need to verify the actor list before delivering their data. Other performance related issues (cost of the actor list generation, load balancing, maintenance costs) are discussed in Section 3.6 and 4.

Optimal bounds. The best possible case one could expect in terms of security effectiveness and cost in our context can be achieved using an idealized trusted server that knows all the nodes and provides a different random actor list for each system computation. This ideal solution reaches a maximal security effectiveness and a security cost of 1, since any verifier node must only check the signature of the trusted entity.

Evidently, this solution is not acceptable since it represents a highly desirable target for attackers, i.e., a central point of attack and contradicts the fully distributed nature of SEP2P. Therefore, we need distributed solutions relying only on the nodes. To underline the existing tension between security effectiveness and cost, we discuss two basic distributed protocols for the actor selection, focusing either on the security cost or on the security effectiveness. To simplify the protocols description, we initially assume a full mesh network overlay, i.e., each node knows the complete list of nodes in the system and its evolution over time.

Baseline cost-optimal protocol. The triggering node (T) selects randomly the actors. The security effectiveness is minimal:

$A_C = \min(A, C)$ since T may be corrupted (which is the case when any node can trigger a computation). There is thus no necessity to provide any signature: the security cost is 0.

Baseline security-optimal protocol. Proposing an optimal protocol in terms of security is challenging in a decentralized architecture (without any supporting trusted party) and considering covert adversaries. This conjunction leads to a situation where no single node in the system can claim to securely provide a list of actors (the provider itself can be corrupted). The work in [8] proposes the CSAR protocol which provides a secure way to generate a verifiable random value under the condition that there is at least one honest node participating in the distributed protocol. Applying to our context, we can ensure generating a real random value only if there are at least $C + 1$ participating nodes. Also, once we obtain a verifiable random value, we can derive up to A random values by repeatedly hashing the initial value $A - 1$ times. The final step is to map the set of A random values to the nodes. This can be easily done, e.g., by sorting the nodes on their public key and associating the random value to a rank in the sorted list. This protocol has an optimal security effectiveness, i.e., 1, since the actors are guaranteed to be selected randomly. On the other hand, checking the CSAR results requires one signature verification per participant. Thus, the security cost is $C + 1$ asymmetric cryptographic operations per verifier node. Since C can be large, such a solution cannot scale with large systems and wide collusion attackers as it would lead to an extreme verification cost.

Moreover, to achieve these security bounds, both protocols require a full mesh network overlay which is also extremely costly to maintain in practice, especially for large networks. This contradicts the efficiency and scalability requirement formulated in Section 2.5. Using a DHT overlay instead of a full mesh solves the problem of communication efficiency/scalability. However, this will impact the optimal bounds of both protocols. For the first protocol, the security cost increases from 0 to up to A since a verifier node which does not “know” any of the actors has to verify their certificates to be sure that the actors are genuine PDMSs (to avoid Sybil attacks). Similarly, for the second protocol, the security cost increases to $2(C + 1) + A$ for the same reason, i.e., checking that participant and selected actors are genuine PDMSs. Even worse, the optimal security effectiveness can no longer be guaranteed since with a DHT, there is no secure way of associating the random values to the nodes unless using secure DHT techniques [39] with a large impact on performance.

3.2 Overview of the Proposed Solution

To address all these problems, we propose a protocol that reaches maximal security effectiveness at a verification cost of $2k$. k is called the *security degree* and is very small. Also, our protocol builds directly on a classical, efficient DHT overlay without requiring any modifications. We describe some important features in SEP2P which make this possible and then sketch the protocol.

Imposed and uniform distribution of node location: the node ID, used when inserting a node in the DHT, is imposed in SEP2P, in a way that leads to a uniformly distributed node location in the DHT virtual space. Consequently, colluding nodes are also evenly distributed in the DHT, thus avoiding spatial clusters. We use extensively this property to drastically reduce the cost of security by taking localized decisions (see below), i.e., limited to the nodes situated in “small” regions in the virtual space. Achieving imposed node location is easy, based on the public

key of the certificate of each node. We compute a cryptographic hash of this key, which is, by construction, uniformly distributed, and use this hash for insertion in the DHT virtual space. The advantages of using the public key are (i) its uniqueness; and (ii) the node location can be checked with a single signature verification.

Probabilistic guarantees: Given the imposed, uniform node location which applies indistinctly to honest and colluding nodes, we can have probabilistic guarantees on the maximum number of colluding nodes in a DHT subspace of a given size, called *DHT region* hereafter. We can compute the probability of having at least k colluding nodes (see Section 3.3) and choose the DHT region size such that the probability is very close to 0. In our context, we want to have a probability smaller than α , the security threshold. The main idea is to set α so that the probability of having k colluding nodes in the same region becomes so low that we can consider that it “never happens”, e.g., $\alpha = 10^{-6}$ (see Section 4.1). Such a guarantee is used in the protocol sketched below and then detailed in the following subsection.

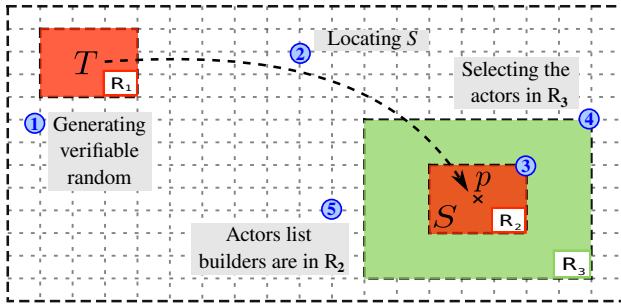


Figure 1: Sketch of verifiable selection

Sketch of verifiable selection protocol of A actors (see Figure 1)

- (1) Run a distributed protocol inspired from CSAR [8] to generate a **verifiable random value**, i.e., proven to have been truly randomly generated by k nodes if at least one is honest (see Section 3.4). The k nodes are selected in a DHT region R_1 , centered on the triggering node (T), whose region size rs_1 is set such that we have probabilistic guarantees to “never” (probability $< \alpha$) have k or more colluding nodes, i.e., at least one of the k nodes is honest.
- (2) Map the hash of that random value into coordinates to define a location p in the DHT virtual space and contact through the DHT the node, called **execution Setter** (S), managing this location.
- (3) S then selects k nodes (the actor list builders) in a region R_2 , centered on p , using probabilistic guarantees, such that we “never” have k or more colluding nodes. Given the uniform distribution of the node on the virtual space, we have $rs_2 = rs_1$.
- (4) Each actor list builder then selects A nodes in a region R_3 , centered on p , whose size rs_3 is such that R_3 includes at least A nodes with high probability (see Section 3.6 and Section 4.3 for rs_3 tuning).
- (5) Run a **distributed verifiable selection protocol** in the spirit of [8] such that the k nodes selected in (3) can: (i) check the validity of the random value generated in (1); (ii) build the actor list securely; (iii) sign both the random value and the list of A actors. This step is detailed in Section 3.5.

The result is a list of A actors that is signed by k nodes, among which at least one is honest. Doing so reduces the verification cost to $2k$ asymmetric cryptographic operations: k to check the certificate of the k list builders, verifying that they belong to region R_2 , centered on p ; and k to check each builder signature.

3.3 Providing Probabilistic Guarantees

To generate verifiable random values or validate the query actor selection, SEP2P employs distributed computations between a small subset of the nodes thanks to the notion of node legitimacy and probabilistic guarantees defined below using the notations in Table 2.

$kpub_n$	Public key of node n
$cert_n$	Trustworthy certificate of node n
$sign_n$	Signature by node n (includes $cert_n$)
TL_i	execution Trigger Legitimate node i
RND_i	Random number generated by TL_i
$(V)RND_T$	(Verifiable) random generated by T
SL_j	execution Setter Legitimate node j
RND_S	Random generated by S
CL_j	Partial candidate list of legitimate nodes w.r.t. R_3
CL	Candidate List of legitimate nodes
$(V)AL$	(Verifiable) Actor List

Table 2: Main notations for Sections 3.3 – 3.5

Definition 4. Legitimate nodes. Given a region R in the virtual space of a DHT, for any node i we say that node i is legitimate w.r.t. R iff $hash(kpub_i) \in R$.

To be able to provide probabilistic guarantees as explained in Section 3.2, we need to estimate the number of nodes in a region:

Lemma. Let R be a DHT region of size rs in a virtual space of a DHT of total size 1 (i.e., normalized) and let N be the total number of network nodes with a uniform distribution of the node location in the virtual space. The probability, PL , of having at least m legitimate nodes in R is:

$$PL(\geq m, N, rs) = \sum_{i=m}^N \binom{N}{i} \cdot rs^i \cdot (1-rs)^{N-i} \quad (1)$$

Proof (sketch): Let us consider a partition of the N nodes into two subsets containing i and $N-i$ nodes. Since the distribution of nodes is uniform in space, the probability of having the i nodes inside R and the $N-i$ nodes outside R is $rs^i \cdot (1-rs)^{N-i}$ and there are $\binom{N}{i}$ possible combinations of generating this node partitioning. The probability of having at least m nodes in R is equal to the probability of having exactly m nodes plus the probability of having exactly $m+1$ plus... the probability of having N , which leads to the equation in (1).

Application to colluding nodes: Let $C < N$ be the maximum number of colluding nodes. We can apply formula 1 to compute the probability, PC of having at least k colluding nodes in R :

$$PC(\geq k, C, rs) = \sum_{i=k}^C \binom{C}{i} \cdot rs^i \cdot (1-rs)^{C-i} \quad (2)$$

We can notice that this probability only depends on C . It does not depend on the region center since we have a uniform distribution of the nodes on the virtual space.

3.4 Verifiable Random Generation

Our goal is to generate a random value, using k nodes and to guarantee that none of the k nodes can choose the final computed random value (or any of its bits). Any node in the system should be able to check the validity of this random value (i.e., to have proofs that it has been correctly generated). This is possible as soon as at least one of the k nodes is honest, this guarantee being obtained thanks to equation (2) by choosing the adequate size for the DHT region R and by using k legitimate nodes w.r.t. R .

A node T wanting to generate a verifiable random, selecting a region of size rs_1 with $PC(rs_1) < \alpha$ centered on itself, executes:

Verifiable random number generation protocol

- (1) T contacts any k legitimate nodes TL_i ($i \in [1, k]$) w.r.t. R_1 .
 - (2) Each TL_i sends $hash(RND_i)$ to T , where RND_i is a random number (on the same domain as the hash function, e.g., 224 bits) TL_i generates.
 - (3) Once T has received the k hashes, it sends back the list L of hashes to the TL_i s; $L = (hash(RND_i))_{i \in [1, k]}$.
 - (4) Each TL_i checks that $hash(RND_i) \in L$, and, in the positive case, returns $sign_i(L)$ and RND_i .
 - (5) T gathers the k messages and builds the verifiable random:
 $VRND_T = (cert_T, (sign_i(L), RND_i)_{i \in [1, k]})$.
-

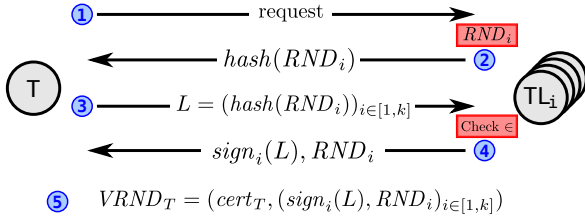


Figure 2: Verifiable random

The above random generation protocol is adapted from [8] which includes a formal proof. Note that the protocol in [8] does not include the notion of node legitimacy and thus needs $C + 1$ participating nodes instead of k . Intuitively, the nodes commit on their selected random value by sending its hash (Step 2), and all the hash values are known by each of the k nodes before providing the final signature (Step 4). Therefore, an attacker controlling $k - 1$ TL_i nodes cannot influence the final random value since these nodes cannot change their random values (committed at Step 2). Thus, the correct random value of a single honest node is enough to obtain a truly random final value RND_T .

To obtain and check the verifiable random value, any node must: (i) check $cert_T$ and compute L by hashing all RND_i ; (ii) for $i \in [1, k]$, check $cert_i$, check the legitimacy of TL_i using $cert_T$ and validate $sign_i(L)$. The final random value is $RND_T = RND_1 \oplus RND_2 \oplus \dots \oplus RND_k$.

In (i), we verify that T is a genuine PDMS, retrieve the center of the region R_1 and compute L , both being necessary for the next verification; (ii) starts by confirming that each TL_i is genuine, then it ensures that they are legitimate w.r.t the location of T and R_1 , after which it confirms the hash list by checking the signatures, and finally, it computes RND_T .

3.5 Distributed Secure Selection Protocol

The main goal of the proposed protocol is to select the A actors such that this selection cannot be influenced by colluding nodes.

Definition 5. The **execution Setter** (S) is chosen randomly based on a verifiable random generated by T . Its role is to coordinate the selection of the computation actors and to setup the execution by sending the appropriate information to each actor.

In the following, we assume that each node n in SEP2P keeps a node cache, called $cache_n$, of the IP address and certificate of legitimate nodes w.r.t. a region of size rs_3 centered on node n location. The cache size and the cache maintenance cost are discussed in Section 3.6 and evaluated in Section 4.3.

SEP2P distributed secure actor selection protocol

- (1) Generates the verifiable random $VRND_T$ (see Section 3.4).
 - (2) Maps $hash(RND_T)$ into coordinates and contact S through the DHT.
 - (3) S contacts any k legitimate nodes w.r.t. R_2 , SL_j ($j \in [1, k]$) and sends to each $VRND_T$ (see Section 3.4).
 - (4) Each SL_j sends $hash(RND_j \parallel CL_j)$ to S , where RND_j is a random number SL_j generates, and CL_j is the set of nodes from $Cache_j$ which are legitimate w.r.t. R_3 .
 - (5) Once S has received the k hashes, it sends back the list L_1 of hashes to all SL_j ; $L_1 = (hash(RND_j \parallel CL_j))_{j \in [1, k]}$.
 - (6) Each SL_j checks that its own $hash(RND_j \parallel CL_j) \in L_1$ and, in the positive case, returns RND_j and CL_j .
 - (7) S gathers the k messages and sends to all SL_j the list $L_2 = ((RND_j, CL_j)_{j \in [1, k]})$.
 - (8) Each SL_j does the following:
 - (a) Checks $VRND_T$ and computes RND_T (see Section 3.4).
 - (b) Checks that each (RND_j, CL_j) from L_2 is consistent with the corresponding $hash(RND_j \parallel CL_j)$ from L_1 .
 - (c) Computes the union, after removing possible duplicates, of all CL_j to obtain a candidate list of legitimate nodes CL .
 - (d) Computes the $RND_S = RND_1 \oplus RND_2 \oplus \dots \oplus RND_k$.
 - (e) Sorts CL on $kpub_n \oplus RND_S$ (where $kpub_n$ is the public key of a node $n \in CL$) and selects the A first candidates to build the actor list AL .
 - (f) Checks the legitimacy of AL nodes w.r.t. R_3 .
 - (g) Signs (RND_T, AL) and sends it to S .
 - (9) S gathers k results and builds the verifiable actor lists:
 $VAL = (RND_T, AL, (sign_j(RND_T, AL)))_{j \in [1, k]}$.
-

The goal of **steps 1 and 2** is to displace the DHT region, where actors will be selected, from T to S with three benefits: (1) T is likely to be corrupted (as any node is allowed to trigger a computation) while S is chosen randomly using the verifiable random protocol; (2) it distributes the potential leaks in a different region for each computation; (3) it balances the load on the whole SEP2P network thus improving the overall performance.

Steps 3 to 6 are similar to steps 1 to 4 of the verifiable random protocol, except that the signature by SL_j is delayed to Step 8.g. Delaying the signature allows SL_j s to check and attest the validity of $VRND_T$ (step 8.a). The protocol cost is increased (since k nodes verify $VRND_T$) but the verifying cost is reduced accordingly since having k SL_j s signing RND_T (step 8.g) means that it is correct (remind that at least one of the k SL_j s is honest).

Steps (8.b) to (8.e) are dedicated to the actor list building (AL) based on the candidate list (CL) and deserve a more detailed explanation: in our context, in order to securely build the actor list, the k participants first have to agree on a common basis and then execute, in parallel, a procedure that is unpredictable and gives identical results to all participants. Since it is unpredictable we are certain that the inputs cannot be manipulated beforehand so as to influence the rest of the procedure. Since it gives identical results for all actor list builders, and since at least one node is honest, we are sure that no colluding node can alter the results. By sorting the nodes in CL using a verifiable random number and the public keys of the nodes fulfills both requirements: the random number takes care of the unpredictability, while the commitment of each SL_j on their intermediary lists in step 4, coupled with the XOR operation on the public keys of CL nodes, is a simple yet effective way of producing identical results.

In **steps 8.f and 8.g**, k SL_j s check the validity of the result, i.e., that any actor of AL belongs to R_3 and attest it by signing the results. Note that this check is not necessary for any actor n in AL that was found in k SL_j since this fact attests that at least

one honest node possesses n in its $Cache_j$. Assuming $Cache_j$ contains only genuine nodes (we say that $Cache_j$ is valid - see Section 3.6) and since $rs_3 > rs_2$, most of the actors in AL will be found in k CL_j , thus diminishing drastically the actor list building cost. Actually the validity of $Cache_j$ is necessary to ensure that a colluding node selected as SL cannot hide honest nodes with the hope of having a larger proportion of colluding nodes in AL . Indeed, at least one of the SL is honest and will provide its full $Cache_j$ that will be thus included in CL . We can observe that $Cache_j$ can be actually seen as the relevant part (for node j) of a full mesh network, which offers its benefits without paying the whole maintenance cost.

To check the verifiable actor list (VAL), any verifier node must do: for $j \in [1, k]$, check $cert_j$, check the legitimacy of SL_j using RND_T and validate $sign_j(AL)$. Thus, the verifying cost is limited to k certificate verifications and k signature verifications, i.e., $2k$ asymmetric crypto-operations. We show in Section 4 that k is generally lower than 6.

3.6 Protocol Implementation Details

In this section we discuss a few important implementation issues of the proposed actor selection protocol.

Despite the uniform distribution of nodes on the DHT virtual space, there is no absolute guarantee of not having **sparse DHT regions**. This can have two negative impacts on the SEP2P protocol: during the selection of k TLs in R_1 (or k SLs in R_2) and A actors in R_3 . Both cases exhibit interesting trade-offs:

Choosing R_1 (or R_2) region size: on the one hand, a small rs leads to a smaller k value, which in turn reduces the protocol verification cost. On the other hand, setting rs too small can lead to situations in which nodes have less than k legitimate nodes in their R region and as such cannot participate in the actor selection protocol (as triggering node or execution setter) which is problematic. For this reason, in SEP2P we provide a table of couples (k_i, rs_i) , named **k -table**, which allows any node to find k_i legitimate nodes in the region of associated rs_i size. The k -table is computed thanks to PL and PC (equations (1) and (2)) to ensure that whatever the couple chosen, the probability of having k or more colluding nodes remains equal. The largest k of the k -table corresponds to the region size allowing any node to find those legitimate nodes with a very high probability, i.e., $1-\alpha$, while lower values allow to reduce the security cost in denser network regions. Thus, the k -table optimizes the overall cost of the SEP2P protocol and warrants that any node can be selected as triggering node or execution setter.

Choosing R_3 region size: Choosing a too small rs_3 has a negative impact on the system performance. If the SLs cannot find enough nodes in R_3 , they can attest it (e.g., in Step 8.c in SEP2P protocol) and S can use the k signatures to displace the actor selection to another region (e.g., selected by rehashing the initial RND_T). This mechanism allows the protocol to be executed successfully even if some network regions are sparser. However, there are two drawbacks. First, the cost of the actor selection increases since (part of) SEP2P protocol must be executed twice (or more times). Second, this also introduces an unbalance in the system load since the sparse regions cannot fully take part in data processing. Finally, setting rs_3 to very large values (see Section 4.3) is not an option since the maintenance cost of the cache increases proportionally when nodes join or leave the network.

Joining the network and $Cache_j$ validity: Due to space limitation, we only sketch the joining procedure in the case of a Chord

DHT (leaving the network can be easily deduced). As mentioned above, any node must maintain a consistent node cache despite the natural evolution of the network. Thus, a node joining the network must ask its successors and predecessors (Chord DHT) to provide their node cache attested by k legitimate nodes in a region of size rs_1 centered on their location. The new node can then make the union of these caches and keep only legitimate nodes w.r.t R_3 centered on its location. The resulting cache contains only genuine nodes and is thus valid since it has been attested by at least k nodes in a region of size rs_1 centered on the successors or predecessors of the new node (a recurrence proof can be established).

Reusing an actor list: If there is no mechanism that prevents an attacker from reusing an actor list, then she only has to keep generating such lists until she obtains one she deems satisfactory. To counter this behavior, we put in place two mechanisms: (i) a timestamp and (ii) a limit to the number of triggered executions a node can make. With (i) we prevent any node from reusing an actor list: TLs and SLs add a timestamp to their signatures which will respectively be checked by the SLs and the data sources. If the timestamp is too distant, the computation is cancelled. Enforcing (ii) is possible thanks to the node cache and the k -table: the TLs solicited by T first check if T chose the smallest possible number of TLs (as their node cache contains, by construction, R_1 centered on T , they are capable of judging), thus forcing T to choose the same TLs . They then only have to monitor and limit the number of queries T does in a given amount of time.

Failures and disconnections: In the most complex case of node failures (i.e., unexpected disconnection) of a TL , SL or S , either RND_T or AL cannot be computed and the protocol must be restarted (i.e., T generates a new RND_T). However, the probability of failures during the execution of the secure actor selection being low in our context, such restarts do not lead to severe execution limitations as mentioned above. The case of “graceful” disconnections is easier: we can safely force nodes involved in the actor selection process to remain online until its completion, thus avoiding the restarts. If a node, selected as actor wants to disconnect (or fails), the impact will be mainly on the result quality since part of the results will be missing.

4 EXPERIMENTAL EVALUATION

This section evaluates the effectiveness, efficiency, scalability and robustness of the SEP2P actor selection protocol.

4.1 Experimental Setting

Reference methods. To better underline our contributions and to provide a comparison basis, we implemented three strategies in addition to the SEP2P actor selection protocol. We discarded the baseline cost-optimal and security-optimal protocols from the evaluation since the former does not provide any security while the latter is much too costly and not scalable (w.r.t N and C) to be used in practice. Hence, we used for comparison more advanced actor selection strategies based on these protocols but using our verifiable random generation protocol with k participants (see Section 3.4).

The first two strategies use the verifiable random to designate the execution Setter (S) which freely chooses the actor list (as in the cost-optimal protocol). These strategies differ only in the verification process. The first one, ES.NAV (for Execution Setter, No Actor Verification) requires verifying the legitimacy of S but not of the actors. The second one, ES.AV requires, in addition,

to verify that actors are genuine PDMSs. ES.AV is expected to provide better security effectiveness than ES.NAV at a higher verification cost. The third strategy, M.Hash (for Multiple Hash) is derived from the security optimal protocol, but uses a DHT instead of a full mesh network. Verifiers must check that actors are genuine PDMSs and that they are “near” the random values determined by the initial verifiable random, hashed as many times as there are actors.

Strategy	Description	
ES.NAV	Execution Setter with No Actor Verification	
ES.AV	Execution Setter with Actor Verification	
M.Hash	Multiple Hash (with Actor Verification)	
SEP2P	Proposed protocol (Section 3.5)	
Param.	Description	Values(default)
N	Number of nodes	10K; 100K ; 10M
$C\%$	% of colluding nodes	0.001%; 0.01%; 0.1%; 1% ; (10%)
A	Number of actors	8; 16; 32 ; 64; 128; 256
α	Security threshold	10^{-4} ; 10^{-6} ; 10^{-10}
$ Cache_j $	Node cache size	48 or varying from 8 to 32K
MTBF	Mean time betw. failure	from 1h to 5 days
Metrics		Unit(s) & comments
Security effectiveness		Ratio (1 = ideal, C/N = worst)
Verification cost		Number of asymmetric crypto-operations
Latency of setup cost		Number of exchanged messages and
Total work setup cost		number of asymmetric crypto-operations
Maintenance overhead		(per minute for the maintenance overhead)
Security degree (k)		Ratio (1 = ideal, C/N = worst)

Table 3: Strategies, parameters and metrics

Simulation platform. We identified all the parameters that may impact the security and efficiency of the proposed strategies and considered all the metrics (see Table 3) that are worth evaluating to analyze the strengths and weaknesses of the proposed strategies, i.e., security effectiveness and cost, setup cost, scalability, robustness w.r.t. failure or disconnections. Let us note that a real implementation of the SEP2P distributed system is not very useful if we consider the above listed objectives of the evaluation. Also, measuring the scalability for very large systems (e.g., 10M nodes) with many parameters is practically impossible. Therefore, as in most of the works on distributed systems [30, 34], we base our evaluation on a simulator and objective metrics. That is, the latency is measured as the number of asymmetric crypto-operations and exchanged messages between peers instead of absolute time values. This allows for a more precise assessment of the system performance than time latency, which can greatly vary in our context because of the node heterogeneity (e.g., TEE resources or network performance).

Our simulator is built on top of a DHT network. Currently, we implemented Chord and CAN as DHT overlays and use Chord for the results presented in this paper. The simulator allows to force choosing a given Execution Setter (by artificially fixing the RND_T value). We used this feature to obtain the exhaustive set of cases for a given network setting, each node being the Execution Setter, and then capture the average, maximum and standard deviation values for our metrics. The parameters and metrics of the simulator are described in Table 3. Values in bold are the default choices and their tuning is discussed throughout the Section. Note that (1) the verification cost is given by verifier node; (2) the latency indicates the “duration” of the protocol executed in parallel; (3) the total work indicates the cumulative number of cryptographic operations and communications during the execution of a protocol.

Security threshold value: Generating several networks and varying the security threshold α , we experimentally observed that for $\alpha = 10^{-4}$, an attacker never controls k or more nodes. However, given the importance of this parameter for the system security, we set $\alpha = 10^{-6}$ and show in Figure 6 the impact of choosing $\alpha = 10^{-10}$ on a small (10K) and large (10M) network. Indeed, if an attacker could master by chance k colluding nodes in a region of size $rs_1 = rs_2$, then she could completely circumvent the security mechanism of SEP2P since, for example, she can obtain k signatures from these regrouped colluding nodes for an actor list of her willing. Note that increasing α reduces the probability accordingly but increases the verification cost in a logarithmic way (as discussed below in Section 4.3).

4.2 Security Effectiveness versus Efficiency

Figure 3 represents the security effectiveness (Y axis) versus the verification cost (X axis) for the four measured strategies and with $C\%$ varying from 0.001% to 10%. Note that the value of 10% is not realistic: it would lead to large disclosure even with an optimal random actor selection protocol, and as mentioned in Section 2.4, is equivalent to state-size attack. We have however run the simulation with 10% to understand its impact on the security effectiveness and cost.

Security effectiveness: SEP2P achieves an ideal security effectiveness, i.e. as good as a trusted server, independently of the number of colluding nodes. Indeed, the selection of actors is truly random, thus providing the same results as the ideal case. In addition, the verification cost ($2k$) is also very low (4 to 8 asymmetric crypto-operations for $C\% \leq 1\%$). Not surprisingly ES.NAV has the same verification cost than SEP2P, but the cost of ES.AV or M.Hash is much larger ($2k + A + 1$ and $2k + A$ respectively) since both must check the certificate of each actor in the list. This check allows ES.AV to have better security effectiveness than ES.NAV when C is very small ($C < A$). With respect to security effectiveness, ES.NAV, ES.AV and M.Hash are far from offering an adequate protection. Let us explain the cause for the poor security effectiveness: while RND_T value is correctly chosen, an attacker mastering a corrupted node located “sufficiently near” from $hash(RND_T)$ can claim to be the Execution Setter and then select a list of actors including a maximum number of colluding nodes. Here, “sufficiently near” means that it satisfies the check made by the verifiers. Note that we tuned the system parameters such that we can be “sure” to have always a node sufficiently near of any random value to allow executing the actor selection protocol for any RND_T . The same problem arrives with M.Hash for each new random destination, thus explaining the poor security effectiveness. Hence, increasing the number of verifications or selecting each actor in a different network region does not solve the intrinsic limitation of these strategies. Note also that this behavior does not affect SEP2P. Indeed, even if the Execution Setter is a corrupted node, it cannot influence the actor list selection since it is done by k SLs (S only routes the messages between the SLs).

