



**HAL**  
open science

# Secure and Distributed Computations for a Personal Data Management System

Riad Ladjel, Nicolas Ancaux, Guillaume Scerri, Philippe Pucheral

► **To cite this version:**

Riad Ladjel, Nicolas Ancaux, Guillaume Scerri, Philippe Pucheral. Secure and Distributed Computations for a Personal Data Management System. 34ème Conférence sur la Gestion de Données – Principes, Technologies et Applications, Oct 2018, Bucarest, Romania. hal-01947808

**HAL Id: hal-01947808**

**<https://inria.hal.science/hal-01947808v1>**

Submitted on 7 Dec 2018

**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.

# Secure and distributed computations for a personal data management system

Riad Ladjel  
Inria & U. Versailles  
Versailles, France  
riad.ladjel@inria.fr

Nicolas AnCIAUX  
Inria & U. Versailles  
Versailles, France  
nicolas.anciaux@inria.fr

Guillaume Scerri  
U. Versailles & Inria  
Versailles, France  
guillaume.scerri@uvsq.fr

Philippe Pucheral  
U. Versailles & Inria  
Versailles, France  
philippe.pucheral@uvsq.fr

## INTRODUCTION

A large amount of economically valuable data is produced every day. Whether they come from smartphones, connected devices, sensors or smart meters, this data is a gold mine for the people holding it. This data is often stored in centralized servers, servers that usually belong to large corporations that collected it in the first place (Google, Amazon, Facebook, insurance companies etc.) The result of this situation is that users lose control over their own data. This represents a real threat to privacy, whether it is intentional (misuse, malicious attack), or just by negligence (data leakage, mismanagement). These threats point to the need for personal platforms which allow their users to collect, manage and share their own data. This is the essence of the self-data movement. Thanks to smart disclosure initiatives, users can access their personal data from the companies or government agencies that collected them. Concurrently, personal data management system (*PDMS*) solutions are flourishing. Their goal is to empower users to leverage their personal data for their own good.

While storing data in *PDMS* increases user control over data, in the *PDMS* context collaborative use of data is often overlooked. The benefits derived from exploiting data are considerable. A user may want to share her GPS position to have accurate traffic prediction [9], or her medical records to train a shared neural network so that it can detect several diseases [5]. She may also want to adapt her energy contract based on her actual consumption without jeopardizing her privacy [3]. A naive approach to this problem is to send personal data to a trusted third party who will perform said collaborative computations. This, however involves very strong trust in the third party's honesty. The goal of this work is to overcome this unrealistic trust assumption and propose a privacy preserving distributed computation framework for performing collaborative computations over a large number of *PDMS*.

Our objective is to propose a secure distributed computation protocol offering the same guaranties as secure multi-party computation (*SMC*) [7] while keeping a profit-to-cost ratio as low as possible in case of an attack. In other words, it should be possible for parties to compute any function  $f(x_1, x_2, \dots, x_n)$  over their inputs while keeping them private. An individual cannot infer any information other than what she can infer from her own input and the final result. The result should be exact and not approximate. Unlike *SMC*, the proposed protocol should be scalable, with respect to the number of parties and the complexity of the computation.

## SHORTCOMINGS OF EXISTING SOLUTIONS

**Secure multi-party computation (*SMC*)** [7] allows  $N$  users to perform a secure computation on their personal data. The users involved learn nothing more than what they can infer from their input and the result. *SMC* is based on complex cryptographic techniques, which limit their scalability, in particular for very large number of parties. Ad-hoc *SMC* protocols have been proposed but these are not generic and do not allow all types of computations.

**Fully Homomorphic Encryption (*FHE*)** [4] schemes allow one to compute arbitrary functions over encrypted data without a decryption key. Given  $c_1$  and  $c_2$  ciphertexts of  $m_1$  and  $m_2$ , one can compute  $f(m_1, m_2)$  by decrypting  $f(c_1, c_2)$ . *FHE* is not suitable in our context due to its high cost and because users need to have the same encryption key, which creates serious security issues.

**Schemes based on gossip protocols** [8] Gossip protocols are based on a communication technique that works the same way a rumor spreads. On every round each node randomly selects a node from its neighborhood to exchange information, after a sufficient number of rounds, the result of each node converges to the final result. Gossip protocols scale well but are not generic in terms of possible computations, which make them unsuitable for our case.

## OUR APPROACH

We propose a protocol for secure distributed computations based on *Trusted Execution Environments (TEE)*. A *TEE* is a secure part of the processor, which guarantees confidentiality by providing isolated enclaves and integrity provided by attestation mechanisms<sup>1</sup>. Code executed inside a *TEE* is protected against all elements outside it, including the operating system. Several vendors propose their own Trusted Execution Environment : *Platform Security Processor* for *AMD* , *TrustZone* for *ARM* and *Software Guard Extensions* for *Intel*, the latter one will be used for the implementation and evaluation of our protocol. To be usable in our protocol a *TEE* must be able to do *Attested Computation*. Our architecture is composed of a querier and several users who agree to contribute to the computation. They are all equipped with a *TEE*. The use of *TEE* increases the cost of an attack while distributing the computations over plenty of nodes decreases the benefits of attacking a single computation node.