Setup costs: Figures 4 and 5 show the setup costs (Y axis in log scale) in terms of asymmetric crypto-operations and exchanged messages respectively, once more with respect to the verification cost (X axis). Curves with empty symbols represent latency while plain symbols represent total work. The results show that SEP2P is the slowest in latency and has the higher total setup cost for crypto-operations. These “bad” results are the consequence of two design choices: (1) to increase the security effectiveness, we

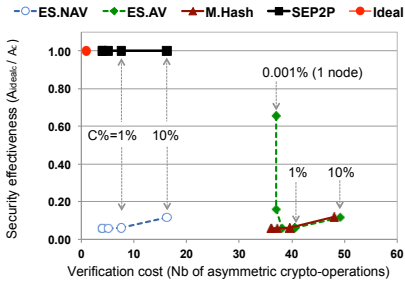


Figure 3: Sec. Effectiveness vs Verification

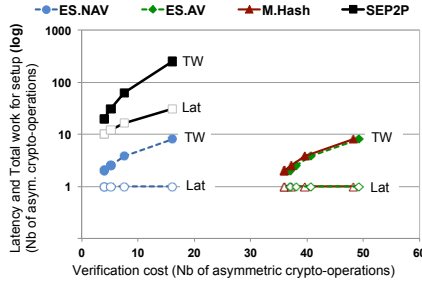


Figure 4: Setup asymmetric crypto-costs

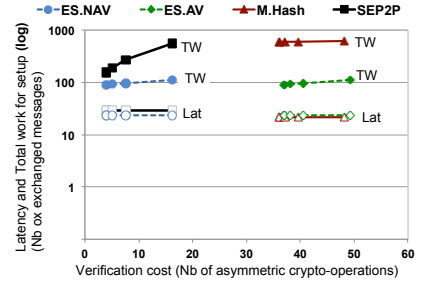


Figure 5: Setup communication costs

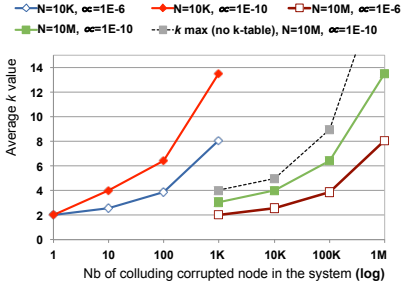


Figure 6: k versus C (N and α vary)

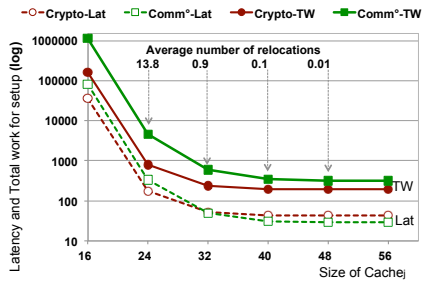


Figure 7: Setup costs varying $R3$ size

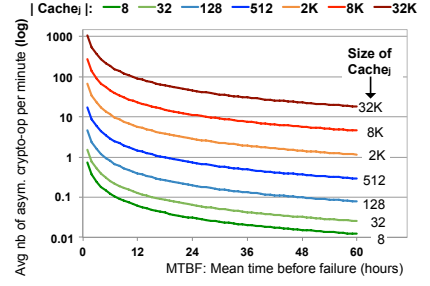


Figure 8: Maintenance overheads

run our protocol on k SL nodes thus increasing the total setup cost; and (2) we voluntarily make most of the checks during the setup (e.g., checking the actor certificates or verifying their availability) in order to reduce, as much as possible, the subsequent verification cost. Since this verification process will potentially be performed by a (very) large set of nodes (e.g., data sources), it is in our best interest to reduce it to avoid overloading the entire system. Figures 4 and 5 illustrate this aspect: our non-optimal setup cost is balanced by an optimized verification cost (and ideal disclosure in Figure 3). Note also that most operations are done in parallel (either by k TLs or SLs), thus leading to a reasonable setup latency (around 20 crypto-operations and 30 exchanged messages). We can also note in Figure 5 that M.Hash achieves the worst total work for setup (exchanged messages), because of the A routings in the DHT. Finally, we can remark the almost identical latency of ES.NAV, ES.AV and M.Hash on both metrics. Indeed, they all run the same initial protocol to compute RND_T . With respect to communication, the results are also identical because all DHT routings for M.Hash are done in parallel.

4.3 Scalability and Robustness

We now concentrate on SEP2P to study its scalability and its robustness to node failure.

Scalability: To study the scalability, we compute the averaged k value varying C and N . Indeed, k is the main factor in the verification cost, setup latency and total work (since everything is done k times). As seen in Section 3.6, depending on C and N , we can compute a k -table which gives several increasing values of k with increasing region size. We have considered small (10K) to very large (10M) networks and four values for $C\%$, leading to eight different SEP2P network configurations. For each configuration, we have computed, for each node, the minimal value for k with respect to the k -table and then averaged the results. Figure 6 shows the average k (Y axis) versus the $C\%$ (X Axis in log scale) for several network size considering two values for α : 10^{-6} and 10^{-10} . We also plot on the same figure the value of k without

k -tables (the grey curve) to highlight the benefit brought by k -tables (only shown for the large network with $\alpha = 10^{-10}$). This figure offers many insights. (1) **SEP2P is highly scalable w.r.t. N :** Indeed, k values are identical for small and large networks independently of α if we consider the percentage of colluding nodes and not the absolute value (e.g., 1% colluding nodes is equivalent to absolute values of $C = 100$ and $C = 100K$ for the small and large networks). Indeed, scaling N and C in the same proportion leads to reduce $rs_1 = rs_2$ size accordingly. Note that with a single corrupted node, the k optimization is useless ($k = C + 1$ in that case) regardless of the α value. (2) **k increases slowly when $C\% < 1\%$:** k remains smaller than 6 even with $\alpha = 10^{-10}$. For $N = 10M$ and $C\% = 1\%$, the k -optimization reduces the number of participants in the verifiable random generation from 100K to 6. (3) **α has a small influences on k :** increasing α by four orders of magnitude increases k from 1 unit (e.g., 1K colluding nodes for $N = 10M$) to 5 units (e.g., 1K colluding nodes for $N = 10K$ or 1M colluding nodes for $N = 10M$). (4) **the k -table optimization is important:** k -tables allow reducing k by 1 unit up to 9 units (for 10% colluding nodes).

Number of actors: We also studied the impact of the variation of the number of actors. Overall, this results in a linear increase in the total work in terms of communications as the k SLs must check for the availability of A legitimate nodes to construct their respective CLs. For the sake of brevity, we omit here these results.

Node cache size: We now focus on adapting the node cache size to the maximum number of required actors. Our goal is to evaluate the impact of the cache size on the global performances. To do so we take a reference network with $N = 100K$, $C\% = 1\%$ and $A = 32$ and vary the average cache size on the whole network (we compute rs_3 easily dividing the cache size by N). Figure 7 shows the results (Y axis in log-scale). For each cache size, we simulated an execution on each node of the network and computed the average values for our metrics. Our measures show that with a very small cache, the probability of relocating the actor selection process is high (the SLs do not find enough legitimate nodes in

their cache w.r.t. R_3), which then leads to an increased latency and total work. Choosing a cache size greater than A , the query is almost never relocated (see Figure 7), giving better performances. This would lead to choose the largest possible cache. However, constructing such a cache also means maintaining it.

Maintenance costs: We also evaluated the impact of the cache size in the presence of node disconnections and, more generally, the impact of disconnections. To observe it, we simulated disconnections and measured their cost depending on the size of the node cache ($Cache_j$) using the default values for C , N , α and resulting k . We then considered those costs as a baseline and computed the global impact in a network where nodes disconnect (and reconnect) every x hours (mean time before failure or MTBF). We represent this cost in terms of asymmetric cryptographic operations (see Figure 8 - Y axis in log scale). The number of exchanged messages is not shown because graphs are very similar. We also computed these metrics for large node cache sizes (up to 32K) to confirm that full mesh networks cannot be an alternative to DHT. Our results show that an overestimated cache is excessively costly even with an MTBF of 5 days: it consumes a large portion of the overall computing power of the entire system just to maintain it up to date. With small MTBFs, the network would be probably not maintainable. Since the number of actors for a computation is likely to be relatively small (e.g., few hundred, see Section 5), we can safely set the node cache size around 512 which leads to a reasonable maintenance cost (less than 1 signature per node per minute on average for MTBF = 1 day) and never trigger relocations (see Figure 7).

5 TASK ATOMICITY

5.1 Proposed Use Cases

We now focus on requirement 2, illustrating task atomicity on the use cases proposed in Section 1.

Use case 1: Mobile participatory sensing is used in many smart city applications for urban monitoring such as traffic monitoring (e.g., Waze or Navigon), evaluating the quality of road infrastructures, finding available parking spaces or noise mapping [36]. In these scenarios, the community members act as mobile probes and contribute to spatial aggregate statistics (density, averaged measures by location and time, spatial interpolation [36]) which in turn, benefit the whole community. As an alternative to the classical centralized architecture, the distributed PDMS paradigm increases the privacy guarantees for the users, thus encouraging their participation. A mobile user can generate sensing data (e.g., using her smartphone or vehicular systems) which is securely transmitted and recorded into her PDMS (e.g., a home box). This way each PDMS becomes a potential data source in the system. These data can then be aggregated by a small subset of data processor nodes to produce the required spatial aggregate statistics, which can be broadcasted to all the participating nodes.

Use case 2: Users can **subscribe to information flows based on their preference or user profile** (e.g., RSS feeds, specific product promotions or ads, etc.). A user profile can be represented by a set of concepts associating metadata terms (e.g., location, age, occupation, income, etc.) to values specific to each user. These associations are traditionally stored at a publication server to allow targeting the interested nodes. Instead, we propose to distributively store and index those profiles in SEP2P, thus greatly improving users' privacy. We call a concept index, an index associating for each concept the list of node addresses having this

concept. Storing and searching this concept index is straightforward with a DHT. Each node does a $store(concept, IPaddress)$ for each concept in its profile. To find all the nodes matching a certain target profile (e.g., a logical expression of concepts), a DHT search is launched for each concept in the profile. Then, a set of randomly selected data processors are used to pick up the scattered pieces of the concept index, apply the logical expression of the target profile and compute the matching target nodes (TN), i.e., their IP addresses. Finally, the information is sent to the selected targets.

Use case 3: We consider **queries over the personal data contributed by a large set of individuals**, e.g., to compute recommendations, make participative studies. To achieve a high degree of pertinence and avoid flooding the system, such queries should target only a specific subset of the nodes, i.e., the nodes exposing a given user profile. Query examples are numerous, e.g., get the top-10 ranked movies by academics from Paris, or find the average number of sick leave days of pilots in their forties. The query processing is done in two steps which roughly correspond to the use case 2 combined with use case 1. First, the relevant subset of nodes, which match the query profile, must be discovered (use case 2). Then, the selected subset of target nodes become data sources which supply the required data (e.g., number of sick leave days) to compute the query result (use case 1). The main differences are that only the selected nodes provide data and that the result is transmitted only to the querier node and not to the entire system.

5.2 Detailed Node Roles

From the above description, we can define new node roles:

Node role 4. A **metadata indexer (MI)** stores part of the metadata shared by the nodes, allowing pertinent and efficient distributed data processing.

Node role 5. A **target finder (TF)**, applies a logical expression on its input to produce a list of target nodes.

Node role 6. A **data aggregator (DA)** applies an aggregative function to its input and produces partially aggregated results.

Node role 7. A **main data aggregator (MDA)** aggregates its input and produces the final result.

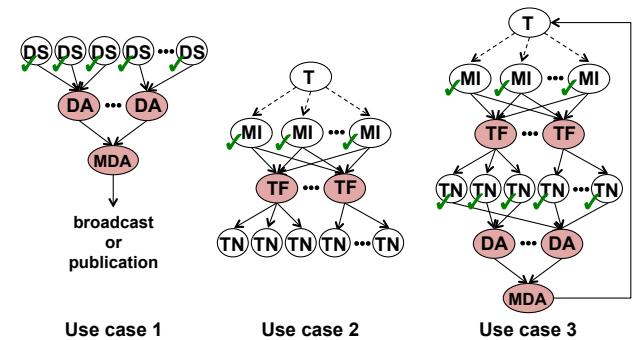


Figure 9: Distributed execution plans for the use cases

These roles allow designing distributed execution plans for the three use cases as shown in Figure 9. The nodes that must be chosen using the SEP2P protocol are shown in pink, and we used the symbol \checkmark to denote that a node is a verifier (as specified in Section 3.6). This must be done each time a node discloses some sensitive data, thus on data sources and metadata indexers.

5.3 Towards Task Atomicity

The node roles and DEP proposed above already provide some task compartmentalization dividing the whole processing in tasks. However, much more can be done to minimize the impact of data leakage. In this section we present a few methods to achieve task atomicity. Our objective is mainly to show that task atomicity can be indeed performed and that it can significantly improve the system security when used in conjunction with the secure random actor selection. Given the space limitation, a detailed study of task atomicity is left for future work.

Metadata index protection: The concept index design already exhibits some form of task atomicity: (1) it is evenly distributed among all the nodes using the DHT mechanisms; (2) the imposed location of nodes in the DHT (see Section 3.2) leads to a randomized association between concepts and *MI* nodes. Nevertheless, a single corrupted node could disclose all the index information it owns. Further security improvements can be obtained by splitting each concept into s shares using the Shamir’s secret sharing technique [32] which requires knowing at least p ($p \leq s$) shares to reconstruct the secret. Disclosing a single concept will now require p colluding nodes randomly selected.

User data protection: We consider here sensed data in use case 1 or the result of queries performed on a single PDMS in use case 2. Considering several *DA*s already reduces the impact of potential data leakage by a corrupted *DA* node. A simple way to reduce further this impact is to realize the aggregation on anonymized data (e.g., average traffic speed without user identity) or data without semantics (e.g., averaging data, a salary for instance, without knowing its meaning) or even encrypted data (with deterministic encryption). Note also that aggregation is continuous in the mobile sensing use case and that selected *DA* node will change at each iteration.

User identity protection: User’s PDMS actively participate in the DEP either by receiving information (use case 2) or queries (use case 3) or by sending information (use cases 1 and 3). They thus communicate with *DA* nodes or receive messages from *TF* nodes, both being potentially corrupted. The reception / transmission task should be “isolated” to make one more step towards task atomicity. This can be achieved using the notion of proxy-forwarder that we illustrate for the *TN-DA* communication in the use case 3. The *TN* (which is actually a data source) must transmit its local result (e.g., number of sick leave days) to the *DA* node. *TN* can choose randomly any node P in the system and send the data, encrypted with the public key of the *DA* (known from the Verifiable Actor List). P will receive this data and transmit it to the *DA*. Thus, *DA* will have the data without knowing the sender, while P will know the sender but not the data. Note that (1) *TN* has good reasons to choose randomly P since it is the most interested in protecting its data; (2) the probability that both *DA* and P to be colluding nodes is extremely low ($\approx (C/N)^2$); and (3) we could use several proxies, thus mimicking anonymization network techniques (e.g., Tor).

6 RELATED WORK

DHT security. Several works focus on DHT security [40] considering the following attacks: (i) Sybil attack: an attacker generates numerous false DHT nodes to outnumber the honest nodes. Introducing an (offline) certificate authority, is deemed to be among the most effective defenses against the Sybil attack [11]. (ii) Routing table poisoning (eclipse attack): an attacker attempts to control

most of the neighbors of honest nodes to isolate them. According to [40] the best strategy against such attacks is to constrain the DHT node identifiers. Again, using a central authority to provide verifiable identifiers is the simplest yet most effective way of achieving this goal [34]. (iii) Routing and storage attacks: Sybil and eclipse attacks do not directly impact the DHT, they are mainly necessary means for future attacks, like various denials of service (DoS). For instance, the objectives might be to prevent a lookup request from reaching its destination, denying the existence of a valid key, or impersonating another node to deliver false data. These DoS attacks are usually classified as routing and storage attacks and most of the mechanisms employed to negate them are based on redundancy at the storage and routing levels [40]. Thus, none of these works consider the secure and efficient actor selection for distributed processing as in SEP2P.

Secure Multi-party Computation and differential privacy. Cryptographic protocols have been proposed to protect the users’ privacy in distributed computations with a focus on data confidentiality enforcement in personal data aggregation. Examples of computations related to this work are personal time-series clustering [2], kNN similarity queries [17], and location-based aggregate statistics [28]. However, MPC raises major scalability issues which in practice limit such protocols to specific types of computations [31].

Although it yields interesting results in privacy protection [15], differential privacy generally requires a central trusted aggregating node and ad-hoc adaptations depending on the targeted queries. As we search to provide a generic framework and exclude having a central actor to avoid a single point of failure, both requirements cannot be met by differential privacy. Even though local differential privacy [13] tries to address our first requirement, the solutions offered until now are still not generic, while the pertinence or the quality of the results may still be problematic with some applications [13]. Also, differential privacy exhibits intrinsic limitations with applications requiring continuous data flow aggregation (e.g., such as mobile participatory sensing) because of temporal correlation between consecutive data batches [10].

Distributed data aggregation using secure hardware. To overcome the limitations of MPC or differential privacy, several works propose using secure hardware at the user-side. Several secure protocols have been proposed for SQL aggregation [37], spatio-temporal aggregation [36], top- k full-text search [21], or privacy-preserving data publishing [3]. SEP2P also considers a secure PDMS at the user-side but our attack model considers having many colluding nodes. Moreover, the focus in SEP2P is on the secure and efficient random node selection. Differently, existing work focus on data aggregation or publishing and consider that all the nodes in the network participate in the protocol with their data being thus complementary to SEP2P.

Secure server-centric approaches. The above cited solutions are based on fully-distributed (P2P) or hybrid architectures. Alternatively, one could envision a solution based on a secured centralized server [6]. However, this raises important issues. First, users are exposed to sophisticated attacks, whose cost-benefit is high on a centralized database. Second, centralizing all users’ data into one powerful server makes little sense in the PDMS context in which data is naturally distributed at the users’ side. Hence, users might be reluctant to use such a massively centralized data service. Finally, new legislation such as the European GDPR [27] may hinder the development of such centralized solutions.

7 CONCLUSION

Personal Data Management Systems arrive at a rapid pace allowing users to share their personal data within large P2P communities. While the benefits are unquestionable, the important risks of private personal data leakage and misuse represent a major obstacle on the way of the massive adoption of such systems. This paper is one of the first efforts to deal with this important and challenging issue. To this end, we proposed SEP2P, a fully-distributed P2P system laying the foundation for secure, efficient and scalable execution of distributed computations. By considering a realistic threat model, we analyzed the fundamental security and efficiency requirements of such a distributed system. We showed that the secure selection of random actor nodes is the basis of security for any distributed computation. Then, we proposed secure and highly efficient protocols to address the actor selection problem. Our simulation-based experimental evaluation indicates that our protocol leads to minimal private information leakage, i.e., increasing linearly with the number of colluding nodes. At the same time, the cost of the security mechanisms depends only on the maximum number of colluding nodes and remains very low even with wide collusion attacks.

This work opens the way for several interesting research problems. In particular, to further minimize the impact of a private data leakage, the random actor selection needs complemented with task atomicity, i.e., decompose the computation process such that it minimizes the amount of sensitive data the processor nodes have access to. To underline this requirement, we discussed in this paper three types of representative applications in the PDMS context and provided sketches of solutions to achieve task atomicity. Certainly, this problem deserves a deeper look and constitutes our main objective as future work.

Acknowledgment. This research is partially supported by the ANR PersSoCloud grant ANR-16-CE39-0014.

REFERENCES

- [1] Tristan Allard, Nicolas Ancaux, Luc Bouganim, Yanli Guo, Lionel Le Folgoc, Benjamin Nguyen, Philippe Pucheral, Indrajit Ray, Indrakshi Ray, and Shaoyi Yin. 2010. Secure personal data servers: a vision paper. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 25–35.
- [2] Tristan Allard, Georges Hébrail, Florent Masseglia, and Esther Pacitti. 2015. Chiaroscuro: Transparency and privacy for massive personal time-series clustering. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 779–794.
- [3] Tristan Allard, Benjamin Nguyen, and Philippe Pucheral. 2014. METAP: revisiting Privacy-Preserving Data Publishing using secure devices. *Distributed and Parallel Databases* 32, 2 (2014), 191–244.
- [4] Nicolas Ancaux, Philippe Bonnet, Luc Bouganim, Benjamin Nguyen, Philippe Pucheral, Iulian Sandu Popa, and Guillaume Scerri. 2019. Personal Data Management Systems: The security and functionality standpoint. *Information Systems* 80 (2019), 13–35.
- [5] Nicolas Ancaux, Luc Bouganim, Philippe Pucheral, Yanli Guo, Lionel Le Folgoc, and Shaoyi Yin. 2014. MLo-DB: a personal, secure and portable database machine. *Distributed and Parallel Databases* 32, 1 (2014), 37–63.
- [6] Arvind Arasu, Spyros Blanas, Ken Eguro, Manas Joglekar, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, Prasang Padhyaya, and Ramarathnam Venkatesan. 2013. Secure database-as-a-service with cipherbase. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 1033–1036.
- [7] Yonatan Aumann and Yehuda Lindell. 2007. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography Conference*. Springer, 137–156.
- [8] Michael Backes, Peter Druschel, Andreas Haerberlen, and Dominique Unruh. 2009. CSAR: A Practical and Provable Technique to Make Randomized Systems Accountable. In *NDSS*, Vol. 9. 341–353.
- [9] Blue Button. 2010. Find Your Health Data. (2010). Retrieved October 12, 2018 from <https://www.healthit.gov/topic/health-it-initiatives/blue-button>
- [10] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. 2017. Quantifying Differential Privacy under Temporal Correlations. In *33rd IEEE International Conference on Data Engineering, ICDE 2017*. 821–832.
- [11] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S Wallach. 2002. Secure routing for structured peer-to-peer overlay networks. *ACM SIGOPS Operating Systems Review* 36, SI (2002), 299–314.
- [12] Cozy Cloud. 2013. Your digital home. (2013). Retrieved October 12, 2018 from <https://cozy.io/en>
- [13] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. 2018. Privacy at Scale: Local Differential Privacy in Practice. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018*. 1655–1658.
- [14] Yves-Alexandre de Montjoye, Erez Shmueli, Samuel S Wang, and Alex Sandy Pentland. 2014. openpds: Protecting the privacy of metadata through safeanswers. *PLoS one* 9, 7 (2014), e98790.
- [15] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. 3574–3583.
- [16] Fing. 2013. The mesinfos project explores the self data concept in france. (July 2013). Retrieved October 12, 2018 from <http://mesinfos.fing.org/english>
- [17] Davide Frey, Rachid Guerraoui, Anne-Marie Kermerrec, Antoine Rault, François Taïani, and Jingjing Wang. 2015. Hide & Share: Landmark-based Similarity for Private KNN Computation. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 263–274.
- [18] Javier González, Michael Hölzl, Peter Riedl, Philippe Bonnet, and René Mayrhofer. 2014. A practical hardware-assisted approach to customize trusted boot for mobile devices. In *International Conference on Information Security*. Springer, 542–554.
- [19] Anne-Marie Kermerrec and François Taïani. 2015. Want to scale in centralized systems? Think P2P. *J. Internet Services and Applications* 6, 1 (2015), 16:1–16:12.
- [20] Saliha Lallali, Nicolas Ancaux, Iulian Sandu Popa, and Philippe Pucheral. 2017. Supporting secure keyword search in the personal cloud. *Information Systems* 72 (2017), 1–26.
- [21] Thi Bao Thu Le, Nicolas Ancaux, Sébastien Gillot, Saliha Lallali, Philippe Pucheral, Iulian Sandu Popa, and Chao Chen. 2016. Distributed secure search in the personal cloud. In *19th International Conference on Extending Database Technology (EDBT 2016)*. 652–655.
- [22] Sangmin Lee, Edmund L Wong, Deepak Goel, Mike Dahlin, and Vitaly Shmatikov. 2013. π Box: A Platform for Privacy-Preserving Apps. In *NSDI*. 501–514.
- [23] Petar Maymounkov and David Mazières. 2002. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*. Springer, 53–65.
- [24] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. 1996. *Handbook of Applied Cryptography*. CRC Press.
- [25] MiData. 2011. The midata vision of consumer empowerment. (2011). Retrieved October 12, 2018 from <https://www.gov.uk/government/news/the-midata-vision-of-consumer-empowerment>
- [26] Nextcloud. 2016. Protecting your data. (Jun 2016). Retrieved October 12, 2018 from <https://nextcloud.com>
- [27] European Parliament. 2016. General Data Protection Regulation. Law. (27 April 2016). Retrieved October 12, 2018 from <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679>
- [28] Raluca Ada Popa, Andrew J Blumberg, Hari Balakrishnan, and Frank H Li. 2011. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 653–666.
- [29] Christian Priebe, Kapil Vaswani, and Manuel Costa. 2018. EnclaveDB: A Secure Database using SGX. In *EnclaveDB: A Secure Database using SGX*. IEEE, 0.
- [30] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. 2001. *A scalable content-addressable network*. Vol. 31. ACM.
- [31] Eyad Saleh, Ahmad Alsa'deh, Ahmad Kayed, and Christoph Meinel. 2016. Processing over encrypted data: between theory and practice. *ACM SIGMOD Record* 45, 3 (2016), 5–16.
- [32] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
- [33] Solid. 2018. Solid empowers users and organizations to separate their data from the applications that use it. (2018). Retrieved October 12, 2018 from <https://solid.inrupt.com/>
- [34] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review* 31, 4 (2001), 149–160.
- [35] ARM Security Technology. 2008. *Building a Secure System using TrustZone Technology*. Technical Report. ARM.
- [36] Dai Hai Ton That, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. 2016. PAMPAS: Privacy-Aware Mobile Participatory Sensing Using Secure Probes. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*. ACM, 4.
- [37] Quoc-Cuong To, Benjamin Nguyen, and Philippe Pucheral. 2016. Private and scalable execution of SQL aggregates on a secure decentralized architecture. *ACM Transactions on Database Systems (TODS)* 41, 3 (2016), 16.
- [38] J. C. Tomàs, B. Amann, N. Travers, and D. Vodislav. 2011. RoSeS: a continuous query processor for large-scale RSS filtering and aggregation. In *Proc. of the 20th ACM Conf. on Information and Knowledge Management*. 2549–2552.
- [39] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. 2011. A survey of DHT security techniques. *ACM Computing Surveys (CSUR)* 43, 2 (2011), 8.
- [40] Qiyang Wang and Nikita Borisov. 2012. Octopus: A secure and anonymous DHT lookup. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 325–334.

Appendix G

PAMPAS: Privacy-Aware Mobile Participatory Sensing Using Secure Probes

PAMPAS: Privacy-Aware Mobile Participatory Sensing Using Secure Probes

Dai Hai Ton That[†], Iulian Sandu Popa^{†,‡}, Karine Zeitouni[†], Cristian Borcea^{*}

[†]DAVID Laboratory - University of Versailles Saint-Quentin, Versailles, France

[‡]INRIA Saclay-Ile-de-France, Palaiseau, France

^{*}Department of Computer Science, New Jersey Institute of Technology, Newark, New Jersey, USA
{dai-hai.ton-that,iulian.sandu-popa,karine.zeitouni}@uvsq.fr, borcea@njit.edu

ABSTRACT

Mobile participatory sensing could be used in many applications such as vehicular traffic monitoring, pollution tracking, or even health surveying. However, its success depends on finding a solution for querying large numbers of users which protects user location privacy and works in real-time. This paper presents PAMPAS, a privacy-aware mobile distributed system for efficient data aggregation in mobile participatory sensing. In PAMPAS, mobile devices enhanced with secure hardware, called secure probes (SPs), perform distributed query processing, while preventing users from accessing other users' data. A supporting server infrastructure (SSI) coordinates the inter-SP communication and the computation tasks executed on SPs. PAMPAS ensures that SSI cannot link the location reported by SPs to the user identities even if SSI has additional background information. In addition to its novel system architecture, PAMPAS also proposes two new protocols for privacy-aware location-based aggregation and adaptive spatial partitioning of SPs that work efficiently on resource-constrained SPs. Our experimental results and security analysis demonstrate that these protocols are able to collect the data, aggregate them, and share statistics or derived models in real-time, without any location privacy leakage.

CCS Concepts

•Information systems → Mobile information processing systems; •Security and privacy → Security in hardware;

Keywords

Location privacy; secure protocol; distributed architecture; mobile participatory sensing; spatial aggregates

1. INTRODUCTION

There is an increasing interest in mobile participatory sensing for urban monitoring, which appears to be a bet-

ter alternative to traditional infrastructure-based sensing to cope with the high installation and maintenance costs, as well as the coverage limitation. Many projects have been conducted recently around the world - or are still ongoing - in the area of environmental participatory sensing [15], such as Citi-Sense in Oslo, CamMobSens at Cambridge, MetroSense at Dartmouth, and OpenSense in Switzerland. Also, many applications that exploit the sensing features of smartphones are already available. Examples include community based traffic monitoring (e.g., Waze¹, or Navigon²), finding available parking spaces or noise mapping [9]. In addition, the emerging lightweight low-cost sensors are changing the paradigm of environmental and health monitoring³, and allow measuring in real-time the individual exposure to environmental risk factors or the propagation of an epidemic.

In these scenarios, the community members act as mobile probes and contribute to spatial aggregate statistics, which in turn, benefit the whole community, e.g., dynamic traffic navigation or air quality mapping and alerts. Various statistics are of interest: basic count and density, average of reported measures by location and time, or more complex geo-statistical operations such as spatial interpolation [14]. Unfortunately, most current mobile participatory sensing systems (MPSS) require users to reveal their locations to trusted monitoring servers, which raises serious privacy concerns because user identity could be determined based on several locations that are linked to the same user [7]. We should stress that, even if users might trust a centralized service, privacy violation examples are legions (see for example DataLossDB.org) coming from negligence, abusive use, internal or external attacks, and such violations affect even the most secured servers. In addition to location, sensing data reported by users could be privacy-sensitive as well. These privacy issues prevent a wide adoption of MPSS.

Several works consider the MPSS privacy problem such as [9], [5], [8], [17], [18]. However, most approaches require trusting a proxy server [8], [18], while others are too costly [9], [5], or sacrifice sensing accuracy for privacy [17].

Hence, providing a high-quality MPSS, while protecting the users' privacy, is still a challenge. Recently, the emergence of personal secure devices has opened new perspectives in personal data protection. Be it a secure portable token [1], [19] communicating with the user's smartphone or plugged inside it (e.g., Google Vault⁴), a tamper-resistant hardware

¹<http://www.waze.com>

²<http://navigon.com>

³<http://www.epa.gov/heads/airsensortoolbox/>

⁴<http://www.cnet.com/news/googles-project-vault-is-a>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SSDBM '16, July 18-20, Budapest, Hungary

© 2016 ACM. ISBN 978-1-4503-2138-9... \$15.00

DOI: 10.1145/1235

security module securing the on-board computer of a vehicle [10], or the secure TrustZone CPU [2] of the ARM cortex-A series equipping most of mobile devices today, all such secure devices offer tangible, hardware-based security guarantees. We leverage their secure data processing capability in a distributed, privacy-by-design architecture, providing an alternative to the traditional server-centric architecture. Our belief is that, similar to TrustZone CPU, such secure devices will become ubiquitous in the near future, equipping by default users' mobile devices. As such, there will be no need for users to buy and connect external hardware to participate in MPSS applications.

This paper presents PAMPAS, a Privacy-Aware Mobile Participatory Sensing system for efficient mobile distributed query processing in the context of MPSS. The novelty of PAMPAS is two-fold: (1) it provides a system architecture that protects users' location privacy by preventing location tracking from any third-party server; and (2) it provides efficient aggregation protocols that satisfy the MPSS real-time constraints in spite of the resource limitations of secure devices. The privacy guarantee gives users strong incentives for participation [11], in addition to the benefits they get from MPSS applications. In PAMPAS, all participants have a mobile device enhanced with secure hardware (i.e., any of the types described above), called a secure probe (SP). SPs act as probes for the target phenomenon, perform distributed query processing, and share the results with the users. The secure hardware prevents users from accessing other users' data during the distributed computation. Secure probes exchange data in encrypted form with help from a supporting server infrastructure (SSI). To provide real-time results, PAMPAS employs efficient, parallel, location-based aggregation protocols which partition the probes according to their geographic distribution. The construction and the maintenance of these partitions aim at reducing and balancing the workloads on worker SPs, while precluding the SSI from doing location-based inference attacks against the participants.

We implemented and validated PAMPAS using representative secure hardware platforms. We used two applications for experiments, traffic and noise monitoring, with both real and synthetic datasets representing small and medium-size cities. Using these applications, we compared PAMPAS with a state-of-the-art secure aggregation protocol described in [19]. The experimental results show that PAMPAS outperforms this protocol in terms of latency and scalability, which translates to much lower resource utilization at the user side.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes the system architecture of PAMPAS, the threat model, and the protocol requirements. Section 4 presents the location-based global aggregation protocol, and Section 5 describes the privacy-aware probe partitioning protocol. The security analysis is presented in Section 6, while the experimental results are shown in Section 7. We conclude the paper in Section 8.

2. RELATED WORK

Traditional system architectures used in MPSS such as [8], [18] rely on a centralized server to collect data from mobile participants, process it, and publish the results. This server-centric model is straightforward and easy to deploy,

security-chip-disguised-as-an-micro-sd-card/

run, and maintain. However, this basic approach also raises serious privacy concerns and prevents a wide adoption of MPSS. An attacker who is able to link several location reports to the same user can then determine the identity of the user by leveraging, for example, background information (e.g., user home address). Thus, an attacker can identify the MPSS participants and infer their personal habits and activities [7]. In addition to location which is normally included in MPSS reports, the sensing data reported by users could be privacy-sensitive as well. The works that address this issue belong to three classes: (1) server-centric architecture and (2) cryptographic protocols for MPSS, and (3) secure hardware devices in other contexts.

Server-centric approaches. Virtual trip lines (VTLs) [12] deal with the privacy issue by distributing the traffic monitoring service implementation across several specialized servers and by providing a spatio-temporal cloaking of the users under the VTLs. Although the attack of a single system component prevents linking the identity and location of the users, choosing privacy-insensitive locations for VTLs is tricky and limits the traffic information to a part of the road network. Also, the problem of multi-component attack (or collusion) remains, as well as the high cost of building such a complex system distributed over several components. SpotMe [17] proposes a different approach consisting in mixing real user's location with fake locations before posting them to a central server. Then, the server estimates the aggregated user locations by using probability theory. However, the estimation errors can be important (around 20%), while the number of observed spatial units cannot exceed a few hundreds. Also, SpotMe involves higher communication costs because of the large number of fake locations, while linkability may still be a problem for users who send many consecutive location updates, which limits the usability of this approach to sporadic updates.

By employing a fully decentralized architecture for computation, PAMPAS avoids all the above listed problems. Moreover, the trust is enforced by using cheap but highly secure, tamper-resistant hardware at the user side.

Cryptographic approaches. Another way to protect the users' privacy is to use secure cryptographic protocols [5], [9], [16]. Typically, the cryptographic solutions are based on homomorphic encryption schemes allowing a central-server [16] or the users [9] to aggregate the samples directly on the cyphertext. However, the cryptographic methods have to face two major limitations. First, homomorphic encryption only allows the computation of basic aggregate functions (e.g., count, average, standard deviation), while more advanced functions require fully homomorphic encryption schemes, which are not computationally feasible today. Second, even with the basic aggregate functions, the cryptographic protocols can incur a large computation and communication cost [5], [16]. Hence, the existing works typically limit the size of the monitored space (e.g., the number of roads) and the monitoring frequency. Therefore, such solutions cannot meet the scalability and the real-time requirements of MPSS at the same time, and are not generic w.r.t. the type of aggregate function.

Secure hardware approaches. Recent works have also proposed the use of secure hardware at the user-side [1], [19]. The trust in such a distributed architecture in which all computation is done by user devices arises from two sources: (i) the decentralization, i.e., there is no central-server to be

trusted or to be exposed to attacks having a large benefit/cost ratio; (ii) the (tamper-resistant) secure hardware at the user-side, which protects the devices against physical attacks (even from the device holder).

In [1], Allard et al. propose METAP, a privacy-preserving data publishing protocol in the context of an architecture composed of low power secure devices and a powerful but un-trusted server in order to release sanitized data to third parties. However, this data publishing protocol does not consider the case of spatiotemporal sensed values and cannot be used in participatory sensing aggregations. To et al. [19] propose a similar architecture, but consider the problem of executing SQL queries over a distributed database without revealing any sensitive information to central servers. Considered in our context, this protocol incurs high computation and communication costs because of the specificity of MPSS aggregates (e.g., the aggregate groups are locations, there is a high number of such groups, the computation is continuous and should follow the data generation frequency, the aggregate functions can be complex such as spatial interpolation).

PAMPAS shares the idea of employing a user-centric decentralized architecture with the above mentioned works. However, unlike existing protocols, its secure aggregation protocol is adapted to MPSS requirements, i.e., high dynamics of the participants, real-time constraints for computation, complexity of the aggregate statistics, and low resource utilization.

A centralized solution based on secure hardware could also be devised using recent proposals to ensure shielded application execution over untrusted servers. For example, Haven [3] extends the hardware level protection features provided by the Intel SGX architecture from code snippets to the entire OS. But there are limitations: this solution slows down the computation substantially; the entire security architecture depends on the chip manufacturer’s ability to protect the secret keys; programmers will miss certain features, such as process creation, that are not supported.

3. SYSTEM OVERVIEW

This section presents the system architecture of PAMPAS, the threat model in our system, and the data and computation model of the system. Based on these elements, we derive the requirements for the PAMPAS protocols.

3.1 System Architecture

PAMPAS relies on a hybrid architecture combining secure elements at the user side (secure probes – SP) and a supporting server infrastructure (SSI) that enables secure exchange of messages between the mobile users. SPs and SSI jointly run two protocols to exchange sensed sample updates, continuously compute the spatially aggregated results, and periodically partition SPs according to their location. This architecture fully protects the users’ privacy w.r.t. the SSI. Figure 1 shows the general architecture of our system in the context of traffic monitoring.

Compared to a purely decentralized peer-to-peer (P2P) architecture, this hybrid architecture has the salient advantage of not consuming any resources from the participants to maintain the P2P overlay, which is important given the low resources and availability of the user devices. In addition, it exchanges messages between SPs in $O(1)$ hops as opposed to the typical $O(\log N)$ hops in P2P networks.

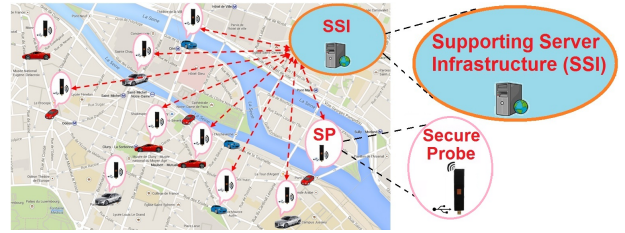


Figure 1: System architecture

Secure Probe (SP). Each user holds a secure portable device, which can be implemented by any type of (tamper-resistant) secure devices flourishing today (see Figure 2) and described in Section 1. Whatever its commercial name and form factor, a secure device, called secure probe (SP) hereafter, embeds at least a secure micro-controller (MCU) for computation (e.g., a smart card chip) connected to a large NAND Flash memory for data storage (e.g., an SD card). An SP plays three roles: (i) mobile probe, (ii) processing node, and (iii) query issuer. The SP sends encrypted samples (containing spatiotemporal sensed measures) to SSI, participates in the data aggregation, and receives the final results from other SPs with the help of SSI. Given their high-level of security, SPs are considered trusted in our system. However, this feature comes at a price. The MCU usually has a low power CPU and a tiny RAM (a few tens of KB). In addition, SPs have low availability since they can be connected/disconnected as required by the users. Therefore, all the computation and communication with the SPs have to be highly optimized.



Figure 2: Examples of secure tokens

Supporting Server Infrastructure (SSI). Different from the typical server-centric architecture, the SSI in our system acts only as a coordinator for exchanging messages between the SPs and for temporary storage purposes. Since the SSI is not trusted, all the temporary results stored in the SSI are encrypted using non-deterministic encryption.

In conclusion, the security and privacy in PAMPAS arise from the combination of secure hardware with a high degree of distribution of the architecture (i.e., all computations are executed by some of the SPs). The challenge is then to be able to continuously compute any type of aggregate functions in real-time in this user-centric architecture given the low resources of the SPs.

3.2 Threat model

The attackers in PAMPAS could be either users or the owners of SSI. Their goal is to collect private user information (e.g., location or sensing data). Using this private information, attackers can determine the user identities and learn their activities and behaviors. Our goal is to ensure that users cannot read the raw data reported by other users. The SSI must not be able to read the user raw data. Also, the SSI must not be able to infer any additional location in-

formation about the participants more than it already knows or could be inferred from the aggregate result. Hence, the scope of PAMPAS is to fully protect the raw data and the aggregation process, and does not consider the privacy exposure risks that arise from analysing the aggregate results, which is a complementary aspect of this work.

Even though the users are untrusted, we assume that all the SPs are trusted, which is reasonable considering that the tamper-resistance of the MCU provides a high level of protection against physical and side-channel attacks, and in particular for the data residing in RAM since the RAM memory is located inside the MCU. We also assume that the hardware manufacturer is trusted and protects the secrets embedded in SPs. In addition, all the persistently stored data in the NAND Flash is cryptographically protected.

Furthermore, we assume an honest-but-curious threat model, i.e., the SSI obeys the protocol it is supposed to do, but may try to infer anything it can from the data or behaviors it sees. Considering a malicious SSI (i.e., the server tampers with the protocol, e.g., by dropping messages to infer more information) is of little interest, since a malicious SSI can be easily detected (e.g., the SPs that aggregate the data verify if their own samples are present in the data sent by the server) leading to critical financial/legal consequences for the service.

Finally, we assume that the communication between SPs and SSI is anonymous, e.g., by using a proxy forwarder or an anonymization network (e.g., Tor). We assume such systems are able to hide the packet origin from an adversary, so that privacy cannot be compromised by a malicious server searching to recognize the origin of the uploaded messages. Let us emphasize that IP anonymity is not enough to protect the user privacy in MPSS because identity information could be determined from the location and sensing data.

3.3 Data and Computation Models

Data model. PAMPAS is designed to be generic with respect to the type of computation required by participatory sensing applications. In most cases, such applications require the aggregation of geo-localized and time-stamped sensed values collected by the sensing devices of the participants. Therefore, a participant's device periodically generates an update in the form $sample = (location, time, value)$, which is encrypted and sent to the SSI. PAMPAS does not impose any restriction on the generation frequency of samples, which may depend on the application sample generation policy. However, the system should be efficient and scalable for a large number of participants and a high generation frequency of samples. Also, the participants' privacy should be fully protected independent of the number and spatiotemporal distribution of the samples. Furthermore, PAMPAS considers two types of locations corresponding to the two typical types of movements of users: (i) free movements in the two-dimensional space, i.e., $location = (x, y)$; (ii) movements constrained by a transportation network (e.g., road or railroad network), i.e., $location = (rid, pos)$, where rid is the road identifier and pos is the relative position on the road. Finally, the $value$ corresponds to the sensed measure (e.g., traffic speed, noise level, etc.).

Query model. Given a stream of $samples$ and an aggregate function, PAMPAS produces a spatiotemporal aggregation of the sample stream such as the stream-SQL-like [13] query formulation in Figure 3. The aggregation is tempo-

ral since the result is computed *continuously* over time as long as it is required or whenever the number of participants exceeds some predefined threshold. In this way, the spatiotemporal evolution of the measure of interest is monitored over time. To this end, PAMPAS divides the stream using a sliding time window (see Figure 3) and computes an aggregate result based on all the samples generated in the time window. The final aggregation result is a spatial function representing the evolution in space of the observed measure in the respective time window. For instance, the result can be the noise heat-map in the covering area of a city or the average travel time in a road network. As with the duration of observation, we do not impose any restrictions regarding the extent of the observed space.

```
SELECT SpatialAggregate (value), [COUNT(*)]
FROM ParticipatorySensingStream
(WINDOW x seconds SLIDE x seconds)
[WHERE condition]
GROUP BY spatialUnit
[HAVING predicateOnSpatialAggregate]
```

Figure 3: Spatial-temporal aggregates in PAMPAS

Spatial units. As shown in the above query, spatial aggregates are based on a discrete referential space, i.e., a finite set of spatial units. Without loss of generality, we consider two types of referential spaces corresponding to the two types of users' movements. In the case of free movement, we consider a uniform grid and each grid cell corresponds to a *spatial unit*. The size of the units is determined based on the application requirements, space size, number of participants, etc. In the case of constrained movement by a transportation network, we consider that a *spatial unit* corresponds to a network (road) segment, i.e., the network path connecting two adjacent network nodes (e.g., the road segment between two intersections). In both cases, the number of *spatial units* can be large (e.g., hundreds of thousands). The COUNT in the query model is optional and is required in the aggregation protocol to check the probes partitioning.

Aggregate functions. PAMPAS can compute most types of aggregate statistics required by participatory sensing applications. Practically, our system can compute in real-time any type of function having reasonable time and space complexity given the relative low CPU power and little RAM of the SP. For illustrative purpose, we consider three classes of functions in this paper: (i) *Typical algebraic functions*: count, sum, average, standard deviation. Such aggregate functions are the most popular in the works related to participatory sensing [9], [5], [16]. These functions allow for example to compute the average travel time or the traffic density in a road network; (ii) *Specific functions*: inverse distance weighting (IDW). For instance, an application monitoring the noise pollution in the city could use the IDW function to compute a heat-map of the noise level in the entire space [14]; (iii) *Holistic functions*: median, percentile, top-k. Such functions are also frequently used in statistical computations. Their particularity is that the computation of the result requires accessing the entire sample set and cannot be achieved incrementally by accessing only subsets of samples as with the previous two classes of functions.

An important observation is that cryptographic solutions based on homomorphic encryption cannot be applied for spe-

Table 1: Notations used in the algorithms

Notation	Description
G_i	Identifier of probe group i
E_k	Symmetric deterministic encryption
nE_k	Symmetric non-deterministic encryption
E_k^{-1}	Symmetric deterministic decryption
nE_k^{-1}	Symmetric non-deterministic decryption
$P_{G_i}^{fake}$	Probability to send a fake message in group i
N	Number of spatial units
N_G	Number of probe groups
QI_{comm}	Degradation factor of the communication time
QI_{comp}	Degradation factor of the computation time
$Comp_time_i$	Computation time for the group i

cific or holistic functions (see Section 2). Also, the holistic functions cannot be computed efficiently in a distributed architecture by the secure protocol proposed in [19] (as shown in Section 7).

3.4 Protocol Requirements

In the light of the above description of the proposed user-centric architecture, the PAMPAS protocols have to deal with the following challenges: (i) *Privacy*: By keeping all the sensitive data in the SPs, the adopted user-centric architecture matches this requirement in contrast with a server-centric architecture. In short, the computation protocol should not reveal to the SSI any additional information about the participants' paths besides what the SSI can infer from the aggregate result. (ii) *Generality*: the protocols should be able to compute any type of function over the spatiotemporal sensed measures by the mobile users and covering a large observation space. This is different from the works based on cryptographic approaches in which, typically, only basic computation (e.g., simple aggregates like sum, average) can be achieved and only in specific locations over limited periods of time. (iii) *Efficiency*: the protocols should be highly efficient to be able to *continuously* compute the aggregate results in *real-time* with very *limited resources*. Indeed, for economic and security reasons, the SPs used for data processing have low resources and limited availability. Hence, it is necessary to minimize the computation and communication costs of the PAMPAS protocols. (iv) *Scalability*: the protocols should allow PAMPAS to scale to a large number of participants (e.g., up to millions of users), high sampling frequencies, and very large regions. (v) *Accuracy*: PAMPAS should continuously reflect the sensed measures with good precision. In other words, protecting the users' privacy should not impact the accuracy of the aggregate result computed by the protocols.

4. GLOBAL AGGREGATION PROTOCOL

The global privacy-preserving protocol in PAMPAS consists in three phases that are repeatedly executed in pipeline (see Figure 4). First, the SSI collects all the sensing updates sent by the participants for a period equal to the sliding time window (i.e., the *collection period*). Each update is encrypted using symmetric non-deterministic encryption so

Algorithm 1: PAMPAS Protocol at the SSI-side

```

1 collection_period()
2   /* Receive encrypted updates from SPs */
3   while (true) do
4     message = ( $E_k(G_i), nE_k(\text{sample})$ )
5     store(message) →
6     list[ $E_k(G_i)$ ][currentTimeWindow]
7
6 processing_period()
7   foreach  $i$  in { $E_k(G_i)$ } do
8     /* feed in parallel the randomly selected SPs */
9     randomly select  $SP_i \in E_k(G_i)$ 
10    while
11    message ← list[ $E_k(G_i)$ ][lastTimeWindow] do
12    send(message,  $SP_i$ )
13
12 foreach  $i$  in { $E_k(G_i)$ } do
13   /*Receive the final results from worker SPs*/
14   enc_result_i^{final} = ( $E_k(G_i), nE_k(\text{result})$ )
15
15 delivery_period()
16 foreach  $i$  in { $E_k(G_i)$ } do
17   /*Push result_i to all requesting SPs*/
18   multicast(enc_result_i^{final}, { $SP_k$ })

```

that the SSI cannot gain any knowledge from these updates. All the SPs share a secret key, which is renewed periodically to increase security. The key is generated randomly by a randomly chosen SP. To distribute the secret key, we assume the users authenticate using a typical PKI infrastructure, i.e., a certificate is embedded in each user secure hardware. Whenever a new SP connects to the system, it authenticates using its certificate. Then, the SP randomly contacts another connected SP, which sends back the current shared secret key encrypted with the public key of this newly connected SP.

The shared secret key is used by the SPs to symmetrically encrypt the update messages (e.g., by using AES encryption) so that any SP can decrypt messages and aggregate the data. Note that, although an SP can decrypt the updates, a user is not allowed to access the decrypted data in her SP and that the tamper-resistant hardware protects the transiting data from the user. Therefore, as for the SSI, the users have access only to the final results and not to the raw data.

At the end of the collection period, the SSI triggers the *processing period*. In this phase, only a small subset of SPs, which are randomly selected by the SSI, are involved. The SSI partitions the collected samples such that the number of updates in a partition can fit the RAM resources of an SP (otherwise, the persistent Flash storage of the SP has to be used incurring a much higher computation cost). Then, each sample partition is sent to an SP, which computes a partial aggregate result for the received updates. The encrypted results are sent back to the SSI. Finally, the *delivery period* consists of delivering the current partial aggregate results to the queriers. Each querier needs to perform the final aggregation of these partial results, which is merely a concatenation of the demanded partial results.

Algorithms 1 and 2 give the detailed description of the operations executed by the SSI and the SPs respectively. In the following, we denote by E_k and nE_k the symmetric

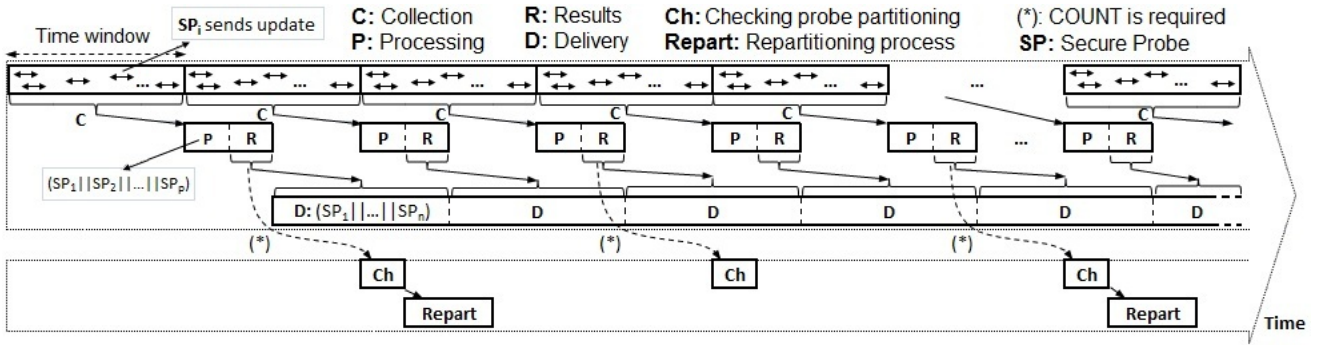


Figure 4: Workflow representation of the global protocol in PAMPAS

Algorithm 2: PAMPAS Protocol at the SP-side

```

1 collection_period(): /* for all SPs */
2   /* Generate and send the sensing update:
   update( $G_i$ , sample) */
3    $message = (E_k(G_i), nE_k(sample))$ 
4   send(message, SSI)
5   /* Send a fake sample to the SSI with probability
    $P_{G_i}^{fake}$  */
6   if  $rand(0, 100) \geq P_{G_i}^{fake}$  then
7      $fake\_message = (E_k(G_i), nE_k(fake\_sample))$ 
8     send(fake\_message, SSI)
9 processing_period(): /* only for the selected SPs,
   one for each  $G_i$  */
10  while  $message = receive(SS I)$  do
11     $sample = nE_k^{-1}(message)$ 
12     $result = result \oplus sample$ 
13     $enc\_result_i^{final} = (E_k(G_i), nE_k(result))$ 
14    send(enc\_result_i^{final}, SSI)
15 delivery_period(): /* for all SPs */
16  /* Pull the results for required  $\{G_i\}$  from the SSI */
17  foreach  $i$  in  $\{E_k(G_i)\}$  do
18    send\_request( $E_k(G_i)$ , SSI)
19     $result_i^{final} = nE_k^{-1}(receive(SS I))$ 

```

deterministic and non-deterministic encryption with the key k , and by E_k^{-1} and nE_k^{-1} the opposite decryption operations while G_i indicates the identifier of group i . All the notations used in the algorithms are listed in Table 1.

To address the performance limitations of the existing protocols [19] (see Section 2), the aggregation protocol in PAMPAS groups the participants based on their location, which permits processing together the generated samples in a group by a single SP. To this end, the users also send the deterministically encrypted value of the spatial unit they are currently located in, in addition to the non-deterministically encrypted value of the sample, i.e., $message = (E_k(groupID), nE_k(sample))$ (Algorithm 1, lines 4-5 and Algorithm 2, lines 3-4). Consequently, the SSI can group the messages based on the encrypted unit number and then send each group of samples to a different SP for aggregation (lines 7-11 in Algorithm 1 and lines 10-12 in Algorithm 2). By doing so, the advantage is manifold. First, the processing period is guar-

anteed to terminate in a single iteration, since each involved SP produces directly the aggregation result corresponding to a spatial unit. This greatly improves both the computation and the communication cost of the aggregation process. Second, data processing by an SP is also efficient since only one aggregate is computed, which greatly reduces the RAM requirements and avoids/reduces the usage of the persistent storage. Third, the final aggregate result is also partitioned and the queriers can demand the results only for specific spatial units, which further improves the communication cost. Furthermore, in order to avoid leaking information regarding the spatial distribution of users, the SPs also generate and send fake messages to the SSI (see Algorithm 2, lines 6-8). The rational and detailed explanation for this technique are discussed in the next section.

However, despite all these benefits, the above approach has one fundamental shortcoming originating from the skewed spatial distribution of the participants. Although the exact location of the updates and the unit ID are hidden, the SSI knows the number of participants in each spatial unit. If the SSI has access to side information about the spatial distribution of the users (e.g., global traffic density information), it may use this information to infer the (approximate) location of the participants and compromise their privacy.

5. PROBE PARTITIONING PROTOCOL

To counter the privacy threats that are rooted in the skewed spatial distribution of the participants, PAMPAS continuously partitions the set of probes based on their current location and the spatial units of the query. Similar to the global aggregation protocol, this privacy-aware partitioning protocol is executed by SPs. The idea is to group SPs located in adjacent spatial units such that the resulted probe groups have approximately the same size. Therefore, in PAMPAS a group G_i covers several spatial units (as defined in Section 3.3) and includes all the SPs in these units.

The probe partitioning has to be recomputed periodically to keep the groups balanced since the users' distribution in space changes over time. Moreover, the groups should contain users located in closely situated spatial units to maximize the lifetime of a partitioning. The challenge is to implement a partitioning algorithm that can be executed periodically at SPs because the typical spatial partitioning algorithms are much too costly to be considered in our context (i.e., limited-resources SPs).

Our algorithm is based on the following idea. We use a

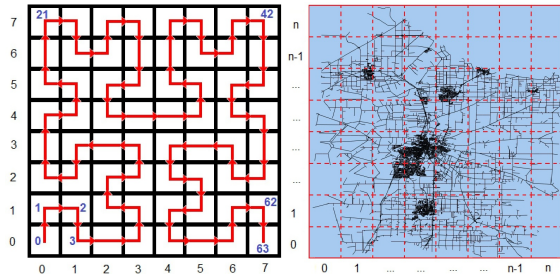


Figure 5: Hilbert indexing of spatial units

space-filling curve to index the spatial units of the application query. A space-filling curve has the property to map a multidimensional space to a one-dimensional space such that, for two objects that are close in the original space, there is a high probability that they will be close in the mapped target space. Then, we sort the spatial units on the space-filing curve index. Once sorted, an approximate balanced grouping can be checked and computed in $O(N_G)$ space complexity and $O(N)$ time complexity, where N_G is the number of probe groups, and N is the number of spatial units.

Indexing the spatial units. In our system, we use Hilbert curves, but other types of space-filling curves can be used as well to index the spatial units considered by the participatory sensing application (e.g., z-curves). In the case of free movement, the indexing is straightforward since the space is already partitioned with a uniform grid (see Figure 5 left). Then, we cover the grid cells with the Hilbert curve corresponding to the grid granularity and label each cell with the obtained Hilbert index. In the case of constrained movement, the indexing requires two steps. First, we cover the transportation network with a uniform grid (see Figure 5 right). The grid granularity is chosen such that the number of network segments (see Section 3.3) intersecting with a grid cell is low for most of the cells. Then, the grid cells are indexed with a Hilbert curve and each network segment is labeled with the Hilbert index of the cell containing the segment center. In case several segments are contained by a cell, the segments are sorted by the x-coordinate and the y-coordinate of their centers and labeled accordingly. Once the spatial units are indexed, they are sorted on the index value and the sorted unit vector is broadcasted to all the participants to be used in the probe partitioning phase.

Checking and repartitioning the probe grouping. Periodically, our system verifies if the current probes partitioning is still balanced with respect to the number of probes in each group. The verification frequency depends on the dynamics of users in space. In PAMPAS, the checking and repartitioning processes can be executed often (i.e., every few seconds) due to their low cost. When a partitioning checking is triggered, the system computes a *count* aggregate in addition to the application aggregate function (see Figure 3), which gives the actual number of users (SPs) in all the spatial units. The count aggregate result is then pushed to an SP randomly chosen by the SSI. The checker SP decrypts the results and updates the weights⁵ of the sorted spatial unit vector (lines 4-7 in Algorithm 3). This operation has $O(N)$ complexity assuming that a small index con-

⁵The weight is the number of probes in a spatial unit.

Algorithm 3: Checking probe partitioning (SP-side)

```

1 check_probe_partitioning() /* one randomly
   selected SP */
2 /* Pull all the results from the SSI */
3 foreach  $i$  in  $\{E_k(G_i)\}$  do
4   send_request( $E_k(G_i)$ , SSI)
5    $enc\_result_i^{final} = receive(SSI)$ 
6    $allCounts[G_i][\ ] \leftarrow E_k^{-1}(enc\_result_i^{final})$ 
7   update locally stored counts for spatial units
   /* also required to compute the probability to
   generate fake samples */
8    $compute\_weights[G_i] = SUM(allCounts[G_i][\ ])$ 
9   compute standard_deviation(weights)
10  if  $standard\_deviation(weights) < threshold$  then
11    send_for_broadcast( $nE_k(allCounts)$ , SSI)
12  else
13    execute probes_repartitioning()

```

taining the partitions frontiers is kept in memory by the SP (which requires only N_G Flash addresses to be kept in RAM). At the same time, the SP computes in memory the count by group (since the groups are sent one by one by the SSI, line 8 in Algorithm 3) and compares the counts. If the balancing of the current probes partitioning is within the predefined limits, the checker SP sends the current values to all the other SPs (i.e., exchanged encrypted through SSI), which update the weights of the spatial units with the new count values. Otherwise, the checker SP computes a new partitioning.

Once the sorted vector of spatial units is updated with the new weight values, the probe repartitioning can be efficiently computed in $O(N)$ and $O(N_G)$ time and space complexity respectively (see Algorithm 4). To set the partition borders we use a greedy algorithm, which adds spatial units to a group as long as the total weight of the group is lower than a threshold value (lines 12-16 in Algorithm 4). The threshold is computed as the ratio between the total number of probes and the number of groups (line 10 in Algorithm 4), and represents the average number of users per group. The partitioning result is a list of N_G *milestones* indicating the group borders in the sorted index of spatial units (line 15 in Algorithm 4). The result is then encrypted and delivered, through SSI, to all users, which update their partitioning data and generate new samples accordingly starting from the next computation window.

The proposed probes partitioning algorithm has low complexity and can be efficiently executed even with the low resources of an SP. However, the partitioning algorithm cannot guarantee a certain degree of balancing of the partition weights. Yet, the partitioning balancing is required to avoid leaking any information regarding the spatial distribution of users. To deal with this problem, the SPs generate fake samples in all the probe groups having a number of users lower than the maximum size group. Therefore, in the collection period of each computing time window, an SP sends probabilistically a dummy sample in addition to the real sample. The probability to send a fake sample is proportional to the difference between the maximum size group and the number of users in the SP's group, and inversely proportional to the

Algorithm 4: Repartitioning process (SP-side)

```
1 PROBE_REPARTITIONING()/* one randomly
   selected SP */
2   compute  $QI_{comp}$  and  $QI_{comm}$  for current  $N_G$ 
3   while true do
4     /* adjust the number of groups  $N_G$  */
5     if  $QI_{comp} > QI_{comm}$  then
6        $tN_G = 2 * N_G$ 
7     else
8        $tN_G = N_G/2$ 
9     /* repartition for  $tN_G$  */
10     $avgGroupWeight = \text{SUM}(allCounts[])/tN_G$ 
11     $currentGroupWeight = 0$ 
12    for  $i = 0$  to  $N - 1$  do
13       $currentGroupWeight += allCounts[i]$ 
14      if  $currentGroupWeight \approx avgGroupWeight$ 
15        then
16           $newPartitionMilestones[].add(i)$ 
17           $currentGroupWeight = 0$ 
18      /* check if the new partitioning for  $tN_G$  is
19       better than for  $N_G$ */
20      compute  $tQI_{comp}$  and  $tQI_{comm}$  for  $tN_G$ 
21      if  $tQI_{comp} + tQI_{comm} < QI_{comp} + QI_{comm}$ 
22        then
23           $N_G = tN_G$ ;  $QI_{comp} = tQI_{comp}$ 
24           $QI_{comm} = tQI_{comm}$ 
25        else
26          break
27     $message = allCounts[] || newPartitionMilestones[]$ 
28     $send\_for\_broadcast(nE_k(message), SSI)$ 
```

number of users in the group (see Algorithm 2, lines 6-8). The same approach is used to hide the number of spatial units in each group. At the end of the aggregation phase, each aggregating SP adds to the result a number of fake values equal to the difference between the maximum number of units in the groups and the number of units in the current group. In this way, all the partial aggregate results received by the SSI have the same size and the SSI cannot infer the number of cells in any group. Note that the fake values are filtered out by the worker or querier SPs and therefore have no impact on the accuracy of the results.

$$QI_{comp} = \text{Max}_{i=1, N_G} [Comp_time_i] - \text{Max}_{j=1, N} [Comp_time_j] \quad (1)$$

$$QI_{comm} = \frac{\text{size}(sample)}{\text{bandwidth}} \sum_{i=1}^{N_G} \{ \text{Max}_{j=1, N_G} [Count_j(probes)] - \text{Count}_i(probes) \} + \frac{\text{size}(sample)}{\text{bandwidth}} \sum_{i=1}^{N_G} \{ \text{Max}_{j=1, N_G} [Count_j(spatialUnits)] - \text{Count}_i(spatialUnits) \} \quad (2)$$

Choosing the Number of Probe Groups. The cost of the aggregation protocol is composed of the computation

cost at the SP side and the communication cost between the SSI and the SP. The number of probe groups impacts both the computation and the communication costs. Specifically, the computation cost decreases with the increase in the number of groups and attains the minimum value when the number of groups is equal to the number of spatial cells, i.e., an SP is used to aggregate the samples for each spatial unit. But increasing the number of groups leads to a higher imbalance in the groups' weights, which in turn requires injecting more fake samples and enlarges the communication cost. Therefore, modifying the number of groups has opposite effect on the computation and the communication cost.

PAMPAS computes two quality indicators to measure the impact of the number of groups on the computation and communication costs, i.e., QI_{comp} and QI_{comm} , as defined by Formulas (1) and (2). QI_{comp} estimates the degradation of the computation time at the SP side generated by the fact that several spatial cell aggregates are delegated to one SP instead of using one SP for each cell. Estimating the computation time is fairly simple since the time is typically linear with the number of samples to be processed by the SP, assuming that the aggregation can be entirely processed in RAM. However, the cost model can be extended to the case in which it is required to access the secondary storage. QI_{comm} estimates the degradation of the communication cost caused by the imbalance of the group weights. The first term indicates the overhead incurred by the fake samples generated to balance the groups, while the second term measures the overhead of generating fake results to balance the number of aggregate results in each group.

Each time an SP computes the probe partitioning, it also computes the values of QI_{comp} and QI_{comm} (line 2 in Algorithm 4). If $QI_{comp} > QI_{comm}$, the SP multiplies by two the number of groups and re-partitions the probes. If $QI_{comp} < QI_{comm}$ the SP divides by two the number of groups and re-partitions the probes (lines 5-8 in Algorithm 4). The SP continues to adjust the number of groups until $QI_{comp} + QI_{comm}$ has minimum value (lines 19-23 in Algorithm 4), meaning that the aggregation cost is near optimal. Thus, this process allows adapting the number of groups to the number and the spatial distribution of the probes.

6. SECURITY AND PRIVACY ANALYSIS

6.1 Security Analysis

The users cannot read the raw data of other users because the data stored in memory is protected by the secure MCU (i.e., the RAM is located inside the MCU) and the data stored in NAND Flash are cryptographically protected.

The SSI does not have the encryption key, so it cannot access the transiting data. In addition, the non-deterministic encryption protects the data against frequency-based attacks. The SSI may also try to buy an SP and pass for a user to gain access to the shared encryption key. However, this would be useless since the tamper-resistance of an SP protects the key. The SSI could collude with a querier, but it will gain access only to the aggregate result. Finally, since the samples are communicated through anonymizers, the SSI cannot identify the senders or link consecutive messages from the same user.

The SSI could try to infer information from the deterministically encrypted group ID values. Nevertheless, the SSI cannot perform a frequency-based attack using the en-

encrypted group ID, since all the groups contain approximately the same number of messages. Therefore, the SSI cannot infer the corresponding (approximate) location of a group or the topological neighborhood of the groups (which would be the first step to attack the users’ privacy). Hence, the only knowledge the SSI acquires is the number of groups and its evolution over time, which does not endanger the users’ privacy. Note that even if the SSI has somehow access to the full partitioning information and the corresponding encrypted ID, a user is still hidden under the corresponding group area and within the crowd in the same group (let us recall that the messages are sent anonymously so it is hard for the server to link the messages coming from the same user). Hence, the protocols are secure and fully protect the privacy of the users.

Although, protecting the privacy of users beyond the aggregate results is out of the scope of this paper (as discussed in Section 3.2), one can easily integrate basic protection mechanisms in PAMPAS for such cases. For example, to avoid the risk of exposure for the users situated in very sparse areas (e.g., a single user or very few users located in a spatial unit), we can simply add a predicate in the HAVING clause of the aggregate query (see Figure 3) indicating the minimum number of users in a spatial unit. In this way, the sparse spatial units are eliminated from the aggregate results. Another solution is to increase the size of the sliding window, or of the spatial units accordingly.

6.2 Privacy Analysis

To underline the high level of privacy protection of PAMPAS w.r.t. the SSI, we consider an entropy-based metric and apply it in the context of two scenarios that are related to our architecture. We then compare the privacy leakage in these two scenarios with the privacy leakage in PAMPAS.

Entropy is a popular metric to describe location privacy in general [6], and it is also appropriate to describe the privacy-preserving mechanism of PAMPAS. Commonly, entropy is used to quantify the average degree of uncertainty associated with a set of events. In the case of location privacy, the idea is to prevent user identification by obfuscating her exact location in a spatial region containing a certain number of individuals. Therefore, the level of privacy is directly related to the popularity (i.e., number of individual footprints) of the region. This means the higher the popularity, the higher the privacy level of the users in that region. Then, entropy is used to quantify the degree of popularity of a region. Formally, let reg be a spatial region and let $U(reg) = \{u_1, u_2, \dots, u_p\}$ be the set of users in region reg . Let f_i (with $1 \leq i \leq p$) be the number of sample updates (i.e., footprints) that user u_i sends from reg and $F = \sum_{i=1}^p f_i$ be the total number of sample updates sent from reg .

Definition 1. Entropy of a region: the entropy of region reg is defined as: $E(reg) = -\sum_{i=1}^p \frac{f_i}{F} \cdot \log \frac{f_i}{F}$

Definition 2. Popularity of a region: the popularity of region reg is defined as: $Pop(reg) = 2^{E(reg)}$

Definition 3. Privacy leakage: the privacy leakage for each $update_k$ sent by user u_i is defined as:

$$priv_leak_{u_i}(update_k) = \frac{1}{Pop(location\ of\ update_k)}$$

To compute the privacy leakage in different cases, we consider a simple numerical example inspired by the datasets used in our experimental evaluation (see Section 7.1). Let us consider that 200 thousand users participate in a mobile sensing application that aggregates data over 20 thousand spatial units (e.g., road segments in a road network). To keep the formulas tractable (but without loss of generality), let us consider that each user produces 50 samples from 50 distinct spatial units. This implies that on average, there are 500 footprints (i.e., updates) in each spatial unit.

Scenario 1: there is no grouping of the probes. Each participant sends the non-deterministically encrypted value of a sample together with the deterministically encrypted value of the spatial unit identifier to allow an efficient aggregation of the data. However, no fake sample is inserted by the probes. Although the spatial unit identifiers are encrypted, the SSI could easily determine the location of the spatial units if it has access to the global spatial distribution of the probes (i.e., a frequency-based attack). In this case, the average entropy of a spatial unit by applying Definition 1 is $E(s.unit) = -\sum_{i=1}^{500} \frac{1}{500} \cdot \log \frac{1}{500} = \log(500)$, which gives a popularity of $Pop(s.unit) = 500$ and an average privacy leakage of $priv_leak = 0.002$ for each update. Clearly, the privacy leakage can be lower or higher for each spatial unit depending on the popularity value compared with the average value.

Scenario 2: there is a static partitioning of probes, i.e., the spatial units are statically partitioned into a number of groups containing closely located spatial units. As in the previous case, the probes send the deterministically encrypted value of the spatial group and are also exposed to a frequency-based attack from the SSI. However, grouping many spatial units leads to decreasing the privacy leakage risk (but at the cost of increased aggregation time). For instance, partitioning the spatial units in 200 groups (i.e., 100 spatial units per group), leads to an average popularity $Pop(group) = 10^3$ and thus an average privacy leakage $priv_leak(update) = 10^{-3}$, which is smaller than in the previous scenario. Also, the obfuscation region is much larger since it corresponds to 100 spatial units instead of one.

PAMPAS goes even further in the protection of the participants’ privacy by using a dynamic partitioning of the probes based on their location and spatial distribution. The adaptive partitioning produces nearly balanced groups of probes. In addition, the eventual imbalance of the groups is corrected by injecting fake tuples, which precludes the SSI doing frequency-based attacks. This means that it is extremely difficult for the SSI to estimate even the approximate corresponding area of each group. Therefore, in our case, the entropy applies indistinguishably to all the participants leading to a popularity $Pop(group) = 2 \cdot 10^6$ and an average privacy leakage $priv_leak(update) = 5 \cdot 10^{-7}$. Practically, in PAMPAS, the privacy leakage depends only on the total number of participants, while the obfuscation area corresponds to the entire observation space. Besides, the number of groups is adaptively chosen such as to minimize the aggregation cost.

7. EXPERIMENTAL EVALUATION

The goals of our experimental evaluation are twofold: (i) compare the execution time and scalability of PAMPAS with those of a state-of-the-art protocol described in [19]; (ii) quantify the cost and scalability of our partitioning pro-

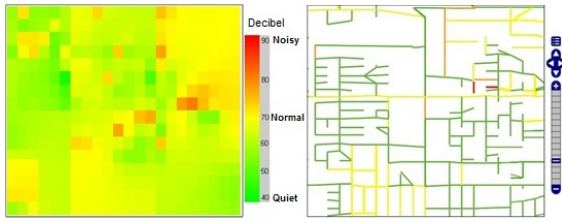


Figure 6: Aggregation maps for two applications: noise monitoring (left) and traffic monitoring (right)

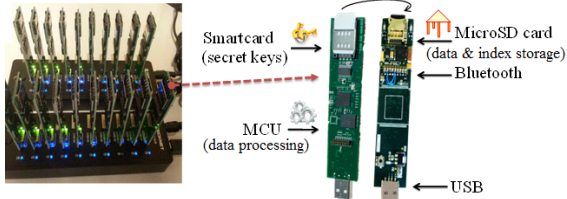


Figure 7: Secure tokens

toocol. We implemented and validated PAMPAS through emulations using secure tokens which have a hardware configuration representative for secure hardware platforms. As applications for our experiments, we used traffic monitoring and noise monitoring with both real and synthetic datasets representing small and medium-size cities. Figure 6 illustrates our graphic interface for these applications; it shows the aggregate results for the noise heat-map and the average travel time for the road network. A demo of our prototype was presented in [20] using a traffic monitoring scenario.

7.1 Experimental Setting

Hardware platform. In our experimental evaluation, the SSI is hosted on a PC (3.1 GHz i5-2400, 8GB RAM, running Windows 7) which also displays the aggregate results in a graphical form for validation purpose. The SPs are implemented by representative secure hardware devices (see Figure 7) which includes an MCU with a 32-bit RISC CPU at 120MHz, a cryptographic co-processor implementing AES and SHA, 128KB of static RAM and 1MB of NOR Flash to store the software stack, a smartcard chip hosting the cryptographic credentials (i.e., the secret encryption keys) and an SD card reader allowing for a large storage capacity. We used a commodity SD card (Samsung SDHC Essential Class 10 of 32GB) as secondary Flash storage. The SSI in our testing system manages a multi-channel Ethernet connection with a global bandwidth of 100Mbps. Importantly, on the SP’s side, our implementation limits the RAM consumption to only 30KB and the maximum communication bandwidth to 200Kbps to validate the proposed protocols with less powerful secure devices. Thus, in our experiments, all the SPs have this minimalist configuration. To emulate a very large number of SPs, we execute sequentially on an SP the aggregate computations and communications for all the worker SPs and measure the “parallel” execution time as the maximum aggregation time in the execution sequence.

Baseline system. To underline the importance of the PAMPAS protocols, we implemented the *secure protocol* proposed in [19] and took it as the baseline. This protocol can be applied without modification to aggregate the samples

collected in each time window. Since PAMPAS offers the same level of security and privacy as the baseline protocol, our experimental evaluation focuses on the efficiency part. Note that in [19], two more protocols are proposed that are even more expensive than the secure protocol if considered in our context.

Datasets and aggregate functions. We use both synthetic and real datasets to test the efficiency and scalability of PAMPAS. We employed the well-known Brinkhoff generator [4] to generate mobility traces on two real road networks of the cities of Oldenburg (Germany) and Stockton (San Joaquin County, CA). Oldenburg is a small size network having 7035 road segments, while Stockton is a medium size road network having 24123 segments. Depending on the network size, we generated traces corresponding to a medium and large number of users. With Oldenburg, the medium and large datasets contain 47 thousand and 270 thousand users respectively. With Stockton, the medium and large datasets contain 200 thousand and 1.35 million users respectively. The spatial distribution of the traces follows the network spatial density. Compared to the existing real datasets, the synthetic datasets have the prominent advantage of having excellent spatial and temporal coverage. However, it is also important to validate the proposed protocol with real datasets. To this end, we used the T-Drive Taxi trajectory dataset [21]. This dataset contains around 15 million trajectory units collected from 10357 taxis over a period from Feb. 2 to Feb. 8, 2008 in Beijing. Because the density of taxis is too low compared to the synthetic dataset, we extracted and merged a period of one hour in our tests, in order to generate a dataset containing 191 thousand trajectories covering 32800 road segments.

To show the generality of PAMPAS, we selected three aggregate functions, i.e., *average*, *IDW* [14] and *median*, corresponding to the three aggregate types described in Section 3.3. We associate the average and median aggregates with the traffic monitoring application, i.e., compute the average travel time and the median speed for each road segment in a road network. Hence, these two scenarios consider the constrained movement type. The IDW aggregate is associated to the noise-level monitoring application and a free movement type. In this case, we use the same generated mobility traces, but consider them in the 2D space instead of the network space. Also, we use a 64x64 grid to divide the observed 2D space into 4096 spatial units for the free movement scenario. The speed sample values are directly generated by the moving objects generator, while the noise values are generated by us proportionally to the number of probes in the spatial unit.

7.2 Performance Evaluation

Execution time. Figure 8 shows the aggregation time (in a logarithmic scale) for the three functions of both Baseline and PAMPAS protocols with 191 thousand probes in Beijing and with 200 thousand probes in Stockton. The aggregation time is global, i.e., it includes both the computation and communication time. The results indicate that PAMPAS is very efficient since it requires only a few seconds to compute the aggregate results for all the tested functions in both datasets. Also, the aggregation times of PAMPAS are similar between the real and the synthetic datasets. However, in both cases the baseline protocol is much more costly (especially for complex aggregate functions) leading

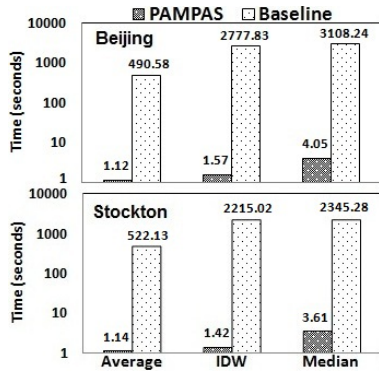


Figure 8: Aggregation time of PAMPAS and Baseline protocols in real dataset (top) and synthetic dataset (bottom)

to aggregation times up to three orders of magnitude higher than PAMPAS. Moreover, the aggregation times with the baseline protocol are larger for the Beijing dataset. The explanation is that the number of spatial units is larger in Beijing (i.e., 32800) than in Stockton (i.e., 24123). On the other hand, PAMPAS is scalable with respect to both the aggregation function and the number of spatial units in the query.

Scalability. We further test the scalability of the protocols with different number of probes, spatial units, and aggregate functions. Figure 9a shows the aggregation time for the two protocols for the average (top graph) and median (bottom graph) functions with medium and large number of users on both road networks. The results confirm that only PAMPAS is scalable w.r.t. all the varying input parameters. In the worst case, the computation time attains 14 seconds to compute the median speed for 1.35 million samples covering 24123 spatial units.

The baseline protocol does not scale with the number of samples and especially with the number of spatial units. Practically, the baseline can provide real-time aggregation only for a small number of spatial units (i.e., 7000 in Oldenburg) and basic aggregate functions (e.g., average). The very limited RAM of the SPs and the impossibility to efficiently parallelize the aggregate computation make the baseline inadequate for the requirements of participatory sensing applications.

Cost and scalability of partitioning protocol. Figure 9b (top) presents the partitioning computation time for both Oldenburg and Stockton networks. A new partitioning can be computed in a few seconds by an SP. This means that the checking and probes re-partitioning can be executed frequently, which allows PAMPAS to adapt to even fast changes in the spatial distribution of the probes. Most of the partitioning cost resides in reading and writing the partitioning data to the secondary Flash storage. This also explains the increase of the partitioning time with the number of partitions, since in this case the I/O operations are executed at a smaller granularity, which is more costly.

Figure 9b (bottom) indicates that the partitioning unbalance factor, i.e., the ratio between the maximum and the average partition size, increases with the number of partitions. The unbalance factor is an important indicator in PAMPAS since the higher the unbalance, the higher the number of fake injected samples and, therefore, the communication cost.

Figure 9c shows the impact of the number of partitions on the global aggregation time as well as on the computation and communication cost, which compose the total time. The computation time decreases with the increase of the number of partitions since the amount of work done by the aggregation SPs also decreases. Conversely, the communication time increases with more partitions since more fake samples are injected into the system as explained above. Globally, the near-optimal aggregation time is obtained with a number of partitions that minimizes the cumulated degradation of the computation and communication costs (see Section 5). We obtained similar results with the real dataset, for which the optimal number of partitions is 100 while the network partitioning is computed in just 2 seconds. The aggregation costs are partially shown in Figure 8 (top). Given the space limitation and the similarity of the results with the synthetic datasets, we omit here the details of the results with the real dataset.

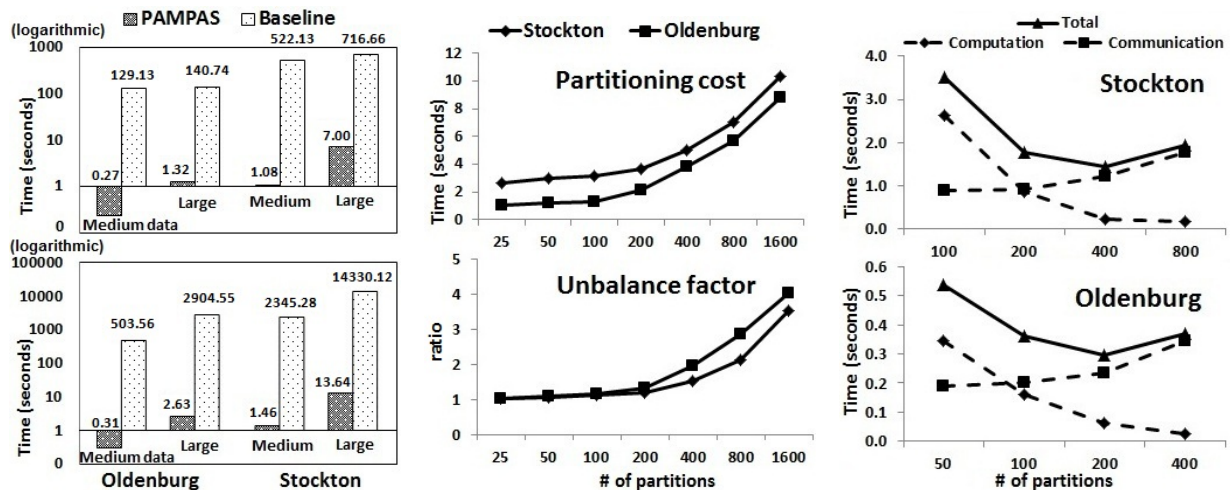
Discussion. It is worth mentioning that the aggregation time can be greatly improved by increasing the processing power and the communication bandwidth of the SSI. For example, increasing the server bandwidth from 100Mbps to 1 Gbps, makes the maximum aggregation time (i.e., median function with the large Stockton dataset) to drop from 14 seconds to less than 7 seconds. Also, in some scenarios, pushing the computation in the user devices may be problematic (e.g., battery powered devices, concurrent applications running in the device). However, PAMPAS minimizes this type of problem thanks to its design and its high efficiency. For instance, in our tests, a user participating in the system for one hour, has a probability between 3.5% and 8.7% to participate once to an aggregate computation assuming that aggregates results are produced every 30 seconds, and a probability between 0.004% and 0.12% to do a repartitioning assuming that the probes partitioning is checked every 1 minute. In all cases, the computation is done in a few seconds at most and requires only modest resources. Moreover, the computation effort is inversely proportional to the probability to be picked.

8. CONCLUSION

This paper proposes PAMPAS, a privacy-aware mobile participatory sensing system based on a distributed architecture and personal secure hardware. This combination allows PAMPAS to achieve the same level of privacy as cryptographic solutions without having to sacrifice generality, scalability, and accuracy. The proposed aggregation solution is, to the best of our knowledge, the first proposal of a distributed protocol that is secure, efficient, and scalable and that fits both the strict hardware constraints of secure personal devices and the real-time constraints of participatory sensing applications. The experimental evaluation based on representative hardware for secure platforms validates the proposed solution.

9. REFERENCES

- [1] T. Allard, B. Nguyen, and P. Pucheral. METAP: revisiting privacy-preserving data publishing using secure devices. *Distributed and Parallel Databases*, 32(2):191–244, 2014.
- [2] ARM. *ARM Security Technology - Building a Secure System using TrustZone Technology*. ARM Technical White Paper, 2009.



(a) Scalability of the PAMPAS and Baseline protocols with Average function (top) and Median function (bottom)

(b) The partitioning costs (top) and the partitioning imbalance factor (bottom) with different number of partitions

(c) Communication and computation costs of Median function with different number of partitions

Figure 9: Performance evaluations

- [3] A. Baumann, M. Peinado, and G. Hunt. Shielding applications from an untrusted cloud with haven. In *OSDI*, pages 267–283, 2014.
- [4] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, June 2002.
- [5] J. W. S. Brown, O. Ohrimenko, and R. Tamassia. Haze: privacy-preserving real-time traffic statistics. In *ACM SIGSPATIAL*, pages 540–543, 2013.
- [6] M. L. Damiani. Location privacy models in mobile applications: conceptual view and research directions. *GeoInformatica*, 18(4):819–842, 2014.
- [7] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3, 2013.
- [8] E. D’Hondta, M. Stevens, and A. Jacobs. Participatory noise mapping works! an evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. *Pervasive and Mobile Computing*, 9(5):681–694, October 2013.
- [9] G. Drosatos, P. S. Efraimidis, I. N. Athanasiadis, and M. Stevens. A privacy-preserving cloud computing system for creating participatory noise maps. In *COMPSAC*, pages 581–586, 2012.
- [10] M. Faezipour, M. Nourani, A. Saeed, and S. Addepalli. Progress and challenges in intelligent vehicle area networks. *Magazine Communications of the ACM*, 55(2):90–100, 2012.
- [11] H. Gao, C. H. Liu, W. Wang, J. Zhao, Z. Song, X. Su, J. Crowcroft, and K. K. Leung. A survey of incentive mechanisms for participatory sensing. *IEEE Comm. Surveys and Tutorials*, 17(2):918–943, 2015.
- [12] B. Hoh, T. Iwuchukwu, Q. Jacobson, D. Work, A. M. Bayen, R. Herring, J. C. Herrera, M. Gruteser, M. Annavaram, and J. Ban. Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines. *IEEE Tran. on Mobile Computing*, 11(5):849–864, 2012.
- [13] N. Jain, S. Mishra, A. Srinivasan, J. Gehrke, J. Widom, H. Balakrishnan, U. Çetintemel, M. Cherniack, R. Tibbetts, and S. B. Zdonik. Towards a streaming sql standard. In *PVLDB 1(2)*, pages 1379–1390, 2008.
- [14] S. Nittel, J. C. Whittier, and Q. Liang. Real-time spatial interpolation of continuous phenomena using mobile sensor data streams. In *ACM SIGSPATIAL*, pages 530–533, 2012.
- [15] M. Penza. Cost action TD1105: New sensing technologies for environmental sustainability in smart cities. In *IEEE SENSORS*, 2014.
- [16] R. A. Popa, A. J. Blumberg, H. Balakrishnan, and F. H. Li. Privacy and accountability for location-based aggregate statistics. In *CCS*, pages 653–666, 2011.
- [17] D. Quercia, I. Leontiadis, L. Mcnamara, C. Mascolo, and J. Crowcroft. Spotme if you can: Randomized responses for location obfuscation on mobile phones. In *ICDCS*, pages 363–372, 2011.
- [18] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *ACM SenSys*, pages 85–98, 2009.
- [19] Q.-C. To, B. Nguyen, and P. Pucheral. Privacy-preserving query execution using a decentralized architecture and tamper resistant hardware. In *EDBT*, pages 487–498, 2014.
- [20] D.-H. Ton-That, I. Sandu-Popa, and K. Zeitouni. PPTM: Privacy-aware participatory traffic monitoring using mobile secure probes. In *IEEE MDM*, 2015. Demo paper.
- [21] J. Yuan, Y. Zheng, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL*, pages 99–108, 2010.

Appendix H

DIVERT: A Distributed Vehicular Traffic Re-Routing System for Congestion Avoidance

DIVERT: A Distributed Vehicular Traffic Re-Routing System for Congestion Avoidance

Juan (Susan) Pan, Iulian Sandu Popa, and Cristian Borcea

Abstract—Centralized solutions for vehicular traffic re-routing to alleviate congestion suffer from two intrinsic problems: scalability, as the central server has to perform intensive computation and communication with the vehicles in real-time; and privacy, as the drivers have to share their location as well as the origins and destinations of their trips with the server. This article proposes DIVERT, a distributed vehicular re-routing system for congestion avoidance. DIVERT offloads a large part of the re-routing computation at the vehicles, and thus, the re-routing process becomes practical in real-time. To take collaborative re-routing decisions, the vehicles exchange messages over vehicular ad hoc networks. DIVERT is a hybrid system because it still uses a server and Internet communication to determine an accurate global view of the traffic. In addition, DIVERT balances the user privacy with the re-routing effectiveness. The simulation results demonstrate that, compared with a centralized system, the proposed hybrid system increases the user privacy by 92 percent on average. In terms of average travel time, DIVERT's performance is slightly less than that of the centralized system, but it still achieves substantial gains compared to the no re-routing case. In addition, DIVERT reduces the CPU and network load on the server by 99.99 and 95 percent, respectively.

Index Terms—Proactive driver guidance, vehicular congestion avoidance, distributed traffic re-routing, VANET

1 INTRODUCTION

THE problem addressed in this article is how to perform vehicular traffic re-routing for congestion avoidance in a scalable and privacy-preserving way. Previously, we proposed in [29] a centralized vehicular traffic re-routing system for congestion avoidance. The centralized system collects real-time traffic data from vehicles and potentially road-side sensors, and it implements several re-routing strategies to assign a new route to each re-routed vehicle based on actual travel time in the road network. Rather than using simple shortest path algorithms (e.g., Dijkstra), the re-routing strategies use load balancing heuristics to compute the new path for a given vehicle to mitigate the potential congestion and to lower the average travel time for all vehicles. This individualized path is pushed to a driver when signs of congestion are observed on his current path.

However, despite achieving a substantial decrease in the travel time experienced by drivers, centralized solutions such as ours suffer from two intrinsic problems. First, the central server has to perform intensive computation (to re-assign vehicles to new paths) and communication with the vehicles (to send the paths and to receive location updates) in real-time. This can make centralized solutions infeasible for large regions with many vehicles. Second, in a centralized architecture, the server requires the real-time locations as well as the

origins and destinations of the vehicles to estimate the traffic conditions and provide effective individual re-routing guidance. This leads to major privacy concerns for the drivers and may prevent the adoption of such solutions due to “big brother” fears. As long as vehicles’ traces are fully disclosed, user’s identity can easily be inferred even if pseudonyms are used [16]. This is due to the fact that location can contain identity information [31]. Moreover, a sequence of location samples will eventually reveal the vehicle’s identity [40]. Therefore, it is important to make the system work without disclosing the users’ origin-destination (OD) pairs and with the least number of location updates along a user trip.

These requirements suggest a distributed system architecture. However, a fully decentralized architecture is not suitable for a proactive re-routing system. For example, by creating vehicular ad hoc networks (VANETs), the vehicles can exchange information using multi-hop communication, and thus can detect signs of congestion in small regions while preserving their privacy. However, VANETs do not permit vehicles to get an accurate global traffic view of the road network, resulting in wrong or at least sub-optimal re-routing decisions. In addition, in a fully distributed architecture, due to the lack of a coordinator, the vehicles cannot take synchronized actions at the same time, which makes it infeasible to make collaborative decisions in real-time.

To tackle all these problems, this article proposes DIVERT, a distributed vehicular re-routing system for congestion avoidance, which leverages both cellular Internet and VANET communication. DIVERT is a hybrid system because it still uses a server, reachable over the Internet, to determine an accurate global view of the traffic. The centralized server acts as a coordinator that collects location reports, detects traffic congestion and distributes re-routing notifications (i.e., updated travel times in the road network) to the vehicles. However, the system offloads a large part of

- J. Pan and C. Borcea are with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102-1982. E-mail: {jp238, borcea}@njit.edu.
- I. Sandu Popa is with the DAVID Laboratory, University of Versailles Saint-Quentin-en-Yvelines, Versailles 78000, France, and Inria Saclay-Ile-de-France, Palaiseau 91120, France. E-mail: iulian.sandu-popu@uvsq.fr.

Manuscript received 9 Mar. 2015; revised 1 Jan. 2016; accepted 22 Feb. 2016.
Date of publication 3 Mar. 2016; date of current version 1 Dec. 2016.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2016.2538226

the re-routing computation at the vehicles and thus the re-routing process becomes practical in real-time. To take collaborative re-routing decisions, the vehicles situated in the same region exchange messages over VANETs. Also, DIVERT implements a privacy enhancement protocol to protect the users' privacy, where each vehicle detects the road density locally using VANET and anonymously reports data with a certain probability only from high traffic density roads. When signs of congestion are detected, the server sends the traffic map only to the vehicles that sent the latest updates. Subsequently, these vehicles disseminate the traffic data received from the server in their region. User privacy is greatly improved since this protocol reduces dramatically the number of vehicle location updates to the server and, thus, the driver exposure and identification risks. Moreover, in this hybrid architecture, the server does not know the OD pairs of the users.

Hence, the main contribution of this article is the distributed system for re-routing. This system, DIVERT, has four main features: (1) a scalable system architecture for distributed re-routing, (2) distributed re-routing algorithms that use VANETs to cooperatively compute an individual alternative path for each vehicle that takes into account the surrounding vehicles' future paths. (3) privacy-aware re-routing that significantly decreases sensitive location data exposure of the vehicles, and (4) optimizations to reduce the VANET overhead and thus improve vehicle-to-vehicle communication latency.

We measured the effectiveness and efficiency of DIVERT through extensive simulations over two real medium-size urban road networks. The experimental results show that, in comparison with the centralized system, DIVERT can decrease the privacy exposure by 92 percent in addition to not revealing the OD pairs of the user trips. In terms of average travel time, DIVERT's performance is slightly less than that of the centralized system, but it still achieves substantial gains compared to the no re-routing case. DIVERT is more scalable since it offloads most of the computation burden to the vehicles and reduces the network load on the server by 95 percent.

The rest of this article is organized as follows. Section 2 summarizes the related work. Section 3 explains the design principles of DIVERT. Section 4 introduces the privacy enhancement mechanism and the privacy metrics. Section 5 introduces the two distributed re-routing strategies. The four VANET optimization techniques are presented in Sections 6. The results and associated analysis are discussed in Section 7. We conclude in Section 8.

2 RELATED WORK

In this section we present works relevant to DIVERT. We discuss the aspects related to traffic re-routing, traffic information sharing in vehicular networks, and preserving privacy in location-based services.

2.1 Traffic Guidance and Re-routing Systems

Services such as INRIX [1] provide real-time traffic information at a certain temporal accuracy, which allows drivers to choose alternative routes with lower travel times. Google Maps and Microsoft's Bing are able to forecast congestion

and its duration by performing advanced statistical predictive analysis of traffic patterns. Various mobile navigation applications [2], [3], [4], [5] use such traffic information to transform smart phones into navigation devices. However, these services share the same problem: when congestion happens, they provide the same path for the affected vehicles which potentially generates another local congestion.

The above problem can be solved by dynamic traffic assignment (DTA) which assigns each driver either system-optimal or user-optimal routes [10]. However, DTA algorithms may not be able to compute the equilibrium fast enough to inform the vehicles about their new routes in time to avoid congestion. DIVERT, on the other hand, is designed to be effective and fast, although not optimal, in deciding which vehicles should be re-routed when signs of congestion occur as well as computing alternative routes for these vehicles.

Several other projects have also aimed to provide near-optimal routes to drivers but better scalability compared to DTA. In [24], the first k shortest paths from source to destination are calculated, and then the system determines which path each vehicle should take by minimizing a Lyapunov-style cost function. In [7], the authors proposed a genetic algorithm method to compute the alternative paths and assign them to cars under the assumption that the traffic is known a priori. This method computes the assignment only once as opposed to traditional assignments methods, which iteratively re-compute the assignments in order to approach the optimum solution (i.e., user or system equilibrium). Our previous work [29] and the research in [39] emphasize that the previous route planning decisions should be considered when determining the next route. Themis [25] uses a similar approach, but it computes the re-routing alternatives based on real-time speed, predicted travel time, and anticipated traffic volume.

DIVERT differs from the above research in three aspects. First, we take full advantage of both cellular and VANET communication to perform scalable re-routing. Thus, each vehicle can get accurate global knowledge of the travel time and, at the same time, is able to exchange route planning decisions with surrounding vehicles more efficiently. Second, the route computation is performed in a distributed way over VANETs. Therefore, it is more scalable since it reduces the computation burden of the central server. Third, we designed and evaluated a privacy enhancement mechanism, where each vehicle only uploads its location report when located in low sensitivity areas.

The work in [23] proposes two urban traffic prediction models that can be used in conjunction with DIVERT. These models could improve the accuracy of our congestion estimation, especially when DIVERT tends to overestimate congestion.

2.2 Traffic Information Sharing in Vehicular Networks

VANETs enable traffic information sharing for intelligent transportation systems. To improve dissemination efficiency, Gao et al. [12] proposed an adaptive query evaluation plan by taking into account the road topology. Also, Loulloudes et al. presented in [26] V-Radar, an efficient protocol for traffic information retrieval using V2V communications. Several

works have sought inspiration in Biology and Internet protocols communication [32], [35]. However, since they employ VANETs, the above approaches have only a partial view of the traffic conditions, which may lead to less accurate re-routing. Also, simply treating vehicles as packets which always listen to the guidance ignores the nature of human behavior. Furthermore, these systems react to real-time data without insight into future conditions, thus introducing greater vulnerability to switching congestion from one spot to another.

2.3 Privacy Preserving in Location Based Services

A large body of works consider the problem of preserving the user's privacy in the context of location based services (LBSs). For instance, the middle layer of DSRC defines the security services for application and message management [22]. Authentication schemes are designed to preserve the driver privacy in DSRC-based VANETs [33]. To prevent malicious tracking, a vehicle could change its anonymous key within an interval of a few minutes [38]. DIVERT has a different goal from all these works: it focuses on protecting the driver's location privacy from the central server, not from the other drivers in VANET. For driver-to-driver privacy, DIVERT can leverage the existing solutions.

SCMS [37] provides privacy protection from both outsiders (i.e., other vehicles or eavesdroppers) and insiders (i.e., administrators of the servers). DIVERT is complementary to SCMS, as its goal is to minimize the amount of privacy sensitive information uploaded to the server (i.e., location and OD pairs), not to protect the information privacy once it has been uploaded. Furthermore, SCMS relies on the organizational separation assumption to protect against insider attacks (i.e., different server component are distributed between different organizations which do not collude with each other). DIVERT, on the other hand, achieves a good level of location privacy protection even if this assumption does not hold.

Many works focus on spatial cloaking [15], [40] to provide k -anonymity. The work in [14] argues that both spatial and temporal dimensions should be considered in the algorithm to achieve better k -anonymity. Fundamentally, k -anonymity reduces the quality of the user's localization, which is not applicable for continuous location based services such as real-time vehicle re-routing.

A number of mechanisms provide solutions for highly accurate real-time location updates, while achieving good privacy protection [19], [27]. However, these mechanisms require a trusted centralized entity such as a proxy server for location reporting. Our privacy aware mechanism works in a distributed and probabilistic fashion without any help from trusted entities. Thus, the risk of location tracking is distributed over VANETs, and we argue that this is qualitatively better than trusting a single central entity.

3 SYSTEM OVERVIEW

This section presents an overview of DIVERT, describing design principles and its system architecture.

3.1 Design Principles

DIVERT is built around two design principles corresponding to the two major requirements described in Section 1. First,

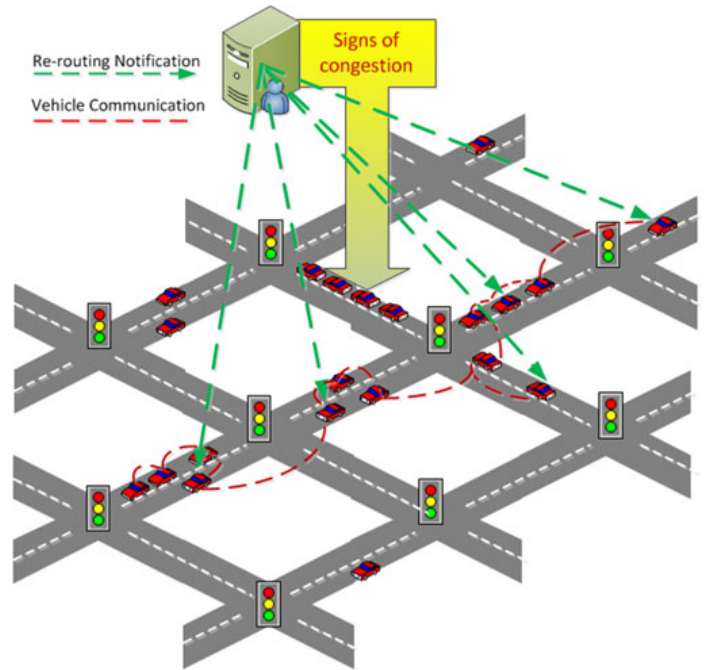


Fig. 1. DIVERT's hybrid architecture.

the re-routing path computation should be offloaded from the central server to the vehicles to reduce the computation and communication burden on the server and achieve better scalability. Therefore, the alternative routes should be computed by vehicles when there are signs of congestion on the roads. At the same time, the re-routing computation should be collaborative in order to achieve a better re-routing effectiveness. To this end, the vehicles could exchange messages over VANETs and implement a distributed re-routing process.

Second, DIVERT should be designed to respect the privacy of the users from its conception, i.e., a privacy-by-design system, which can be essential for the wide acceptance of the system. Implicitly, by offloading the path computation to the vehicles, the drivers' exposure is reduced significantly since very sensitive location information (i.e., the OD pairs) is not sent to the server anymore. Nevertheless, protecting only the OD of a vehicle is not sufficient. DIVERT needs a mechanism to protect the identity of vehicles while reporting location data.

3.2 System Architecture

Given the described design principles, a hybrid architecture is proposed to implement DIVERT as shown in Fig. 1. The architecture is composed of a central server and a software stack running on an on-board device (e.g., a smart phone) in each participating vehicle. DIVERT uses two types of communication. The vehicles communicate with the server over a cellular network to report local traffic density data and to receive the global traffic density in the road network (i.e., the green arrows in Fig. 1). The vehicles report data according to a privacy-aware algorithm that is detailed in Section 4. Also, the vehicles that are closely located communicate with each other over VANETs to determine the local traffic density, to disseminate the traffic data received from the server, and to implement a distributed re-routing strategy (i.e., the red lines in Fig. 1) as detailed in Section 5.

The server uses the vehicle traffic reports to build an accurate and global view of the road network traffic. The network

is represented as a directed graph where each edge corresponds to a road segment. In addition, each edge has associated a dynamic weight representing the real-time traffic density on the edge. A road segment is considered to exhibit signs of congestion when the traffic density is greater than a threshold value. Each time new road segments exhibit congestion signs, the server sends a partial weighted graph (i.e., only the edges having a travel time different from the free flow travel time) to the cars that reported recently and are close to the congestion segments (see Section 4).

The notified vehicles disseminate the information (i.e., traffic graph and vehicle route) in their regions with a limited number of hops to avoid excessive flooding. The dissemination also has a timeout, which is a constant parameter in the proposed system. When the time is up, based on the traffic graph and route information shared by other vehicles, each vehicle, whose current path traverses the congestion spot, locally computes a new route to its destination. This re-routing process is presented in Section 5.

4 PRIVACY-AWARE TRAFFIC REPORTING

DIVERT's goal is to protect driver's location privacy against attackers at the server side because the server could link traffic reports (which include locations) to driver identities. The traffic reports need to be frequent to compute a global traffic view and detect congestion accurately. Yet, the location reports, when sent frequently, can create severe privacy leakage [16], [31]. Even if pseudonyms are used for location reports and are changed frequently, attackers at the server side can use background information to discover the user identity for certain location points (home, work, etc.) and then use prediction algorithms to identify the whole location trace [16]. Therefore, DIVERT strives to minimize the driver's privacy leakage by reducing the amount of location reports uploaded at the server, while maintaining good traffic accuracy.

To this end, we introduce first a privacy metric in Section 4.1. Then, we propose in Section 4.2 a privacy-aware traffic reporting mechanism based on the road traffic density to reduce the privacy leakage for the reporting vehicles. Our system considers that the vehicles are trusted. Indeed, to avoid congestion, vehicles make collaborative decisions which require a vehicle to share shortest path or OD pair information with other nearby vehicles. While privacy enforcement at the vehicle side is out of the scope of this paper, it is worth mentioning that several recent works consider the problem of shared data protection by leveraging the Trusted Execution Environment of (personal) secure devices. Be it the secure TrustZone CPU [41] of the ARM cortex-A series equipping most of mobile devices today, a tamper-resistant hardware security module securing the on-board computer of a vehicle [42], a secure portable token [43] communicating with the user smart phone or plugged inside it (e.g., Google Vault), all such secure devices offer tangible, hardware-based security guarantees. Such technologies can be leveraged for secure data processing in a distributed architecture as DIVERT to protect the shared information and prevent malicious vehicles obtaining access to unauthorized data [44]. Instead of hardware-based security, other techniques such as secure multiparty

computing or protocols based on homomorphic encryption could also be considered, but they may impact the real-time constraints of DIVERT. Finally, in Section 4.3, we present the algorithm used by the central server to compute the travel time in the road network.

4.1 Privacy Metric

In order to measure the privacy loss due to each location update, each location report is associated with a weight. Similar to [40], the weight of a location report depends on the popularity of the location road segment. That is, the more popular a spatial region is, the more difficult it is for an adversary to single out the report sender. However, the number of vehicles along the segment is not sufficient to quantify its popularity, because some vehicles may have a dominant presence in that space. Instead, a metric is applied that is based on the entropy of the road segment.

Definition 1 (Road segment popularity). Let rs be a road segment and $S(rs) = v_1, v_2, \dots, v_m$ be the set of vehicles that send location updates in rs . Let n_i ($1 \leq i \leq m$) be the number of location updates that vehicle v_i sent from rs and $N = \sum_{i=1}^m n_i$. Then the entropy of the road segment rs is $E(rs) = -\sum_{i=1}^m \frac{n_i}{N} \log \frac{n_i}{N}$ and the popularity of rs is $P(rs) = 2^{E(rs)}$.

Definition 2 (Privacy leakage). Given the above defined popularity measure of a location, the global privacy leakage for vehicle v_i is defined as:

$$pleak_{v_i} = \sum_{\text{all updates}} \frac{\text{update}_i}{P(\text{location of update}_i)}. \quad (1)$$

Definition 3 (Average privacy leakage.). The average privacy leakage for all the vehicles is:

$$pleak_{avg} = \frac{\sum_{\text{all vehicles}} pleak_{v_i}}{\text{total number of vehicles}}. \quad (2)$$

4.2 Density Reporting

Our privacy-aware reporting is based on the observation that in dense areas, vehicles naturally experience a higher degree of anonymity similar to a person walking through an inner-city crowd. Therefore, a density-based traffic reporting mechanism is proposed wherein vehicles report to the server only if the road density is higher than a predefined threshold. The server computes the smoothed average of the traffic density on each road segment as it receives new traffic reports. Computing the smoothed average of the traffic density at each vehicle (using a moving time window) is of little use in our case because the vehicles do not report often due to our privacy-aware reporting protocol (e.g., a vehicle rarely reports twice from the same road segment). This mechanism is beneficial for both the re-routing effectiveness and the vehicle privacy, since the server can still accurately detect the congestion signs at the cost of lower user privacy exposure.

Our goal is to minimize the number of vehicle reports, i.e., only a fraction of the vehicles situated on a road segment will send traffic reports. Specifically, density reports

sent to the server conform to the following rules: (1) cars submit reports only when they perceive that the density on the road segments is above a threshold that would signal a chance of congestion, (2) cars decide probabilistically when to submit data as function of the density—i.e., the more cars there are, the fewer reports each car submits as the reports are distributed among the cars on the segment, (3) cars send their messages through anonymizers (e.g., Tor [11]) to protect their identities.

The estimated density is computed locally by each vehicle, which obtains information about its neighbor vehicles by periodic exchange of beacons. Each vehicle counts the received beacons in a short time window (i.e., 5 seconds in our implementation) and each vehicle emits beacons with the same frequency (such that each vehicle is counted exactly once in each period). As depicted in Algorithm 1, each vehicle periodically checks the number of vehicles N_i on the current road r_i . To obtain accurate traffic information, each vehicle encapsulates in the beacon the current road identifier (i.e., r_i) and direction of traffic (i.e., $side$). When a vehicle estimates the number of cars (line 2), it only counts the beacons with the same r_i and $side$ as itself.

Algorithm 1. Privacy-Aware Density Reporting

```

1: procedure onDenistyCheckTimeout()
2:  $N_i = \text{getEstimatedNumberOfCars}(r_i, \text{side}, \{\text{beacon}_k\})$ 
3: if  $N_i \geq \max[\theta * N_{max}, \text{bound}]$  then
4:    $p = 1/N_i$ 
5:   if  $\text{rand} < p$  then
6:      $\text{sendtoServer}(N_i, r_i, \text{side})$ 
7:   end if
8: end if

```

The vehicle reports the detected density to the central server with probability $p = 1/N_i$ only if $N_i \geq \max[\theta * N_{max}, \text{bound}]$ (lines 3-4). N_{max} is the maximum number of vehicles that can occupy road r_i , and θ is a system parameter threshold. $N_{max} = L_i * \text{Lane}_i / \text{veh_len}$, where L_i and Lane_i are the length and the number of lanes of road r_i . The parameter veh_len is the sum of the average vehicle length ($\approx 5\text{meters}$) and the minimal gap between vehicles ($\approx 2.5\text{meters}$). The parameter bound specifies that, when reporting, there must be at least bound cars on the road. Practically, privacy is protected by N_{max} for longer roads with multiple lanes and by bound for short roads with potentially few lanes (i.e., bound gives a minimal privacy guarantee). Additionally, after a car reports, it stops reporting for a time period (e.g., 5 seconds in our prototype) to prevent frequent reporting that could lead to privacy leakage.

If every vehicle applies the same reporting procedure, then the probability for the server to receive x location reports from r_i is $\binom{N_i}{x} (1-p)^{N_i-x} p^x$. The expected number of updates on road r_i is $nu_i = \sum_{x=0}^{N_i} x \binom{N_i}{x} (1-p)^{N_i-x} p^x$.

A potential problem of the above algorithm is that it does not consider the case in which the vehicle transmission range is shorter than the road segment length. First of all, this is a very rare case since the typical transmission range is 500 meters, which can cover most urban road segments (e.g., the average road segment length is less than 300 meters for the two real road networks used in our

experiments). Second, if indeed a segment is longer than the transmission range, the reported density is inaccurate especially when the traffic density along the segment is extremely skewed (e.g., congestion starts to form at one end of the segment, while the other end still has light traffic). In this case, our algorithm overestimates the density and may trigger re-routing. However, this is not necessarily an issue because if nothing is done, the congestion is expected to increase anyway when the incoming traffic on the segment is heavy.

4.3 Report Collection and Travel Time Computation

The server receives reports from vehicles indicating the number of vehicles on a road segment and computes the traffic density on the roads. Every time the server receives a report concerning a road r_i , it will smooth the computed density value K_i using the following formula: $K_i^{\text{current}} = \alpha * K_i^{\text{old}} + (1 - \alpha) * K_i^{\text{new}}$. The value of α is experimentally chosen to be 0.05. The server then estimates the travel time of each road segment by considering the following three cases.

Case 1: For the road segments without any traffic reports, the server estimates the travel time to be the free flow travel time. Note that this travel time is an approximate value for some roads, as vehicles only report when the vehicle density is above the threshold density.

Case 2: For the road segments with a non-zero traffic density but for which the last report time is older than a predefined time interval, the server does an expiration operation. Specifically, if the difference between the last update time and the current time is greater than τ times the free flow travel time of the road, then the road density is reset to zero. The value for τ is also based on empirical results and is chosen to be equal to four in this system.

Case 3: For the road segments that do not fit in Cases 1 and 2, the server uses the Greenshield model [6] to estimate the travel time according to the speed-density-volume relation. This model is used extensively by transportation researchers and was shown empirically to describe well the speed-density relation for relatively low densities. The model considers a linear relationship between the estimated road speed V_i and the traffic density K_i (vehicles per meter) on road segment r_i :

$$V_i = V_f \left(1 - \frac{K_i}{K_{jam}} \right) \quad T_i = L_i / V_i, \quad (3)$$

where K_{jam} and V_f are the traffic jam density and the free flow speed for r_i , while T_i and L_i are the estimated travel time and length for the same segment. The free flow speed V_f is defined as the average speed at which a motorist would travel if there were no congestion or other adverse conditions. To simplify our implementation, we consider that the free flow speed is the legal speed limit, and the traffic jam density is when the road is fully occupied by cars. In this case, K_i / K_{jam} equals N_i / N_{max} , where N_{max} is the maximum number of vehicles on the road segment and N_i is the current number of vehicles obtained from the traffic data collected by the system.

Note that, because of the density-based reporting policy, the traffic density may not be fully accurate.

However, the higher the traffic density of a road is, the more accurately the traffic density will be estimated. This is important since the re-routing effectiveness mainly depends on the traffic accuracy of the dense traffic roads. In Section 7, we show that the loss of accuracy in the traffic view has only a marginal effect on the re-routing effectiveness, but greatly improves the privacy protection of the users.

5 DISTRIBUTED RE-ROUTING STRATEGIES

If the server detects signs of congestion in the road network, it will alert the vehicles by sending the updated map information, i.e., the “updatedMap” parameter in Algorithm 2 containing tuples (*road id*, *new computed travel time*) for all the roads that have a current travel time different from the free flow travel time. The server sends messages only to the vehicles that reported most recently and that are located near a congestion spot, i.e., no further than three road segments from the congested segment. The server notification triggers the re-routing process that consists of a dissemination phase and a route computation phase. In addition, the dissemination phase has two sub-phases as presented in Algorithm 2. When a vehicle receives such a notification message either directly from the server or from the surrounding vehicles, it executes the procedure described in Algorithm 2. The first part of the procedure (lines 2–4 in Algorithm 2) consists in disseminating to other vehicles the updated travel times in the network. In the second part of the dissemination phase, the vehicles that received the notification broadcast personal route information to the other vehicles. The route information depends on the re-routing strategy employed by DIVERT, i.e., either the k-shortest paths or the OD pair of the vehicle (lines 6–10 and 11–15 in Algorithm 2).

Algorithm 2. When Vehicle Receives a Congestion Notification

```

1: procedure onCongestionNotification(updatedMap)
2: updateTravelTime(updatedMap) {update the travel time of
  the map}
3: T ← computeBroadcastTime(this.rank) {compute when to
  start the broadcast based on this vehicle’s rank}
4: broadcastUpdatedMap(T) {broadcast the updated travel
  time map}
5: if this.currentPath intersects congestionSpots then
6:   if dEBkSP then
7:     computekShortestPaths(this, k) {compute the k shortest
      path for itself}
8:     wait( $T_{mapBroadcast}$ ) {wait until the map broadcast phase
      finishes}
9:     broadcastkShortestPaths(T) {broadcast the k shortest
      paths at time T}
10:  end if
11:  if dAR* then
12:    getODPair(this) {get the OD pair for itself}
13:    wait( $T_{mapBroadcast}$ ) {wait until the map broadcast phase
      finishes}
14:    broadcastODPair(T) {broadcast OD pair at time T}
15:  end if
16: end if

```

Algorithm 3. When Vehicle Receives the Broadcast

```

1: procedure onReceived(vehiclemsg)
2: v = unpack(vehiclemsg) {unpack the message and extract
  the vehicle data, e.g., rank, k paths or OD pair}
3: receiveddata.push(v) {put the vehicle data into the priority
  queue based on the rank}

```

On receiving a route information message, the vehicles store the received data as indicated in Algorithm 3. The received data will be used in the route computation phase to compute a new best path for the current vehicle. Note that all the notified vehicles participate in the updated map data dissemination, but only the vehicles whose current paths traverse a congestion spot execute the computation phase (line 5 in Algorithm 2). We only re-route vehicles that are directly impacted by congestion since this is sufficient to alleviate congestion and improve the travel times for all vehicles. Moreover, this approach reduces the re-routing frequency for a driver and thus the computation and communication overhead [29]. In the remainder of this section, we present an overview of our two centralized re-routing strategies that have proven to be the most effective in alleviating congestion, and then describe their distributed counterparts which allow the vehicles to compute alternative paths in a collaborative manner when congestion happens.

5.1 Overview of Centralized Re-Routing

The first centralized re-routing strategy is the Entropy Based k Shortest Paths (EBkSP). The server first ranks the vehicles to be re-routed as a function of their remaining travel time to destinations (i.e., shortest time first). Then, it computes k alternative shortest paths for each vehicle. The server sequentially goes through the ranked list and assigns to each vehicle the best path out of the k computed paths. This is the least popular path among the k alternatives in order to avoid potential future congestion. To compute the least popular path, a weighted footprint counter as defined below is attached to each road segment.

Definition - Weighted footprint counter. A *weighted footprint counter*, fc_i , of a road segment r_i is defined as follows: $fc_i = n_i \times \omega_i$, where n_i is the total number of vehicles that are assigned to paths that include segment r_i , and ω_i is a weight associated with r_i . $\omega_i = \frac{len_{avg}}{len_i \times lane_i} \times \frac{Vf_{avg}}{Vf_i}$, where len_{avg} is the average road segment length in the network, Vf_{avg} is the average free flow speed of the network, len_i is the length of r_i , Vf_i is the free flow speed of r_i , and $lane_i$ is number of lanes of r_i .

Specifically, the first vehicle is assigned the current best path without considering others. Then, the footprints counters are updated based on the new path. When assigning the second vehicle, the popularity scores of its k-shortest paths are calculated as defined below, and the least popular path will be chosen. The process is then repeated for the rest of the re-routed vehicles.

Definition - Popularity of a path. Let (p_1, \dots, p_k) be the set of paths computed for the vehicle which will be assigned next. Let (r_1, \dots, r_n) be the union of all segments of (p_1, \dots, p_k) , and let (fc_1, \dots, fc_n) be the set of weighted footprint counters associated with these segments. The popularity of p_j is defined as

$Pop(p_j) = e^{E(p_j)}$. $E(p_j)$ is the weighted entropy of p_j and is computed as $E(p_j) = -\sum_{r_i \in p_j} \frac{f_{c_i}}{N} \ln \frac{f_{c_i}}{N}$, $N = \sum_{r_i \in p_j} f_{c_i}$.

The second centralized strategy is the A* with Repulsion (AR*) algorithm. AR* modifies the classical A* algorithm to incorporate the other vehicle paths into the path computation of the current vehicle as a repulsive force. As with EBkSP, the server ranks the vehicles to be re-routed and computes the alternative path for each vehicle in this order. The classical A* [18] uses a best-first search and a heuristic function to determine in which order to visit the network nodes (crossroads in our case). Given a node x , a heuristic function $F(x)$ is computed as the sum $G(x) + H(x)$. $G(x)$ is the path-cost from the start node to x , which corresponds to the travel time in our case, while $H(x)$ is a heuristic estimation of the remaining travel time from x to the destination node. Typically, $H(x)$ is computed as the Euclidean distance divided by the maximum speed on the roads. In AR*, we modified the heuristic function $F(x)$ to include the other vehicles sharing the same path as a repulsive force. Specifically, we define the repulsive score $R(x)$ of a node x as the sum of the weighted footprint counters from the starting node to node x . Thus, the path-cost function becomes $F(x) = (1 - \beta) \times (G(x) + H(x)) + \beta \times R(x)$, where $G(x)$ and $H(x)$ are computed as in the original algorithm and β is a weighting parameter. $G(x) + H(x)$ measures the travel time factor, while $R(x)$ reflects the impact of other vehicle traces on the examined path. Since the travel time and the repulsive force use different metrics, we normalize their values and compute $F(x)$ as a linear combination of the two factors. The parameter β allows a variable weighting between the travel time and the repulsive force; its value is determined empirically to achieve the best AR* effectiveness.

5.2 Distributed Re-Routing

EBkSP and AR* greatly improve the vehicles' travel time. However, these strategies are designed for a centralized architecture in which all the re-routing computation is done at the server side. Our objective is twofold. First, we want to provide re-routing strategies that are based on the same ideas as the effective centralized strategies, but that can run in DIVERT. This is challenging, since the whole computation can only be done by the vehicles in order to comply with the system design principles (see Section 3.1). Second, the distributed re-routing should ideally have similar effectiveness as the centralized re-routing. In the following, we present dEBkSP (distributed EBkSP) and dAR* (distributed AR*), two distributed re-routing strategies that achieve these objectives.

Both dEBkSP and dAR* require the re-routed vehicles to be ranked. In the centralized version, the rank of each vehicle is assigned by the server. Here, each vehicle picks a random rank value between 0 and $rank_{max}$, which is the estimated total number of re-routed vehicles; $rank_{max}$ is calculated by each re-routed vehicle based on the traffic density of the incoming road segments no further than L segments from the congestion point (e.g., L is 3 in our experiments). A vehicle of a certain rank computes a new route by considering the higher ranked vehicle paths. In

dEBkSP, each vehicle affected by congestion calculates the k loop-less shortest paths based on its current OD pair and the updated travel times on the roads. Then, the vehicles disseminate their rank and k shortest paths in their region for a predefined time interval (see Section 6). At the end of the route dissemination phase, each vehicle receives the k shortest paths of the vehicles in the region. However, given the nature of the dissemination process, the information gathered by a vehicle can be incomplete and different from one vehicle to another. In the final route computation phase, each vehicle iterates through the local sorted list of vehicles for which it has received information. It selects the (potentially) best path based on the original EBkSP algorithm for each vehicle with a higher rank and eventually selects the best path for itself.

Similarly, in the case of dAR*, the notified vehicle chooses a random rank but it does not compute the k shortest paths. Instead, the notified vehicles only broadcast their OD pairs. In the event of a broadcast timeout, for each received OD pair in the buffer, the vehicle applies the original AR* algorithm to compute a virtual path. The current vehicle assumes that the vehicle with that OD pair will take that virtual path. By the end of the process, the current vehicle computes the best shortest path for itself based on other vehicles' paths.

Algorithm 4. Distributed EBkSP and AR*

```

1: procedure onBroadcastTimeOut()
2: receiveddata {all the received data}
3: Q = empty {a queue that stores the vehicle objects that have
   already been processed}
4: while  $v_i \neq$  this do
5:    $v_i =$  receiveddata.pop()
6:   if dEBkSP then
7:     getkShortestPath( $v_i$ ) {get the  $k$  shortest path for this
       vehicle}
8:     doEBkSP( $v_i$ , Q) {pick a path from the  $k$  paths for vehi-
       cle  $v_i$  based on vehicles' paths with higher rank}
9:     Q.push( $v_i$ ) {label the vehicle  $v_i$  as a processed vehicle}
10:  end if
11:  if dAR* then
12:    getODpair( $v_i$ ) {get the OD pair for this vehicle}
13:    doAR( $v_i$ , Q) {Compute a A star shortest path with
       repulsion for this vehicle based on vehicles' paths with
       higher rank }
14:    Q.push( $v_i$ ) {label the vehicle  $v_i$  as a processed vehicle}
15:  end if
16: end while
17: if dEBkSP then
18:   getkShortestPath(this) {get the  $k$  shortest path for itself}
19:   doEBkSP(this, Q) {pick a path from the  $k$  paths for itself
       based on vehicles' paths with higher rank}
20: end if
21: if dAR* then
22:   getODpair(this) {get the OD pair for itself}
23:   doAR(this, Q) {Compute a A star shortest path with repul-
       sion for itself based on vehicles' paths with higher rank }
24: end if

```

Algorithm 4 describes how dEBkSP and dAR* are executed when the information dissemination phase ends. dEBkSP and dAR* have opposite behaviors with regards to the two main phases of the re-routing process. dEBkSP

incurs a higher overhead in the communication phase than dAR*. The packets in dEBkSP are significantly larger than those in dAR*: dEBkSP packets contain the k shortest paths, while dAR* packets contain only the vehicle OD pair. On the other hand, the computation phase is more efficient in dEBkSP than in dAR*. Since the paths are individually computed and disseminated in dEBkSP, a vehicle only has to choose between the k paths for all the vehicles from which it has received re-routing data, which has a very low computation cost. In dAR*, a vehicle has to compute the shortest paths for all the vehicles, ranked higher than itself, from which it has received re-routing data. This computation difference is explained in detail in Section 7.

6 VANET OPTIMIZATIONS FOR RE-ROUTING INFORMATION SHARING

The effectiveness of the distributed re-routing strategies mainly depends on the amplitude of the re-routing information dissemination among vehicles. This dissemination has two dimensions that are related. The first is represented by the total number of vehicles that receive re-routing information in a congested region. The second regards the average volume of information received by the vehicles. Clearly, the higher the number of receiving vehicles and the higher the amount of information are, the more effective the re-routing process is. Ideally, each vehicle affected by congestion should receive re-routing information about all the vehicles in their region. In this case, the re-routing process can have a similar effectiveness with centralized re-routing. However, achieving this level of dissemination in VANETs is challenging for two main reasons. First, the data dissemination has to be done in real-time and therefore the dissemination time interval is short. Typically, the data dissemination phase is limited to 0.2 seconds in the system. Second, regular data dissemination in VANETs exhibits poor performance in congested areas because of wireless contention [28].

In this section, we present four optimization techniques implemented in DIVERT which are applied together to improve the data dissemination efficiency in VANETs. These techniques are: i) prioritized data dissemination, ii) k -shortest paths compression, iii) XOR coding for packet loss recovery, and iv) distance-based timer for efficient broadcast.

6.1 Prioritized Data Dissemination

DIVERT uses a prioritized dissemination to avoid that all the notified vehicles in a region start broadcasting at the same time, and thus reduce the network contention. When receiving a congestion notification, vehicle v_i waits T_i seconds before broadcasting its OD pair or its k -shortest paths. The waiting time is determined based on the rank of the vehicle defined in section 5. The rationale is that the higher rank vehicle information is more important since each vehicle computes its own path based on the higher ranked vehicle paths. Therefore, the waiting time function in equation (4) gives the higher ranked vehicles more time to disseminate their path data:

$$t_i = \alpha * rank_i^\gamma + T_{max}, \alpha = -T_{max}/rank_{max}^\gamma. \quad (4)$$

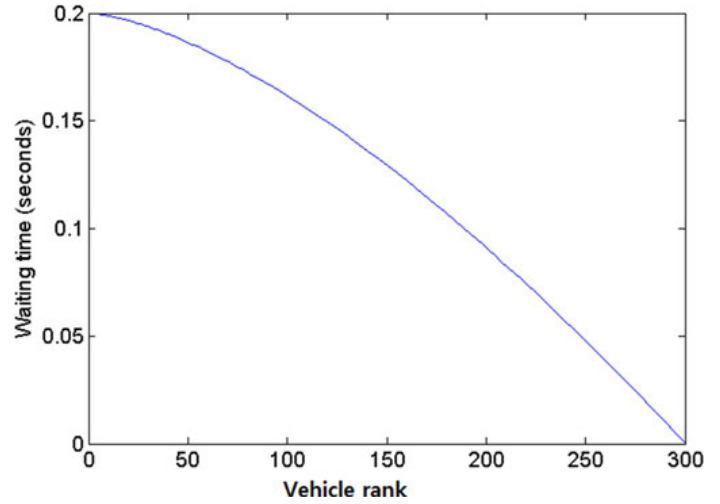


Fig. 2. The ranking function for prioritized data dissemination.

T_{max} is the total dissemination time introduced in Section 5, i.e., the time after which everyone stops disseminating and starts computing the new route. $rank_i$ is the rank of vehicle v_i and $rank_{max}$ is the maximum rank of all vehicles (e.g., the total number of selected vehicles). $rank_{max}$ is estimated by each vehicle from the road network density data received at the beginning of the re-routing process. γ is a predefined system parameter that is set to be 1.5 in our implementation. The waiting function has the following properties: i) the waiting time t_i for each vehicle is a value in the interval $[0, T_{max}]$; ii) the higher ranked vehicles wait less time than the lower ranked vehicles. Specifically, the vehicle with maximum rank transmits without waiting, while the vehicle with lowest rank waits T_{max} time.

Fig. 2 illustrates the ranking function when T_{max} is 0.2 s and $rank_{max}$ is 300. As shown, if a vehicle has high rank, it waits little time before broadcasting. Conversely, if the vehicle's rank is low, it waits a long time before broadcasting.

6.2 K Path Compression

Another option to optimize the information dissemination between vehicles is data compression. On the one hand, a large packet size increases the communication overhead and decreases the dissemination effectiveness. On the other hand, the MAC layer protocol limits the payload of a packet that can be sent on the communication channel. The dAR* re-routing strategy is very efficient from the communication point of view since vehicles only disseminate their OD pair. However, this is not the case in dEBkSP algorithm that requires vehicles to transmit their k -shortest paths. Hence, depending on the k value and the distance between the origin and destination, the size of the messages can be large. Since the k -shortest paths generally present a high degree of overlapping, a compression algorithm is proposed to exploit this feature.

Fig. 3a shows a simple example of the potential space saving that could be obtained for three paths between the roads A and J. If all the three paths are naively broadcasted, 15 spaces are needed in total. However, the three paths only cover nine distinct roads and therefore optimally need nine spaces to be transmitted. To obtain a compact representation of the k paths without any loss of information, we represent only the differences between the k paths. Specifically,

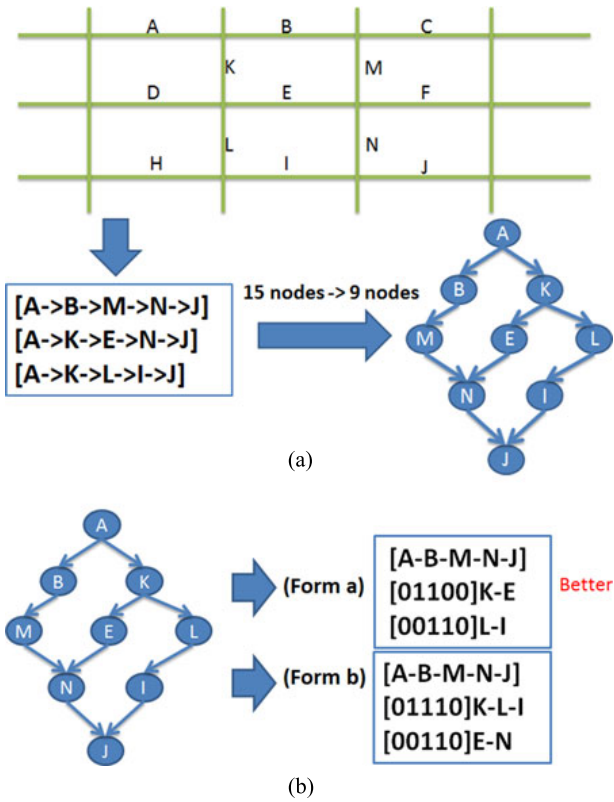


Fig. 3. Example of k-shortest paths compression.

for path i , only the edges that are different from path $i - 1$ are included in the packet. A bit vector is used to represent the position of the edges. As depicted in Fig 3b, the three paths can be represented either in form (a) or form (b). The '1' in the bit vector of each row indicates that the edge in that position is a different edge compared to the previous path. Obviously, form (a) is better than form (b) since form (a) uses one space less.

The problem comes down to finding the sequence of the k paths resulting in the best compression. However, this problem is reducible to the Hamiltonian path problem and therefore it is NP complete. Hence, a "greedy" algorithm is described to iteratively compress the k paths, based on the number of overlapping edges as shown in Algorithm 5. The function $Compress(P, P_k)$ compresses path P with respect to path P_k . The function produces a bitmap of size equal to the number of segments of P , in which bit i corresponds to the segment i in path P . Bit i is set to 0 if the segment i belongs to path P_k and to 1 otherwise. For each bit equal to 1 in the bitmap, the corresponding node id is also generated by the compression function in order to be able to re-compute path P based on P_k and the compressed value of P .

Algorithm 5. K Shortest Path Compression

```

1: procedure compresskpath()
2:  $k = KPaths.size()$  {the number of paths}
3:  $P = KPaths[0]$  {the shortest path}
4: while  $k > 0$  do
5:    $P_k = FindMostOverlappingPath(P)$ 
6:    $P = Compress(P, P_k)$ 
7:    $k = k - 1$ 
8: end while

```

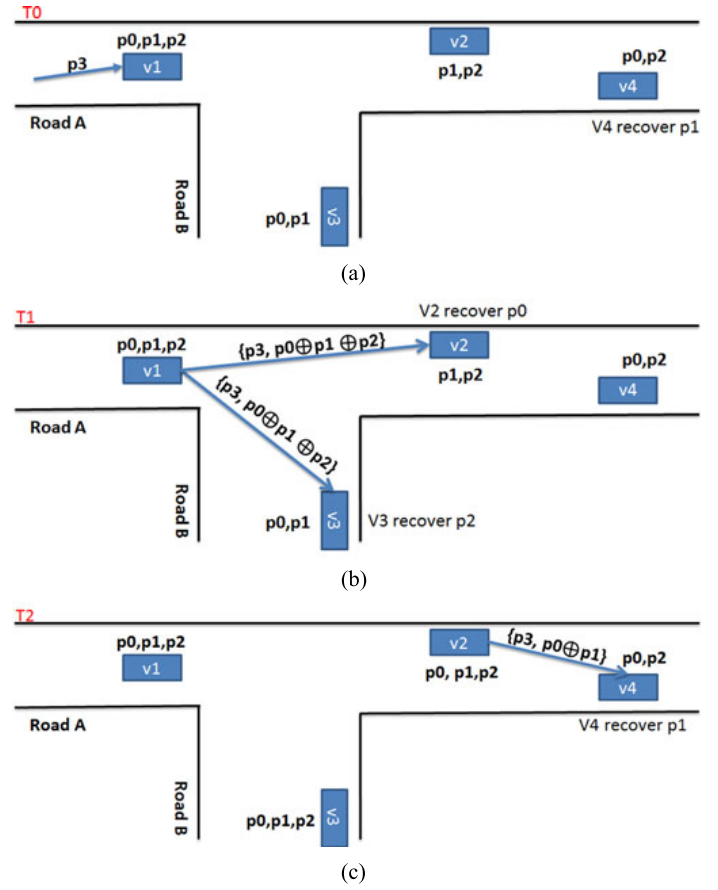


Fig. 4. Message forwarding with XOR coding field.

6.3 XOR Coding for Packet Loss Recovery

Data dissemination in VANETs can be significantly affected by packet losses. To allow vehicles to recover lost packets, a supplementary XOR coding field is appended to each transmitted packet similar to the work in [21]. When receiving a packet, a vehicle may recover one of the lost packets from the XOR field. This technique helps reducing the number of transmissions and also improves vehicles' knowledge coverage.

Fig. 4 illustrates this concept wherein four vehicles $\{v1, v2, v3, v4\}$ are presented on roads A and B. At T_0 , packet p_3 arrives at v_1 . Due to losses, the packets that the vehicle has are: $\{v_1: p_0, p_1, p_2\}$; $\{v_2: p_1, p_2\}$; $\{v_3: p_0, p_1\}$; and $\{v_4: p_0, p_2\}$. At time T_1 , v_1 broadcasts the latest received packet p_3 with an appended field $p_0 \oplus p_1 \oplus p_2$. Hence, both v_2 and v_3 can recover their missing packets p_0 and p_2 , respectively, from p_3 . Without a XOR coding field, two messages would be required to recover p_0 at v_2 and p_2 at v_3 . Similarly, at T_2 , v_2 broadcasts p_3 with the coding field $p_0 \oplus p_1$, so that v_4 can recover p_1 as well.

The question is how to figure out which packets should be coded together. For example, if v_1 appends the coding field $p_0 \oplus p_1$ instead of $p_0 \oplus p_1 \oplus p_2$, then only v_2 can recover p_0 , but v_3 cannot recover p_2 . Therefore, to find the most efficient coding, each vehicle needs to be aware of the other vehicle's packets. Using channel eavesdropping, vehicles can obtain a certain amount of knowledge of the neighborhood. To further improve this knowledge, a Bloom filter is used for a compact representation of the knowledge of each vehicle. Each time a packet is sent, the forwarding vehicle attaches a Bloom filter encoding the set of vehicles' identifiers it has received so far. This is similar to the distributed

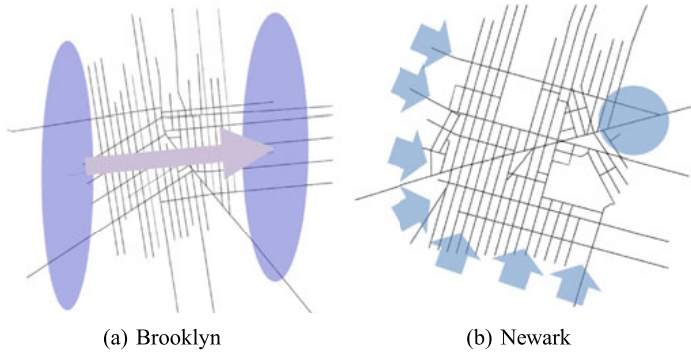


Fig. 5. Traffic flow in Brooklyn and Newark road networks.

caching solution in [9]. When a vehicle A receives a packet from a neighbor B, it obtains the current knowledge of B from the Bloom filter. In this way, each vehicle can build up a neighbor table containing the knowledge of its neighbors. This table is essential for finding the best coding according to COPE [21], and it is employed as described in Algorithm 6. Specifically, to determine the best XOR coding field, each vehicle uses its neighbor table to check if a neighbor can decode $P \oplus Q$ (line 5), where P and Q are two vehicle identifiers in the received data buffer.

Algorithm 6. Append XOR Coding Field

```

1: procedure setcodingfield()
2: P = receiveddata.pop()
3: for each neighbor i in neighbortable
4: repeat
5:   while Q = receiveddata.pop() do
6:     if neighbor i candecode (P xor Q) then
7:       P = P xor Q
8:     end if
9:   end while
10: i = i+1
11: until i = neighbortable.end
12: end for

```

6.4 Distance-Based Timer for Efficient Broadcast

In DIVERT, a distance-based timer approach is used to reduce excessive broadcasting when multiple vehicles are within communication range. After receiving a broadcast message, the vehicle waits for a certain time period until re-broadcasting the message. The waiting time period is inversely proportional to the distance between the receiving vehicle and the source vehicle. Therefore, a vehicle that is farther from the message source should re-broadcast the message earlier. During the waiting period, if the current vehicle receives copies of the message, it means that another vehicle has already forwarded the message. Thus, the current vehicle drops the message.

7 EXPERIMENTAL EVALUATION

The main objective of our simulation-based evaluation is to study the performance of the distributed re-routing strategies in DIVERT. Specifically, the evaluation has four goals:

- Assess the effectiveness and efficiency of DIVERT compared to the centralized system.

TABLE 1
Statistics of Brooklyn and Newark Networks

	Brooklyn	Newark
Network area	75.85km ²	24.82km ²
Total number of road segments	551	578
Total length of road segments	155.55km	111.41km
Total number of intersections	192	195

- Investigate the performance difference between DIVERT with and without privacy-aware traffic reporting.
- Quantify the strength of the privacy protection mechanism.
- Understand which VANET optimizations provide the most benefits.
- Compare the CPU and network load at the server between DIVERT and the centralized system.

7.1 Simulation Setup

We use Veins [34], an open source framework for running vehicular network simulations, to facilitate our experiments. TraCI [36] is employed to send commands to vehicles to change their routes. We use sections of Brooklyn, NY and of Newark, NJ for the road networks, which were downloaded in osm format from OpenStreetMap [17]. We use the Netconvert tool in SUMO [8] to convert the maps into a SUMO usable format, and Trafficmodeler [30] to generate vehicle trips. All roads have the same speed limit (13.9 m/s); some roads have one lane in each direction, while others have just one lane.

Fig. 5 shows the traffic flows. We used Trafficmodeler to generate a total of 1,000 cars from the left area to the right area in Brooklyn, while in Newark we generate bidirectional traffic with 883 cars (775 go in the congestion direction). The number of cars is chosen in such a way as to create heavy traffic and congestion (i.e., the travel time for a majority of cars is significantly higher than the free flow travel time). We generate the traffic at constant rate by deploying one car each second in the simulator. For Brooklyn, the origins and the destinations are randomly picked from the left area and the right area, respectively. For Newark, the origins are from the left and bottom areas of the map, while the destinations are in the region at the top right corner, as shown in Fig. 5. The statistics of the road networks are shown in Table 1. By default, the shortest travel time paths are automatically calculated and assigned to each vehicle at the beginning of simulation based on the road speed limits. For each scenario, we average the results over 10 runs, which is sufficient because the result variation between the runs is not significant. Tables 2 and 3 show the parameters used in the distributed re-routing algorithm and in the Veins/Omnet++ simulator, respectively.

Most of the re-routing parameters of the algorithms (i.e., the re-routing frequency, the congestion threshold, the vehicle selection depth, the number of paths k , and the repulsion weight β) are set as the default values of the centralized system [29] and offer the best trade-off between reducing the average travel time and system scalability in terms of computation time. We use the same values for DIVERT in order to be fair to the centralized version. However, since DIVERT

TABLE 2
Parameters in Distributed Re-Routing Algorithms

Period	The frequency of triggering the re-routing; by default period=450s
Threshold δ	Congestion threshold; if $K_i/K_{jam} > \delta$, the road segment is considered congested; by default $\delta = 0.7$
Level L	Network depth to select vehicles for re-routing starting from the congested segment; by default $L = 3$
# Paths k	The max number of alternative paths for each vehicle; by default $k = 8$
Repulsion weight β	The weight of repulsion in dAR*; by default $\beta = 0.05$
Broadcast timeout T_{max}	The maximum duration of the broadcast of the trip data, by default $T_{max} = 0.2s$
Privacy threshold θ	The privacy threshold; if $K_i/K_{jam} > \theta$, the vehicle starts to report the road density in the privacy enhancement component; by default $\theta = 0.5$

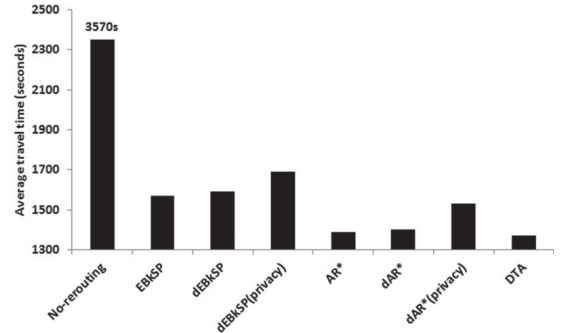
offloads the computation to individual cars, it can afford higher re-routing frequency or lower congestion threshold, and thus it is expected to obtain better results. Also, having less frequent re-routing periods reduces the number of re-routings and thus helps with the overall usability of the system. The interested reader can refer to [20] for a detailed study of the impact of the re-routing parameter values on the re-routing efficiency and effectiveness.

7.2 Results and Analysis

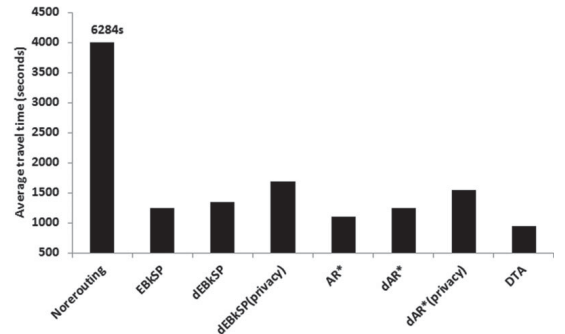
Average travel time. Fig. 6 shows the average travel time for DIVERT compared to the centralized system. DIVERT achieves similar travel time as the centralized system for both dEBkSP and dAR*, and both distributed strategies improve the travel time by more than 200 and 300 percent compared to the no re-routing case in Brooklyn and Newark, respectively. Specifically, the travel time for dEBkSP without privacy-aware traffic reporting is only 1.4 percent less in Brooklyn and 8 percent less in Newark than the time for EBkSP. When privacy-aware reporting is used, the performance decrease is 6.6 percent in Brooklyn and 25 percent in Newark. Similar results are obtained for dAR*, with performance decrease of 0.9 and 13 percent without privacy and 10.2 and 24 percent with privacy, in Brooklyn and Newark respectively.

TABLE 3
Simulation Parameters

Channel frequency	5.890e9 Hz
Propagation model	Two ray
Fading model	Jakes' model rayleigh fading
Shadowing model	LogNormal
Antenna model	Omnidirectional
Transmission power	20 mW
Propagation distance	500 m
Maximum hop	10
PHY model	802.11p
MAC model	EDCA



(a) Brooklyn



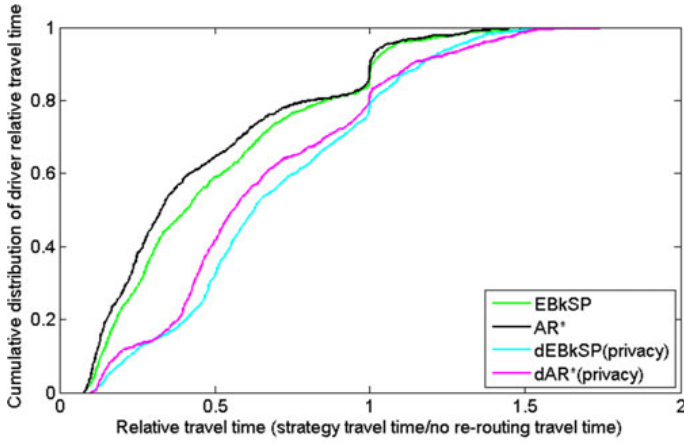
(b) Newark

Fig. 6. Average travel time ($T_{max}=0.2$ s, $k=8$).

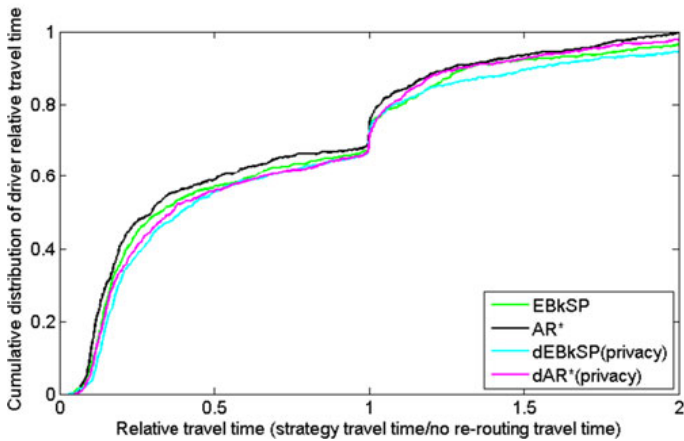
The decrease in performance observed when comparing the distributed strategies without privacy-aware reporting with the centralized strategies is due to lost packets in VANET during the path assignment phase of the strategies. The number of lost packets is small because our VANET dissemination optimizations reduce significantly the effect of network contention. Therefore, the decrease in performance due to missing global information is very small (at most 8 percent) when compared to the increase in performance obtained by these strategies w.r.t. the no re-routing case (200–300 percent).

We learn three lessons from the results: (1) Each vehicle only needs to know the trip information of its neighboring vehicles to make re-routing effective. Global information about all the vehicle routes brings minimal benefits. (2) The re-routing with privacy-aware reporting is less effective because the server may misestimate the travel time on certain road segments. However, the benefits of providing higher privacy overcome this generally acceptable performance loss. (3) dAR* achieves better performance than dEBkSP. This is due to the tiny packet size of dAR*, which leads to less contention and thus fewer packet losses. However, dEBkSP is more robust than dAR* to privacy-aware reporting and packet losses.

One important question is how good DIVERT's results are when compared to state-of-the-art research in traffic re-routing. Fig. 6 also shows the results obtained by a DTA tool [13], which attempts to achieve stochastic user equilibrium through an iterative simulation process. We use 100 iterations, double the number specified in [13]. The higher the number of iterations, the higher the probability to reach a user equilibrium state for the traffic. Despite not being a viable solution for real-time traffic guidance due to its very high computational complexity coupled with high traffic dynamics and imperfect traffic knowledge, DTA is valuable



(a) Brooklyn



(b) Newark

 Fig. 7. CDF of relative travel time ($T_{max}=0.2$ s, $k=8$).

because it gives us a lower bound on the travel time. The experimental results showed that DTA achieves results comparable with our centralized strategies. The experimental results showed that DTA achieves results comparable with our centralized strategies. AR results in an increased average travel time of 1.4 percent and 11.6 percent for Brooklyn and Newark, respectively. EBkSP results in an increased average travel time of 13.1 percent and 14 percent, respectively. However, our previous work [29] has shown that EBkSP and AR* are more computationally efficient and scalable than DTA, which makes them more appropriate for use in real-life applications. Since dEBkSP and dAR* achieve travel times just slightly less than EBkSP and AR*, we conclude that their performance is close to the lower bound provided by DTA.

Distribution of travel time. The average travel time measures the performance of the system from a global point of view. Here, the performance from a driver point of view is investigated. The relative travel time (RelT) is defined as the actual travel time divided by the travel time without re-routing. It measures the travel time gains or losses for individual drivers. Fig. 7 presents the CDF of RelT. We observe that the system manages to improve the travel time for a large majority of drivers. However, there are a few drivers who experience increased travel time. This increase is limited to less than 50 percent for most of these drivers on both Brooklyn and Newark networks. From the figure, we notice that dAR* produces better results than dEBkSP (both

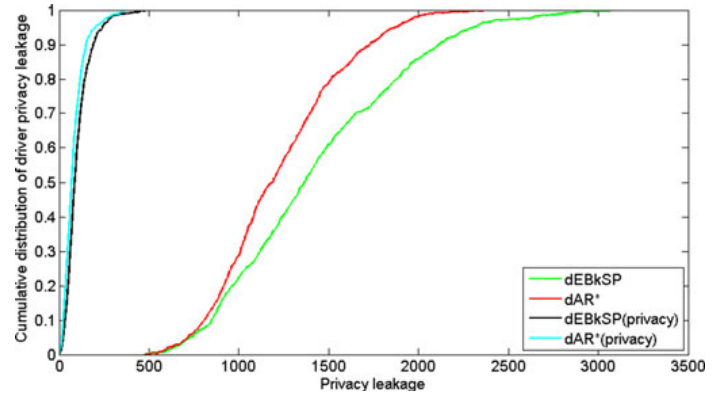


Fig. 8. CDF of privacy leakage.

include privacy-aware reporting). Compared to the centralized versions, both have just slightly worse results.

The explanation for the increase is that our focus is a system-wide optimization of the average travel time, not user equilibrium which is computationally expensive and difficult to achieve in the presence of congestion. From a practical point of view, a few bad experiences could impact the adoption rate. Therefore, we plan to investigate methods to bound this increase to low values.

Given the similarity of the results for the two networks, we provide results only for the Brooklyn network in the remainder of this section.

Distribution of privacy leakage. We use the formula described in Section 4.1 to quantify the privacy leakage as shown in Fig. 8. These results demonstrate one of the major advantages of DIVERT: it reduces the privacy leakage by up to 92 percent for both dEBkSP and dAR*. This is due to the fact that privacy-aware reporting avoids submitting location reports from highly sensitive low density roads and submits reports with low per-vehicle frequency in high density roads. Therefore, drivers are less prone to be identified by the untrusted server and their location reports are difficult to be linked to each other; thus, driver location privacy is protected. On average, dAR* has less privacy leakage than dEBkSP because it has lower travel time: in dAR*, each vehicle finishes the journey faster, and thus there are fewer location reports.

Re-routings frequency per hour. From the system point of view, having a low number of re-routings means decreasing user distraction. Fig. 9 shows the CDF of re-routing frequency per hour in the distributed strategies compared to their centralized counterparts. EBkSP and AR* have a relatively low number of re-routings due to the global knowledge of all vehicles' paths. Nevertheless, the distributed strategies only result in 4.5 and 0.4 percent more re-routings, respectively. Thus, DIVERT proves to be practical from this perspective as well.

Computation cost. In DIVERT, the bottleneck at the server of computing all the alternative paths is removed. If the computation time is high in the centralized system, the drivers would be informed too late about alternative paths, thus defeating the purpose of the system. In DIVERT, each car performs its own path computation using information received from neighboring vehicles. Therefore, this system is expected to have higher scalability. To prove this hypothesis, experiments are performed on a smart phone (the expected computing platform in the vehicles) and the

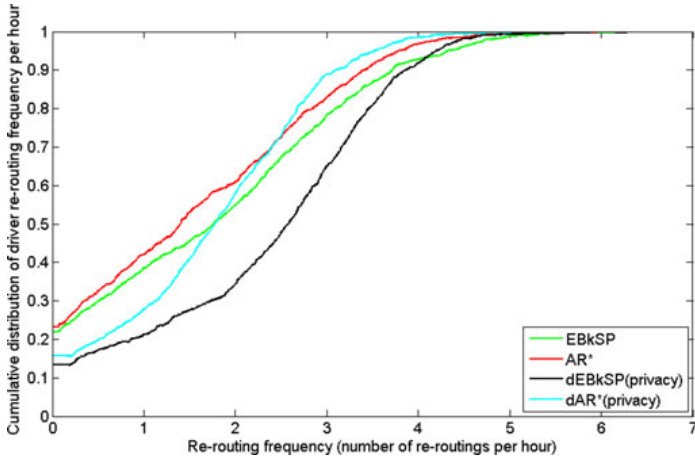


Fig. 9. CDF of re-routing frequency per hour ($T_{max}=0.2s$, $k=8$).

obtained results are plugged into analytical formulas for the two distributed strategies.

The total time consumed for dAR* is the sum of the communication time among vehicles to collect information and the actual local computation time at a vehicle. Since the number of received trip data is a function of T_{max} (the communication time), the total time consumed for dAR* is $T_{dAR^*}^{total} = T_{max} + f(T_{max}) * C(AR^*)$, where $f(T_{max})$ is the number of received re-routing data items and $C(AR^*)$ is the cost of computation to perform AR* on one OD pair.

For dEBkSP, $T_{dEBkSP}^{total} = C(k_paths) + T_{max} + f(T_{max}) * C(EBkSP)$, where $C(k_paths)$ is the computation cost for k loop-less shortest paths for one OD pair and $C(EBkSP)$ is the cost to select one path from k paths. The complexity of EBkSP only depends on k and the average path length which can be considered negligible. Therefore, dEBkSP has higher computational efficiency than dAR* since the computation time of dAR* is influenced heavily by the number of received re-routing data items whereas the computation time of dEBkSP is basically only $C(k_paths)$.

To evaluate the running time of these strategies on mobile platforms, we developed a C++ Android application on Samsung Galaxy SGH-T959V. The average computation time is measured on the initial OD pairs for the 1,000 vehicles in these experiments. Table 4 shows the computation cost of a single OD pair for each algorithm. The maximum number of received trip data items for dAR*, when $T_{max}=0.2$ s, is 82. Thus, the estimated computation time in the worst case for dAR* is $0.2+82*0.14=11.68s$. The estimated computation time for dEBkSP is $0.386+0.2=0.586s$. While both results demonstrate that DIVERT works well in practice for this scenario, it is clear that dEBkSP scales better. In larger regions with many vehicles, dAR* may not be able to meet the real-time constraints.

Impact of VANET optimizations. During our implementation and testing phase, we noticed that the prioritized broadcast and the distance-based timer approach are essential in our system because without them very little re-routing

TABLE 4
Average Computation Time for One OD Pair

dijkstra	k shortest paths $k=8$	AR*
0.244s	0.386s	0.14s

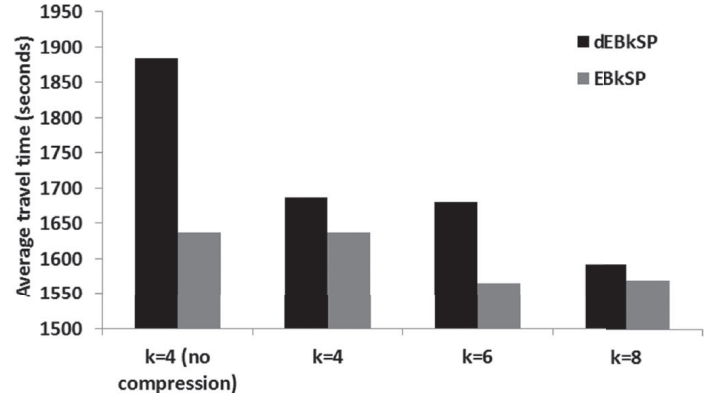


Fig. 10. The average travel time with and without compression for dEBkSP. The other optimizations are applied in both cases.

information is exchanged among the vehicles due to contention in congested regions. Among the remaining two optimizations, i.e., K path compression and XOR coding, the path compression brings the most benefits for the dEBkSP strategy. Fig. 10 shows the benefits of compression on this strategy.

We observe that compression improves the average travel time by 12 percent for $k=4$. This is due to the fact that compression reduces the packet size and improves the number of re-routing data items each vehicle can gather in VANET. When k increases, dEBkSP continues to lower the travel time. Due to the compression, when k turns from four to eight, the addition to the packet size remains small. Therefore, dEBkSP is able to achieve very similar performance as the centralized version.

Let us notice that only dEBkSP can take advantage of larger k values, while the centralized version cannot. A larger k allows for better traffic balancing but introduces higher computational complexity since the centralized server needs to compute k paths for all the selected vehicles. This is not a problem for dEBkSP which distributes the path computation to individual vehicles. Therefore, dEBkSP can result in higher performance than EBkSP when higher k values are used.

The XOR coding optimization also plays an important role in improving the data dissemination process so that each vehicle is able to receive more routing information from the nearby vehicles. The results in Fig. 11 show that XOR coding increases the average number of received re-routing data items by 41 percent for dEBkSP and by 57 percent for dAR*. The benefit is lower for dEBkSP because dEBkSP generates larger packets than dAR*, and appending the XOR coding field increases even more the

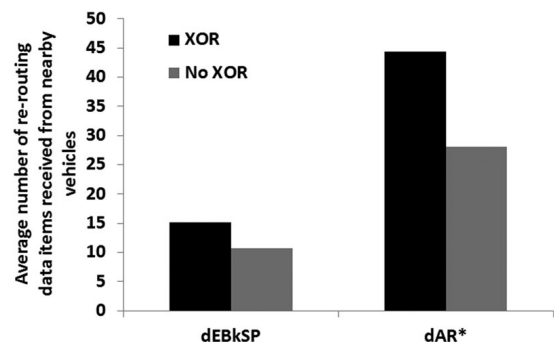


Fig. 11. The average number of received re-routing data items.

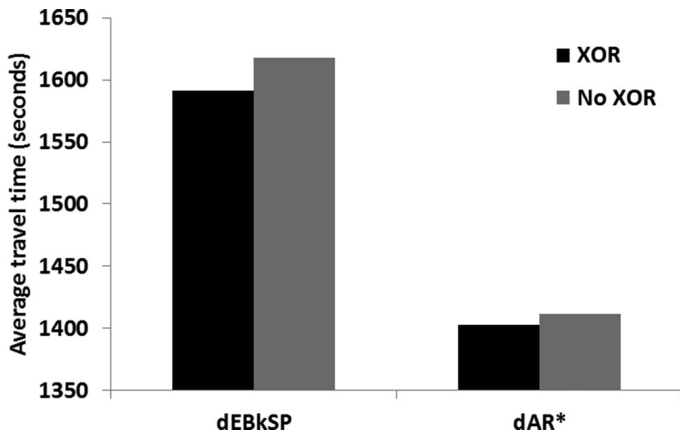


Fig. 12. The average travel time with and without XOR coding.

packet size. However, in spite of this significant improvement in the amount of disseminated data, the gain in the average travel time is only marginal as illustrated in Fig. 12. The explanation is that knowing the nearby vehicles’ path information is sufficient for dEBkSP and dAR* to provide similar re-routing effectiveness with their centralized counterparts, as already indicated in our first observation under the “Average travel time” results.

Scalability. DIVERT improves the system scalability by reducing the CPU utilization at the server side and reducing the number of messages exchanged between the server and the vehicles.

The graph computation in DIVERT has two parts. One is to update the travel time in the road network, equivalent to updating the weights of the graph’s edges. The other is to compute the shortest paths in the road network. Updating the graph has an $O(E)$ complexity, while the path computation has an $O(N \log(N + E))$ complexity. We performed an experiment to compare the two parts for our specific settings. Table 5 shows that the graph weight updating time is approximately 0.001 percent of the path computation for the centralized versions of EBkSP and AR*. Since DIVERT offloads the path computation to the vehicles and the server is only responsible for the graph weight updating, these results demonstrate a substantial CPU load reduction at the server.

DIVERT also reduces the network load on the server, which could become a major bottleneck as the number of vehicles increases. Since the privacy enhancement protocol only allows vehicles to send traffic reports when the privacy metric meets the probabilistic criterion, the number of messages is decreased by 95 percent, as indicated in Fig. 13.

8 CONCLUSIONS

The article demonstrates that a practical, cost-effective, and efficient traffic re-routing system can be implemented and deployed in real-life settings. This system, DIVERT, offloads a large part of the re-routing computation at the

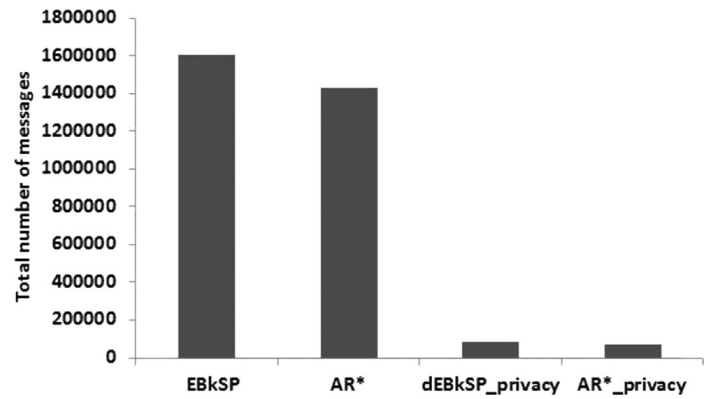


Fig. 13. Total number of messages sent between the server and vehicles.

vehicles, and thus, the re-routing process becomes scalable in real-time. To make collaborative re-routing decisions, the vehicles exchange messages over VANETs. We have optimized VANET data dissemination to allow for efficient distributed re-routing computation. In addition, the system balances user privacy with the re-routing effectiveness. The simulation results demonstrate that, compared with a centralized system, DIVERT increases the user privacy substantially, while the re-routing effectiveness is minimally impacted.

ACKNOWLEDGMENTS

This research was supported by the US National Science Foundation (NSF) under Grants No. CNS 1409523 and DGE 1565478, the National Security Agency (NSA) under Grant H98230-15-1-0274, and the French National Research Agency (ANR) under Grant No. ANR-11-INSE-0005. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, NSA, and ANR. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notice herein.

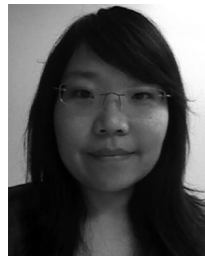
REFERENCES

- [1] (2016). [Online]. Available: <http://www.inrix.com>
- [2] (2016). [Online]. Available: <http://www.waze.com/>
- [3] (2016). [Online]. Available: <http://www.google.com/mobile/>
- [4] (2016). [Online]. Available: http://www.tomtom.com/en_gb/products/mobile-navigation/
- [5] (2016). [Online]. Available: <http://www.navigon.com/portal/us/produkte/navigationsoftware/index.html>
- [6] J. H. Banks. *Introduction to Transportation Engineering*. New York, NY, USA: McGraw-Hill, 2002.
- [7] A. Bazzan, D. Cagara, and B. Scheuermann, “An evolutionary approach to traffic assignment,” in *Proc. IEEE Symp. Comput. Intell. Vehicles Transp. Syst.*, 2014, pp. 43–50.
- [8] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, “Sumo - simulation of urban mobility: An overview,” in *Proc. 3rd Int. Conf. Adv. Syst. Simul.*, Barcelona, Spain, 2011, pp. 63–68.
- [9] A. Broder and M. Mitzenmacher, “Network applications of bloom filters: A survey,” *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.
- [10] Y.C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks, “Dynamic traffic assignment: A primer,” *Transportation Research E-Circular*, (E-C153), 2011.
- [11] R. Dingedine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” Naval Research Lab, Washington DC, Tech. Rep., 2004.
- [12] J. Gao, J. Han, D. Yang, and T. Wang, “Road network based adaptive query evaluation in vanet,” in *Proc. IEEE 9th Int. Conf. Mobile Data Manage.*, 2008, pp. 49–56.

TABLE 5
Ratio between the Cost of UpdateEdgeWeights and OD Path Computation

EBkSP $k=4$	EBkSP $k=8$	AR*
0.002%	0.001%	0.001%

- [13] C. Gawron. "Simulation-based traffic assignment—computing user equilibria in large street networks," Ph.D. dissertation, Univ. Cologne, Germany, Kln, 1999.
- [14] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Trans. Mobile Comput.*, vol. 7, no. 1, pp. 1–18, Jan. 2008.
- [15] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. 1st Int. Conf. Mobile Syst., Appl. Services*, 2003, pp. 31–42.
- [16] M. Gruteser and B. Hoh, "On the anonymity of periodic location samples," in *Proc. Security Pervasive Comput.*, 2005, pp. 179–192.
- [17] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct.–Dec. 2008.
- [18] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.
- [19] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J-C Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson, "Virtual trip lines for distributed privacy-preserving traffic monitoring," in *Proc. 6th Int. Conf. Mobile Syst., Appl. Services*, 2008, pp. 15–28.
- [20] Juan (Susan) Pan. "Vehicle re-routing strategies for congestion avoidance," Ph.D. dissertation, New Jersey Inst. Technol., Newark, NJ, 2014.
- [21] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: Practical wireless network coding," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2006, vol. 36, pp. 243–254.
- [22] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the united states," in *Proc. IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.
- [23] Z. Liang and Y. Wakahara, "Real-time urban traffic amount prediction models for dynamic route guidance systems," *EURASIP J. Wireless Commun. Netw.*, vol. 85, no. 1, pp. 1–13, 2014.
- [24] S. Lim, "Congestion-aware traffic routing for large-scale mobile agent systems," Ph.D. dissertation, Massachusetts Instit. Technol., Cambridge, MA, 2012.
- [25] R. Liu, H. Liu, D. Kwak, Y. Xiang, C. Borcea, B. Nath, and L. Iftode, "Themis: A participatory navigation system for balanced traffic routing," in *Proc. IEEE Vehicular Netw. Conf.*, 2014, pp. 159–166.
- [26] N. Loulloudes, G. Pallis, and M. D. Dikaiakos, "V-radar: A vehicular traffic query protocol for urban environments," in *Proc. 1st Int. Workshop Veh. Traffic Manage. Smart Cities*, 2012, pp. 1–6.
- [27] J. Meyerowitz and R. Roy Choudhury, "Hiding stars with fireworks: location privacy through camouflage," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 345–356.
- [28] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. 5th Annu. ACM/IEEE Int. Conf. Mobile Comput. Netw.*, 1999, pp. 151–162.
- [29] J. Pan, I. Sandu Popa, K. Zeitouni, and C. Borcea, "Proactive vehicular traffic re-routing for lower travel time," *IEEE Trans. Veh. Technol.*, vol. 62, no. 8, pp. 3551–3568, Oct. 2013.
- [30] L. G. Papaleondiou and M. D. Dikaiakos, "Trafficmodeler: A graphical tool for programming microscopic traffic simulators through high-level abstractions," in *Proc. 69th IEEE Veh. Technol. Conf.*, 2009, pp. 1–5.
- [31] R. A. Popa, A. J. Blumberg, H. Balakrishnan, and F. H. Li, "Privacy and accountability for location-based aggregate statistics," in *Proc. 18th ACM Conf. Comput. Commun. Security*, 2011, pp. 653–666.
- [32] H. Prothmann, H. Schmeck, S. Tomforde, J. Lyda, J. Hahner, C. Muller-Schloer, and J. Branke, "Decentralized route guidance in organic traffic control," in *Proc. 5th IEEE Int. Conf. Self-Adaptive Self-Organizing Syst.*, 2011, pp. 219–220.
- [33] K. Sampigethaya, M. Li, L. Huang, and R. Poovendran, "Amoeba: Robust location privacy scheme for vanet," *IEEE J. Select. Areas Commun.*, vol. 25, no. 8, pp. 1569–1589, Oct. 2007.
- [34] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved IVC analysis," *IEEE Trans. Mobile Comput.*, vol. 10, no. 1, pp. 3–15, Jan. 2011.
- [35] B. Tatomir, S. Fitrianie, M. Paltanea, and L. Rothkrantz, "Dynamic routing in traffic networks and manets using ant based algorithms," presented at the 7th Int. Conf. Artificial Evolution, Lille, France, Oct. 2005.
- [36] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J. P. Hubaux, "Traci: An interface for coupling road traffic and network simulators," in *Proc. 11th Commun. Netw. Simul. Symp.*, 2008, pp. 155–163.
- [37] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn, "A security credential management system for V2V communications," in *Proc. IEEE Veh. Netw. Conf.*, 2013, pp. 1–8.
- [38] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos, "Privacy in inter-vehicular networks: Why simple pseudonym change is not enough," in *Proc. 17th Int. Conf. Wireless On-demand Netw. Syst. Services*, 2010, pp. 176–183.
- [39] D. Wilkie, J. P. Van Den Berg, M. C. Lin, and D. Manocha, "Self-aware traffic route planning," presented at the 25th AAAI Conf. Artificial Intelligence, San Francisco, CA, USA, 2011.
- [40] T. Xu and Y. Cai, "Feeling-based location privacy protection for location-based services," in *Proc. 16th ACM Conf. Comput. Commun. Security*, 2009, pp. 348–357.
- [41] ARM. ARM Security Technology - Building a Secure System using TrustZone Technology. ARM Technical White Paper, 2009.
- [42] M. Faezipour, M. Nourani, A. Saeed, and S. Addepalli, "Progress and challenges in intelligent vehicle area networks," *Commun. ACM*, vol. 55, no. 2, pp. 90–100, 2012.
- [43] Q. C. To, B. Nguyen, and P. Pucheral, "Privacy preserving query execution using a decentralized architecture and tamper resistant hardware," in *Proc. 17th Int. Conf. Extending Database Technol.*, 2014, pp. 487–498.
- [44] D. H. Ton-That, I. Sandu-Popa, and K. Zeitouni, "Pptm: Privacy-aware participatory traffic monitoring using mobile secure probes," in *Proc. 16th IEEE Int. Conf. Mobile Data Management*, 2015, pp. 295–298.



Juan (Susan) Pan received the PhD degree from the New Jersey Institute of Technology, Newark, NJ, in 2014. Her research interests include vehicular and ubiquitous computing, distributed systems, and social network analysis.



Julian Sandu Popa received the PhD degree from the University of Versailles Saint-Quentin-en-Yvelines (UVSQ), Versailles, France, in 2009. He is currently an assistant professor of computer science with the UVSQ and a member of the Secured and Mobile Information Systems Team at Inria. His main research interests include embedded database systems, spatiotemporal databases, and mobile data management and systems, with a particular focus on topics revolving around privacy and personal data management.



Cristian Borcea received the PhD degree from Rutgers University, New Brunswick, NJ, in 2004. He is currently an associate professor with the Department of Computer Science, New Jersey Institute of Technology. He is also a visiting associate professor with the National Institute of Informatics, Tokyo, Japan. His research interests include mobile computing and sensing, ad hoc and vehicular networks, distributed systems, and cloud computing. He is a member of the ACM and USENIX.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.

Acronyms

ANR National Research Agency.

DIM Data Integration and Management.

ES-PDMS Extensive and Secure Personal Data Management System.

INRETS National Research Institute on Transports and their Security.

Inria National Institute for Research in Digital Science and Technology.

Intel SGX Intel Software Guard Extensions.

MCU Microcontroller Unit.

NJIT New Jersey Institute of Technology.

PDMS Personal Data Management System.

PETRUS PErsonal and TRUSTed cloud.

PRiSM Parallélisme, Réseaux, Systèmes, Modélisation.

SEP2P Secure and Efficient Peer-to-Peer.

SMIS Secure and Mobile Information Systems.

TEE Trusted Execution Environment.

UVSQ University of Versailles Saint-Quentin-en-Yvelines.

Bibliography

- [1] Serge Abiteboul, Benjamin André, and Daniel Kaplan. “Managing your digital life”. In: *Commun. ACM* 58.5 (2015), pp. 32–35 (cit. on pp. 18, 19).
- [2] Devesh Agrawal, Deepak Ganesan, Ramesh K. Sitaraman, Yanlei Diao, and Shashi Singh. “Lazy-Adaptive Tree: An Optimized Index Structure for Flash Devices”. In: *Proc. VLDB Endow.* 2.1 (2009), pp. 361–372 (cit. on p. 44).
- [3] Devesh Agrawal, Boduo Li, Zhao Cao, Deepak Ganesan, Yanlei Diao, and Prashant J. Shenoy. “Exploiting the Interplay between Memory and Flash Storage in Embedded Sensor Devices”. In: *16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2010*. 2010, pp. 227–236 (cit. on p. 45).
- [4] Tristan Allard, Nicolas AnCIAUX, Luc BouganIM, Yanli Guo, Lionel Le Folgoc, Benjamin Nguyen, Philippe Pucheral, Indrajit Ray, Indrakshi Ray, and Shaoyi Yin. “Secure personal data servers: a vision paper”. In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 25–35 (cit. on pp. 18, 19, 22, 59).
- [5] Tristan Allard, Georges Hébrail, Florent Masseglia, and Esther Pacitti. “Chiaroscuro: Transparency and privacy for massive personal time-series clustering”. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 2015, pp. 779–794 (cit. on p. 61).
- [6] Tristan Allard, Benjamin Nguyen, and Philippe Pucheral. “METAP: revisiting Privacy-Preserving Data Publishing using secure devices”. In: *Distributed and Parallel Databases* 32.2 (2014), pp. 191–244 (cit. on p. 61).
- [7] Mário S. Alvim, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Anna Pazzi. “Local Differential Privacy on Metric Spaces: Optimizing the Trade-Off with Utility”. In: *IEEE CSF*. 2018, pp. 262–267 (cit. on p. 83).
- [8] Nicolas AnCIAUX, Philippe Bonnet, Luc BouganIM, Benjamin Nguyen, Iulian Sandu Popa, and Philippe Pucheral. “Trusted Cells: A Sea Change for Personal Data Services”. In: *Sixth Biennial Conference on Innovative Data Systems Research, CIDR 2013, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*. www.cidrdb.org, 2013 (cit. on pp. 6, 11, 22, 35, 43, 97).
- [9] Nicolas AnCIAUX, Philippe Bonnet, Luc BouganIM, Benjamin Nguyen, Philippe Pucheral, Iulian Sandu Popa, and Guillaume Scerri. “Personal Data Management Systems: The security and functionality standpoint”. In: *Inf. Syst.* 80 (2019), pp. 13–35 (cit. on pp. 7, 11, 13, 17, 25, 28, 30, 35, 36, 59, 62, 95).

- [10] Nicolas AnCIAUX, Luc BouganIM, Philippe Pucheral, Yanli Guo, Lionel Le Folgoc, and Shaoyi Yin. “MILo-DB: a personal, secure and portable database machine”. In: *Distributed and Parallel Databases* 32.1 (2014), pp. 37–63 (cit. on pp. 40, 43, 53, 59).
- [11] Nicolas AnCIAUX, Luc BouganIM, Philippe Pucheral, Iulian Sandu Popa, and Guillaume Scerri. “Personal Database Security and Trusted Execution Environments: A Tutorial at the Crossroads”. In: *Proc. VLDB Endow.* 12.12 (2019), pp. 1994–1997 (cit. on pp. 6, 11, 35, 36, 62, 96).
- [12] Nicolas AnCIAUX, Saliha Lallali, Iulian Sandu Popa, and Philippe Pucheral. “A Scalable Search Engine for Mass Storage Smart Objects”. In: *Proc. VLDB Endow.* 8.9 (2015), pp. 910–921 (cit. on pp. 7, 11, 13, 53, 54, 85, 97).
- [13] Nicolas AnCIAUX, Saliha Lallali, Iulian Sandu-Popa, and Philippe Pucheral. “A scalable search engine for mass storage smart objects”. In: *31èmes journées Bases de Données Avancées (BDA)*. Île de Porquerolles, France, 2015 (cit. on pp. 85, 98).
- [14] Nicolas AnCIAUX, Benjamin Nguyen, and Iulian Sandu Popa. “Personal Data Management with Secure Hardware: How to Keep Your Data at Hand”. In: *2013 IEEE 14th International Conference on Mobile Data Management, Milan, Italy, June 3-6, 2013 - Volume 2*. IEEE Computer Society, 2013, pp. 1–2 (cit. on pp. 6, 35, 97).
- [15] Nicolas AnCIAUX, Benjamin Nguyen, and Iulian Sandu Popa. “Tutorial: Managing Personal Data with Strong Privacy Guarantees”. In: *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*. OpenProceedings.org, 2014, pp. 672–673 (cit. on pp. 6, 35, 97).
- [16] Nicolas AnCIAUX, Benjamin Nguyen, and Iulian Sandu Popa. “Tutorial: Towards an Era of Trust in Personal Data Management”. In: *Proceedings of the 19th East-European Conference on Advances in Databases and Information Systems (ADBIS '15)*. Poitiers, France. 2015 (cit. on pp. 35, 97).
- [17] Arvind Arasu, Spyros Blanas, Ken Eguro, Manas Joglekar, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, Prasang Upadhyaya, and Ramarathnam Venkatesan. “Secure database-as-a-service with cipherbase”. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM. 2013, pp. 1033–1036 (cit. on p. 61).
- [18] Manos Athanassoulis, Michael S. Kester, Lukas M. Maas, Radu Stoica, Stratos Idreos, Anastasia Ailamaki, and Mark Callaghan. “Designing Access Methods: The RUM Conjecture”. In: *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016*. 2016, pp. 461–466 (cit. on p. 45).
- [19] Yonatan Aumann and Yehuda Lindell. “Security against covert adversaries: Efficient protocols for realistic adversaries”. In: *Theory of Cryptography Conference*. Springer. 2007, pp. 137–156 (cit. on pp. 59, 63).
- [20] Michael Backes, Peter Druschel, Andreas Haeberlen, and Dominique Unruh. “CSAR: A Practical and Provable Technique to Make Randomized Systems Accountable.” In: *NDSS*. Vol. 9. 2009, pp. 341–353 (cit. on pp. 67–69, 71).
- [21] Manuel Barbosa, Bernardo Portela, Guillaume Scerri, and Bogdan Warinschi. “Foundations of Hardware-Based Attested Computation and Application to SGX”. In: *IEEE European Symposium on Security and Privacy, EuroS&P*. IEEE, 2016, pp. 245–260 (cit. on pp. 29, 30).

- [22] Mohammad-Mahdi Bazm, Marc Lacoste, Mario Südholt, and Jean-Marc Menaud. “Side Channels in the Cloud: Isolation Challenges, Attacks, and Countermeasures”. working paper or preprint. 2017 (cit. on p. 32).
- [23] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. “Personalized and private peer-to-peer machine learning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 473–481 (cit. on p. 83).
- [24] Elisa Bertino. “Data Security and Privacy in the IoT”. In: *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016*. OpenProceedings.org, 2016, pp. 1–3 (cit. on p. 43).
- [25] BitsAbout.me. *BitsaboutMe is a new service that empowers you to reclaim control over your personal data, in order to better protect your privacy and to get a fair deal when sharing your personal data profile with trustworthy companies and institutions*. 2012. URL: <https://bitsabout.me> (cit. on p. 20).
- [26] Matias Bjørling, Philippe Bonnet, Luc Bouganim, and Björn Þór Jónsson. “uFLIP: Understanding the Energy Consumption of Flash Devices”. In: *IEEE Data Eng. Bull.* 33.4 (2010), pp. 48–54 (cit. on p. 42).
- [27] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahhan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. “Practical secure aggregation for privacy-preserving machine learning”. In: *ACM CCS*. 2017, pp. 1175–1191 (cit. on p. 83).
- [28] Stefan Brenner, Colin Wulf, David Goltzsche, Nico Weichbrodt, Matthias Lorenz, Christof Fetzter, Peter R. Pietzuch, and Rüdiger Kapitza. “SecureKeeper: Confidential ZooKeeper using Intel SGX”. In: *Middleware*. ACM, 2016, p. 14 (cit. on p. 79).
- [29] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. “Direct anonymous attestation”. In: *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS*. ACM, 2004, pp. 132–145 (cit. on p. 29).
- [30] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. “Quantifying Differential Privacy under Temporal Correlations”. In: *33rd IEEE International Conference on Data Engineering, ICDE 2017*. 2017, pp. 821–832 (cit. on p. 61).
- [31] Robin Carpentier, Nicolas Ancaux, Iulian Sandu Popa, and Guillaume Scerri. “Performance of Large Scale Data-Oriented Operations under the TEE Constraints”. In: *34ème Conférence sur la Gestion de Données – Principes, Technologies et Applications (BDA 2018)*. Bucharest, Romania, 2018 (cit. on pp. 86, 98).
- [32] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S Wallach. “Secure routing for structured peer-to-peer overlay networks”. In: *ACM SIGOPS Operating Systems Review* 36.SI (2002), pp. 299–314 (cit. on p. 61).
- [33] Wolfie Christl, Katharina Kopp, and Patrick Urs Riechert. *Corporate Surveillance in Everyday Life. Cracked Labs*. 2017 (cit. on p. 18).
- [34] Cozy Cloud. *A smart personal cloud to gather all your data*. 2012. URL: <https://cozy.io/en> (cit. on pp. 20, 59).
- [35] CloudLocker. *CloudLocker: Your Private and Secure Personal Cloud Device*. 2013. URL: <https://www.cloudlocker.eu/en/index.html> (cit. on p. 22).

- [36] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. “Privacy at Scale: Local Differential Privacy in Practice”. In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018*. 2018, pp. 1655–1658 (cit. on p. 61).
- [37] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. “Answering Range Queries Under Local Differential Privacy”. In: *PVLDB 12.10 (2019)* (cit. on p. 83).
- [38] Intel Corp. *Intel® Software Guard Extensions Programming Reference* (cit. on pp. 13, 79).
- [39] Henry Corrigan-Gibbs and Dan Boneh. “Prio: Private, robust, and scalable computation of aggregate statistics”. In: *NSDI*. 2017, pp. 259–282 (cit. on p. 83).
- [40] Victor Costan and Srinivas Devadas. “Intel SGX Explained”. In: *IACR Cryptol. ePrint Arch.* 2016 (2016), p. 86 (cit. on pp. 32, 33, 79).
- [41] Fergus Dall, Gabrielle De Micheli, Thomas Eisenbarth, Daniel Genkin, Nadia Heninger, Ahmad Moghimi, and Yuval Yarom. “CacheQuote: Efficiently Recovering Long-term Secrets of SGX EPID via Cache Attacks”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018.2 (2018), pp. 171–191 (cit. on pp. 63, 80).
- [42] Yves-Alexandre De Montjoye, Erez Shmueli, Samuel S Wang, and Alex Sandy Pentland. “openpds: Protecting the privacy of metadata through safeanswers”. In: *PloS one* 9.7 (2014), e98790 (cit. on pp. 18, 19, 21, 59).
- [43] Yanlei Diao, Deepak Ganesan, Gaurav Mathur, and Prashant J. Shenoy. “Rethinking Data Management for Storage-centric Sensor Networks”. In: *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research*. 2007, pp. 22–31 (cit. on p. 43).
- [44] Digi.me. *See what your data can do for you with digi.me Private Sharing*. 2009. URL: <https://digi.me> (cit. on p. 20).
- [45] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. “Collecting Telemetry Data Privately”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. 2017, pp. 3574–3583 (cit. on p. 61).
- [46] J.R. Douceur. “The Sybil attack”. In: *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*. 2002, pp. 252–260 (cit. on p. 62).
- [47] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27. *On the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation, GDPR)*. 2016 (cit. on pp. 18, 61).
- [48] Diogo Fernandes, Liliana Soares, João Gomes, Mario Freire, and Pedro Inácio. “Security Issues in Cloud Environments - A Survey”. In: *Int. J. Inf. Secur.: Security in Cloud Computing* (2013) (cit. on pp. 29, 32).
- [49] Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec, Antoine Rault, François Taïani, and Jingjing Wang. “Hide & Share: Landmark-based Similarity for Private KNN Computation”. In: *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE. 2015, pp. 263–274 (cit. on p. 61).

- [50] David Froelicher, Juan Ramón Troncoso-Pastoriza, Joao Sa Sousa, and Jean-Pierre Hubaux. “Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets”. In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3035–3050 (cit. on p. 83).
- [51] Georges Gardarin, Benjamin Nguyen, Laurent Yeh, Karine Zeitouni, Bogdan Butnaru, and Iulian Sandu-Popa. “Gestion efficace de séries temporelles en P2P: Application à l’analyse technique et l’étude des objets mobiles”. In: *Bases de Données Avancées*. Namur, Belgium, 2009 (cit. on p. 99).
- [52] Javier González, Michael Hölzl, Peter Riedl, Philippe Bonnet, and René Mayrhofer. “A practical hardware-assisted approach to customize trusted boot for mobile devices”. In: *International Conference on Information Security*. Springer, 2014, pp. 542–554 (cit. on p. 59).
- [53] Hamed Haddadi, Heidi Howard, Amir Chaudhry, Jon Crowcroft, Anil Madhavapeddy, and Richard Mortier. “Personal Data: Thinking Inside the Box”. In: *CoRR* abs/1501.04737 (2015) (cit. on pp. 19, 21).
- [54] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. “Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data”. In: *ACM Trans. Comput. Syst.* 35.4 (2018), 13:1–13:32 (cit. on p. 80).
- [55] Amani S. Ibrahim, James H. Hamlyn-Harris, and John C. Grundy. “Emerging Security Challenges of Cloud Virtual Infrastructure”. In: *CoRR* abs/1612.09059 (2016) (cit. on p. 32).
- [56] Christopher M. Jermaine, Edward Omiecinski, and Wai Gen Yee. “The partitioned exponential file for database storage management”. In: *VLDB J.* 16.4 (2007), pp. 417–437 (cit. on p. 44).
- [57] Jonathan Katz. “Universally Composable Multi-party Computation Using Tamper-Proof Hardware”. In: *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by Moni Naor. Vol. 4515. Springer, 2007, pp. 115–128 (cit. on p. 29).
- [58] Anne-Marie Kermarrec and François Taïani. “Want to scale in centralized systems? Think P2P”. In: *J. Internet Serv. Appl.* 6.1 (2015), 16:1–16:12 (cit. on p. 59).
- [59] Ahmed Kharrat, Iulian Sandu Popa, Karine Zeitouni, and Sami Faïz. “Caractérisation de la densité de trafic et de son évolution à partir de trajectoires d’objets mobiles”. In: *Actes des 5èmes journées francophones Mobilité et Ubiquité 2009, UBIMOB’09, 7-8 Juillet 2009, Lille, France*. Vol. 394. ACM International Conference Proceeding Series. ACM, 2009, pp. 33–40 (cit. on p. 99).
- [60] Ahmed Kharrat, Iulian Sandu Popa, Karine Zeitouni, and Sami Faïz. “Clustering Algorithm for Network Constraint Trajectories”. In: *Headway in Spatial Data Handling, 13th International Symposium on Spatial Data Handling*. Springer, 2008, pp. 631–647 (cit. on pp. 4, 98).
- [61] Ahmed Kharrat, Karine Zeitouni, Iulian Sandu Popa, and Sami Faïz. “Characterizing the Traffic Density and Its Evolution through Moving Object Trajectories”. In: *KDIR 2009 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*. INSTICC Press, 2009, pp. 319–322 (cit. on p. 97).

- [62] Ahmed Kharrat, Karine Zeitouni, Iulian Sandu Popa, and Sami Faiz. “Characterizing Traffic Density and Its Evolution through Moving Object Trajectories”. In: *Fifth International Conference on Signal-Image Technology & Internet-Based Systems, SITIS 2009*. IEEE Computer Society, 2009, pp. 257–263 (cit. on p. 98).
- [63] Deokjin Kim, DaeHee Jang, Minjoon Park, Yunjong Jeong, Jonghwan Kim, Seokjin Choi, and Brent ByungHoon Kang. “SGX-LEGO: Fine-grained SGX controlled-channel attack and its countermeasure”. In: *Computers & Security* 82 (2019), pp. 118–139 (cit. on pp. 63, 80).
- [64] Peter F. Klemperer, Yuan Liang, Michelle L. Mazurek, Manya Sleeper, Blase Ur, Lujo Bauer, Lorrie Faith Cranor, Nitin Gupta, and Michael K. Reiter. “Tag, you can see it!: using tags for access control in photo sharing”. In: *CHI Conference on Human Factors in Computing Systems, CHI '12*. ACM, 2012, pp. 377–386 (cit. on p. 52).
- [65] Riad Ladjel, Nicolas Anciaux, Philippe Pucheral, and Guillaume Scerri. “A manifest-based framework for organizing the management of personal data at the edge of the network”. In: *ISD. 2019* (cit. on p. 64).
- [66] Riad Ladjel, Nicolas Anciaux, Philippe Pucheral, and Guillaume Scerri. “Trustworthy Distributed Computations on Personal Data Using Trusted Execution Environments”. In: *TrustCom. 2019* (cit. on p. 64).
- [67] Saliha Lallali. “A scalable search engine for the Personal Cloud”. 2016SACLV009. PhD thesis. 2016 (cit. on pp. 13, 39, 85).
- [68] Saliha Lallali, Nicolas Anciaux, Iulian Sandu Popa, and Philippe Pucheral. “A Secure Search Engine for the Personal Cloud”. In: *ACM SIGMOD*. ACM, 2015, pp. 1445–1450 (cit. on pp. 7, 11–13, 40, 54, 85, 97).
- [69] Saliha Lallali, Nicolas Anciaux, Iulian Sandu Popa, and Philippe Pucheral. “Supporting secure keyword search in the personal cloud”. In: *Inf. Syst.* 72 (2017), pp. 1–26 (cit. on pp. 7, 39, 42, 48, 49, 51, 53, 54, 59, 85, 95).
- [70] Thu Le, Nicolas Anciaux, Sébastien Guilloton, Saliha Lallali, Philippe Pucheral, Iulian Sandu Popa, and Chao Chen. “Distributed Secure Search in the Personal Cloud”. In: *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016*. OpenProceedings.org, 2016, pp. 652–655 (cit. on pp. 7, 8, 12, 13, 40, 53, 54, 61, 96).
- [71] Sangmin Lee, Edmund L Wong, Deepak Goel, Mike Dahlin, and Vitaly Shmatikov. “ π Box: A Platform for Privacy-Preserving Apps.” In: *NSDI*. 2013, pp. 501–514 (cit. on p. 59).
- [72] Wolfgang Lehner. “The Data Center under your Desk - How Disruptive is Modern Hardware for DB System Design?” In: *Proc. VLDB Endow.* 10.12 (2017), pp. 2018–2019 (cit. on p. 35).
- [73] Matthew Lentz, Rijurekha Sen, Peter Druschel, and Bobby Bhattacharjee. “SeCloak: ARM Trustzone-based Mobile Peripheral Control”. In: *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2018*. ACM, 2018, pp. 1–13 (cit. on p. 30).

- [74] Tingli Li, Yang Liu, Ye Tian, Shuo Shen, and Wei Mao. “A Storage Solution for Massive IoT Data Based on NoSQL”. In: *2012 IEEE International Conference on Green Computing and Communications, Conference on Internet of Things, and Conference on Cyber, Physical and Social Computing, GreenCom/iThings/CPSCoM 2012*. 2012, pp. 50–57 (cit. on p. 43).
- [75] Yinan Li, Bingsheng He, Jun Yang, Qiong Luo, and Ke Yi. “Tree Indexing on Solid State Drives”. In: *Proc. VLDB Endow.* 3.1 (2010), pp. 1195–1206 (cit. on p. 44).
- [76] Julien Loudet. “Distributed and Privacy-Preserving Personal Queries on Personal Clouds. (Requêtes distribuées et respectueuses de la vie privée de Nuages Personnels)”. PhD thesis. Versailles Saint-Quentin-en-Yvelines University, France, 2019 (cit. on pp. 58, 74, 75, 85).
- [77] Julien Loudet, Luc Bouganim, and Iulian Sandu Popa. “Privacy-Preserving Queries on Highly Distributed Personal Data Management Systems”. In: *34^{ème} Conférence sur la Gestion de Données – Principes, Technologies et Applications*. Proceedings of the BDA 2018 Conference. Bucharest, Romania, 2018 (cit. on pp. 85, 98).
- [78] Julien Loudet, Iulian Sandu Popa, and Luc Bouganim. “DISPERS: Securing Highly Distributed Queries on Personal Data Management Systems”. In: *PVLDB* 12.12 (2019), pp. 1886–1889 (cit. on pp. 7, 8, 11, 12, 74, 85, 96).
- [79] Julien Loudet, Iulian Sandu Popa, and Luc Bouganim. “SEP2P: Secure and Efficient P2P Personal Data Processing”. In: *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. OpenProceedings.org, 2019, pp. 145–156 (cit. on pp. 7, 8, 11, 58, 73, 74, 85, 96).
- [80] Julien Loudet, Iulian Sandu-Popa, and Luc Bouganim. “DISPERS: Securing Highly Distributed Queries on Personal Data Management Systems”. In: *BDA 2019 - 35^{ème} Conférence sur la Gestion de Données - Principes, Technologies et Applications*. 2019, Demo paper. (Cit. on pp. 12, 85, 98).
- [81] Julien Loudet, Iulian Sandu-Popa, and Luc Bouganim. “SEP2P: Secure and Efficient P2P Personal Data Processing”. In: *BDA 2019 - 35^{ème} Conférence sur la Gestion de Données - Principes, Technologies et Applications*. Lyon, France, 2019 (cit. on pp. 85, 98).
- [82] Julien Loudet, Iulian Sandu-Popa, and Luc Bouganim. “SEP2P: Secure and Efficient P2P Personal Data Processing”. In: *APVP 2019 - Atelier sur la Protection de la Vie Privée*. Cap Hornu, France, 2019 (cit. on pp. 85, 98).
- [83] Petar Maymounkov and David Mazieres. “Kademlia: A peer-to-peer information system based on the xor metric”. In: *International Workshop on Peer-to-Peer Systems*. Springer. 2002, pp. 53–65 (cit. on p. 60).
- [84] Michelle L. Mazurek, J. P. Arsenault, Joanna Bresee, Nitin Gupta, Iulia Ion, Christina Johns, Daniel Lee, Yuan Liang, Jenny Olsen, Brandon Salmon, Richard Shay, Kami Vaniea, Lujio Bauer, Lorrie Faith Cranor, Gregory R. Ganger, and Michael K. Reiter. “Access control for home data sharing: evaluating social acceptability”. In: *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010*. ACM, 2010, pp. 645–654 (cit. on p. 52).

- [85] Michelle L. Mazurek, Peter F. Klemperer, Richard Shay, Hassan Takabi, Lujo Bauer, and Lorrie Faith Cranor. “Exploring reactive access control”. In: *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011*. ACM, 2011, pp. 2085–2094 (cit. on p. 52).
- [86] Michelle L. Mazurek, Yuan Liang, William Melicher, Manya Sleeper, Lujo Bauer, Gregory R. Ganger, Nitin Gupta, and Michael K. Reiter. “Toward strong, usable access control for shared distributed data”. In: *Proceedings of the 12th USENIX conference on File and Storage Technologies, FAST 2014*. USENIX, 2014, pp. 89–103 (cit. on p. 52).
- [87] Jonathan M. McCune, Yanlin Li, Ning Qu, Zongwei Zhou, Anupam Datta, Virgil D. Gligor, and Adrian Perrig. “TrustVisor: Efficient TCB Reduction and Attestation”. In: *31st IEEE Symposium on Security and Privacy, S&P 2010*. 2010, pp. 143–158 (cit. on pp. 32, 33).
- [88] Meeco. *Meeco — the distributed technology powering consent and personal data*. 2012. URL: <https://www.meeco.me> (cit. on p. 20).
- [89] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996 (cit. on p. 66).
- [90] MesInfos. *The MesInfos project explores and implements the self data concept in France*. 2012. URL: <http://mesinfos.fing.org/english> (cit. on p. 20).
- [91] Yilin Mo and Richard M Murray. “Privacy preserving average consensus”. In: *IEEE Transactions on Automatic Control* 62.2 (2016), pp. 753–765 (cit. on p. 83).
- [92] MyData.org. *MyData Global’s mission is to empower individuals by improving their right to self-determination regarding their personal data*. 2014. URL: <https://mydata.org> (cit. on p. 20).
- [93] Nextcloud. *The self-hosted productivity platform that keeps you in control*. 2016. URL: <https://nextcloud.com> (cit. on pp. 20, 59).
- [94] Inc. Novathings. *helixee — The French cloud that respects your privacy*. URL: <http://www.helixee.me/home/> (cit. on p. 22).
- [95] Patrick E. O’Neil, Edward Cheng, Dieter Gawlick, and Elizabeth J. O’Neil. “The Log-Structured Merge-Tree (LSM-Tree)”. In: *Acta Informatica* 33.4 (1996), pp. 351–385 (cit. on p. 44).
- [96] Susan Juan Pan, Mohammad A. Khan, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. “Proactive Vehicle Re-routing Strategies for Congestion Avoidance”. In: *IEEE 8th International Conference on Distributed Computing in Sensor Systems, DCOSS 2012*. IEEE Computer Society, 2012, pp. 265–272 (cit. on pp. 9, 97).
- [97] Susan Juan Pan, Iulian Sandu Popa, and Cristian Borcea. “DIVERT: A Distributed Vehicular Traffic Re-Routing System for Congestion Avoidance”. In: *IEEE Trans. Mob. Comput.* 16.1 (2017), pp. 58–72 (cit. on pp. 10, 95).
- [98] Susan Juan Pan, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. “Proactive Vehicular Traffic Rerouting for Lower Travel Time”. In: *IEEE Trans. Vehicular Technology* 62.8 (2013), pp. 3551–3568 (cit. on pp. 9, 96).
- [99] PlugDB. *PlugDB: a secure personal server*. URL: <https://project.inria.fr/plugdb/en/> (cit. on p. 5).

- [100] Iulian Sandu Popa, Ahmed Kharrat, Karine Zeitouni, and Guillaume Saint-Pierre. “Base de données de capteurs à localisation mobile Modèle et langage”. In: *Ingénierie des Systèmes d’Inf.* 14.5 (2009), pp. 35–58 (cit. on pp. 4, 96).
- [101] Iulian Sandu Popa, Dai Hai Ton That, Karine Zeitouni, and Cristian Borcea. “Mobile participatory sensing with strong privacy guarantees using secure probes”. In: *GeoInformatica, online first* (2019), p. 48 (cit. on pp. 10, 22, 55, 95).
- [102] Iulian Sandu Popa, Karine Zeitouni, Georges Gardarin, Didier Nakache, and Elisabeth Métais. “Text Categorization for Multi-label Documents and Many Categories”. In: *20th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2007)*. IEEE Computer Society, 2007, pp. 421–426 (cit. on p. 98).
- [103] Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, Dominique Barth, and Sandrine Vial. “Indexing in-network trajectory flows”. In: *VLDB J.* 20.5 (2011), pp. 643–669 (cit. on pp. 4, 8, 96).
- [104] Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, Dominique Barth, and Sandrine Vial. “PARINET: A tunable access method for in-network trajectories”. In: *ICDE*. IEEE Computer Society, 2010, pp. 177–188 (cit. on pp. 4, 8, 97).
- [105] Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, and Ahmed Kharrat. “Spatio-temporal compression of trajectories in road networks”. In: *GeoInformatica* 19.1 (2015), pp. 117–145 (cit. on pp. 5, 9, 96).
- [106] Raluca Ada Popa, Andrew J Blumberg, Hari Balakrishnan, and Frank H Li. “Privacy and accountability for location-based aggregate statistics”. In: *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 653–666 (cit. on p. 61).
- [107] Christian Priebe, Kapil Vaswani, and Manuel Costa. “EnclaveDB: A Secure Database Using SGX”. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2018, pp. 264–278 (cit. on pp. 59, 79).
- [108] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. “A scalable content-addressable network”. In: *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. Ed. by Rene L. Cruz and George Varghese. ACM, 2001, pp. 161–172 (cit. on pp. 36, 60, 73).
- [109] NST Council report. *Task Force on Smart Disclosure. Smart Disclosure and Consumer Decision Making*. 2013 (cit. on p. 18).
- [110] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. “Trusted Execution Environment: What It is, and What It is Not”. In: *TrustCom/BigDataSE/ISPA (1)*. IEEE, 2015, pp. 57–64 (cit. on pp. 6, 32, 63).
- [111] Eyad Saleh, Ahmad Alsa’deh, Ahmad Kayed, and Christoph Meinel. “Processing over encrypted data: between theory and practice”. In: *ACM SIGMOD Record* 45.3 (2016), pp. 5–16 (cit. on pp. 59, 61).
- [112] Iulian Sandu Popa and Karine Zeitouni. “Modeling and Querying Mobile Location Sensor Data”. In: *The Fourth International Conference on Advanced Geographic Information Systems, Applications, and Services (GEOProcessing 2012)*. 2012, 18pp (cit. on pp. 4, 10, 97).

- [113] Iulian Sandu-Popa, A. Kharrat, and Karine Zeitouni. “CALM : Un système de gestion de données de CApteurs à Localisation Mobile”. In: *Journées SAGEO*. Montpellier, France, 2008 (cit. on p. 99).
- [114] Iulian Sandu-Popa, A. Kharrat, Karine Zeitouni, F. Dupin, and G. Saint-Pierre. “Gestion de données spatiotemporelle des observations sur la conduite en situation naturelle”. In: *2ème atelier SIT (Systèmes d’Information en Transport)*. Fontainebleau, France, 2008 (cit. on p. 99).
- [115] Iulian Sandu-Popa and Karine Zeitouni. “Modélisation et interrogation de données de capteurs à localisation mobile”. In: *Bases de Données Avancées*. Namur, Belgium, 2009 (cit. on p. 99).
- [116] Iulian Sandu-Popa, Karine Zeitouni, G. Saint-Pierre, F. Dupin, and S. Glaser. “Using in-Vehicle Sensor Data for Naturalistic Driving Analysis”. In: *World Congress on Intelligent Transportation Systems*. United States, 2008, p. 6 (cit. on p. 98).
- [117] Ardalan Amiri Sani. “SchrodinText: Strong Protection of Sensitive Textual Content of Mobile Applications”. In: *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys’17*. ACM, 2017, pp. 197–210 (cit. on p. 30).
- [118] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. “VC3: Trustworthy Data Analytics in the Cloud Using SGX”. In: *2015 IEEE Symposium on Security and Privacy, SP 2015*. IEEE Computer Society, 2015, pp. 38–54 (cit. on p. 79).
- [119] Russell Sears and Raghu Ramakrishnan. “bLSM: a general purpose log structured merge tree”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012*. ACM, 2012, pp. 217–228 (cit. on p. 44).
- [120] Solid. *Solid empowers users and organizations to separate their data from the applications that use it*. 2018. URL: <https://solid.inrupt.com/> (cit. on p. 59).
- [121] SpiderOak. *SpiderOak Share provides a secure way to exchange and sync your files using No Knowledge Encryption*. 2018. URL: <https://spideroak.com/spideroak-share> (cit. on pp. 18, 21).
- [122] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. “Chord: A scalable peer-to-peer lookup service for internet applications”. In: *ACM SIGCOMM Computer Communication Review* 31.4 (2001), pp. 149–160 (cit. on pp. 36, 60, 61, 73).
- [123] Sync. *Sync’s end-to-end encrypted storage platform and apps ensure that only you can access your data in the cloud*. 2011. URL: <https://www.sync.com> (cit. on p. 21).
- [124] Chiu Chiang Tan, Bo Sheng, Haodong Wang, and Qun Li. “Microsearch: A search engine for embedded devices used in pervasive computing”. In: *ACM Trans. Embedded Comput. Syst.* 9.4 (2010), 43:1–43:29 (cit. on pp. 41, 43, 53).
- [125] Chiu Chiang Tan, Bo Sheng, Haodong Wang, and Qun Li. “Microsearch: When Search Engines Meet Small Devices”. In: *Pervasive Computing*. Vol. 5013. 2008, pp. 93–110 (cit. on p. 43).
- [126] ARM Security Technology. *Building a Secure System using TrustZone Technology*. Tech. rep. ARM, Dec. 2008 (cit. on pp. 32, 63).

- [127] Dai Hai Ton That. “Efficient management and secure sharing of mobility traces. (Gestion efficace et partage sécurisé des traces de mobilité)”. PhD thesis. University of Paris-Saclay, France, 2016 (cit. on pp. 13, 39, 85).
- [128] Dai Hai Ton That, Iulian Sandu Popa, and Karine Zeitouni. “PPTM: Privacy-Aware Participatory Traffic Monitoring Using Mobile Secure Probes”. In: *16th IEEE International Conference on Mobile Data Management, MDM 2015*. IEEE Computer Society, 2015, pp. 295–298 (cit. on pp. 10–13, 55, 85, 97).
- [129] Dai Hai Ton That, Iulian Sandu Popa, and Karine Zeitouni. “TRIFL: A Generic Trajectory Index for Flash Storage”. In: *ACM Trans. Spatial Algorithms and Systems* 1.2 (2015), 6:1–6:44 (cit. on pp. 9, 11, 13, 55, 85, 96).
- [130] Dai Hai Ton That, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. “PAM-PAS: Privacy-Aware Mobile Participatory Sensing Using Secure Probes”. In: *Proceedings of the 28th International Conference on Scientific and Statistical Database Management, SSDBM 2016, Budapest, Hungary, July 18-20, 2016*. ACM, 2016, 4:1–4:12 (cit. on pp. 10, 13, 22, 55, 59, 61, 85, 96).
- [131] Dai Hai Ton That, Iulian Sandu Popa, Karine Zeitouni, and Cristian Borcea. “Un protocole basé sur des mobiles sécurisés pour une collecte participative de données spatiales en mobilité réellement anonyme”. In: *Revue Internationale de Géomatique* 26.2 (2016), pp. 185–210 (cit. on pp. 85, 96).
- [132] Quoc-Cuong To, Benjamin Nguyen, and Philippe Pucheral. “Privacy-Preserving Query Execution using a Decentralized Architecture and Tamper Resistant Hardware”. In: *EDBT*. OpenProceedings.org, 2014, pp. 487–498 (cit. on pp. 43, 53).
- [133] Quoc-Cuong To, Benjamin Nguyen, and Philippe Pucheral. “Private and Scalable Execution of SQL Aggregates on a Secure Decentralized Architecture”. In: *ACM Trans. Database Syst.* 41.3 (2016), 16:1–16:43 (cit. on pp. 22, 59, 61).
- [134] J. C. Tomàs, B. Amann, N. Travers, and D. Vodislav. “RoSeS: a continuous query processor for large-scale RSS filtering and aggregation”. In: *Proc. of the 20th ACM Conf. on Information and Knowledge Management*. 2011, pp. 2549–2552 (cit. on p. 59).
- [135] Dai Hai Ton That, Iulian Sandu-Popa, and Karine Zeitouni. “PPTM: Privacy-aware Participatory Traffic Monitoring Using Mobile Secure Probes”. In: *31èmes journées Bases de Données Avancées (BDA '15)*. Île de Porquerolles, France, 2015, 4 pages (Demo paper) (cit. on pp. 12, 85, 99).
- [136] Dai Hai Ton That, Iulian Sandu-Popa, Karine Zeitouni, and Cristian Borcea. “PAM-PAS: Collecte participative respectueuse de la vie privée basée sur des mobiles sécurisés”. In: *31èmes journées Bases de Données Avancées (BDA '15)*. Île de Porquerolles, France, 2015 (cit. on pp. 85, 98).
- [137] Nicolas Tsiftes and Adam Dunkels. “A database in every sensor”. In: *Proceedings of the 9th International Conference on Embedded Networked Sensor Systems, SenSys 2011*. ACM, 2011, pp. 316–332 (cit. on pp. 43, 53).
- [138] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. “A survey of DHT security techniques”. In: *ACM Computing Surveys (CSUR)* 43.2 (2011), p. 8 (cit. on pp. 64, 68).

- [139] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. “Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution”. In: *Proceedings of the 27th USENIX Security Symposium*. USENIX Association, Aug. 2018 (cit. on pp. 63, 80).
- [140] Baobing Wang and John S. Baras. “HybridStore: An Efficient Data Management System for Hybrid Flash-Based Sensor Devices”. In: *Wireless Sensor Networks - 10th European Conference, EWSN 2013*. Vol. 7772. 2013, pp. 50–66 (cit. on p. 43).
- [141] Haodong Wang, Chiu Chiang Tan, and Qun Li. “Snoogle: A Search Engine for Pervasive Environments”. In: *IEEE Trans. Parallel Distrib. Syst.* 21.8 (2010), pp. 1188–1202 (cit. on p. 43).
- [142] Qiyan Wang and Nikita Borisov. “Octopus: A secure and anonymous DHT lookup”. In: *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 325–334 (cit. on pp. 60, 61).
- [143] Samuel Weiser, Luca Mayr, Michael Schwarz, and Daniel Gruss. “SGXJail: Defeating Enclave Malware via Confinement”. In: *22nd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2019, Chaoyang District, Beijing, China, September 23-25, 2019*. USENIX Association, 2019, pp. 353–366 (cit. on p. 80).
- [144] Chin-Hsien Wu, Tei-Wei Kuo, and Li-Ping Chang. “An efficient B-tree layer implementation for flash-memory storage systems”. In: *ACM Trans. Embedded Comput. Syst.* 6.3 (2007), p. 19 (cit. on p. 44).
- [145] Tingxin Yan, Deepak Ganesan, and R. Manmatha. “Distributed image search in camera sensor networks”. In: *Proceedings of the 6th International Conference on Embedded Networked Sensor Systems, SenSys 2008*. ACM, 2008, pp. 155–168 (cit. on pp. 41, 43).
- [146] Ching-man Au Yeung, Lalana Kagal, Nicholas Gibbins, and Nigel Shadbolt. “Providing Access Control to Online Photo Albums Based on Tags and Linked Data”. In: *Social Semantic Web: Where Web 2.0 Meets Web 3.0, Papers from the 2009 AAAI Spring Symposium, Technical Report SS-09-08*. AAAI, 2009, pp. 9–14 (cit. on p. 52).
- [147] Karine Zeitouni, Iulian Sandu-Popa, and Ahmed Kharrat. “Exploitation des traces sur la mobilité”. In: *Conférence francophone sur les Technologies de l’Information, de la Communication et de la Géolocalisation dans les Systèmes de Transports*. 2009 (cit. on p. 99).
- [148] Kai Zheng, Wenlong Mou, and Liwei Wang. “Collect at Once, Use Effectively: Making Non-interactive Locally Private Learning Possible”. In: *ICML*. Vol. 70. 2017 (cit. on p. 83).
- [149] Justin Zobel and Alistair Moffat. “Inverted files for text search engines”. In: *ACM Comput. Surv.* 38.2 (2006), p. 6 (cit. on pp. 41, 42, 47, 53).