**Threat model.** The threat model in our context is different from *SMC* one. In *SMC*, the adversary is honest-but-curious. But in our context, data involved in the computation is stored and managed inside an isolated enclave. This makes the computation node honest even in the presence of curious or malicious *PDMS* owner.

<sup>1</sup>"Attestation is the process of demonstrating that a piece of software has been properly instantiated on the platform. In Intel SGX it is the mechanism by which another party can gain confidence that the correct software is securely running within an enclave on an enabled platform" [1]

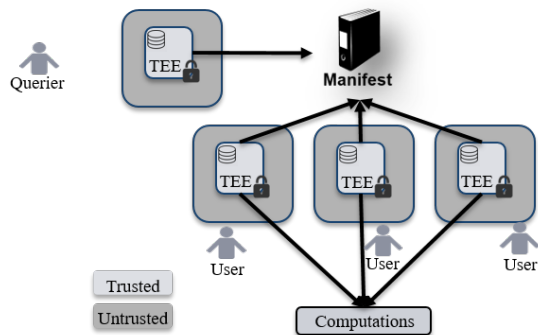


Figure 1: overview of the architecture

The first step in our study was to identify and propose a well-adapted threat model. We consider two different ones (1) honest computation node held by honest-but-curious participants and/or querier who may try to infer some information without breaking the security of the *TEE* and (2) malicious participants and/or querier who may perform a side channel attack (e.g., cache attack using technique presented in [2]) to retrieve some information from the *TEE*. They may also either participate to the computation with many *TEEs* to infer more information than they should or try to run another code than the expected one.

**Problem statement.** Based on the previous threat model and taking advantage of the confidentiality and the integrity guarantees provided by *TEEs*, we propose adopting a manifest based protocol. The manifest will be used to specify how the distributed computation will unfold. The protocol should ensure that the computation is respected as described in the manifest. This is not trivial, for the following reasons: (1) Each user should have the possibility to check the global correctness of the computation from her local view (*local correctness checking basis*). (2) The execution should be *zero knowledge*. In other words, in the case of data oriented operations where the dataflow depends on the data value (e.g., hashing or sorting as part of joins), an attacker able to observe the data exchanges should not be able to infer users' personal data. (3) Finally, the protocol should be *massive leak resilient*. More specifically, in case of a successful side channel attack, a user or a set of colluding users, should not be able to compromise a large amount of data nor targeting a specific user. Such attacks should be taken into account by the protocol by adding again a set of countermeasures to reduce the benefit-to-cost ratio.

## THE PROTOCOL

Let us take as a simple example the case of a group by computation. A naive secure execution approach based on *TEEs* could consist in participants sending their personal data to a single node which perform the computation. This approach is not massive leak resilient nor can it enforce a local correctness checking basis. Another approach could be participants sending their data to different nodes, where a node is in charge of computing data belonging to the same group, this obviously is not a zero knowledge execution. Our approach aims at designing a protocol that supports execution of any distributed computation, on a potentially large number of participants, while providing strong security guarantees. The protocol is organized in three phases (see Fig. 1). The first phase is to

set up the environment; the querier broadcasts the function to be computed, together with the organization of the distributed computation. Users who want to participate to the computation register with the querier. When the number of participants is sufficient, the querier and participants collaborate to produce an execution table, which is a list of tasks to do for each participant (e.g.  $user_1$  executes  $f_1$  on  $user_2$  data and sends her output to  $user_3$ ). The list of participants, the topology and the execution table form a manifest. This manifest has to be public, so that everyone can verify it. Thanks to attestation, the manifest is generated with strong guarantees of integrity. In the second phase, participants establish links between them following the manifest. Users leverage attestation to prove that they are executing their assigned tasks from the manifest. Finally, the last phase is the actual computation and it depends of the algorithm implemented. This protocol provides strong integrity guarantees, as breaking integrity means compromising a tamper resistant *TEE*. Moreover, if a malicious participant performs a side channel attack, retrieving the content of its *TEE*, it only has access to a limited amount of data as prescribed by the manifest. As long as the distributed computation is organized in a sensible way, no party should have access to large amount of personal data, and the threat of large scale attacks is this mitigated. Additionally, note that all computations are done on clear data, the results produced are thus exact. Moreover, only simple cryptographic operations and bookkeeping inside *TEE* are required in addition to the actual computation. This allows for a limited overhead w.r.t. performing the computation in the clear between mutually trusted parties.

## FURTHER WORK

As further work, we aim at designing a protocol solving the three problems defined above. In other words, a protocol giving guarantees on the global integrity of the computation while taking into account side channel attacks against *TEEs*<sup>2</sup>. Moreover, the protocol should be validated for three different kinds of computation, computation with static built of the manifest, those where the manifest has to be built dynamically (group by query based on hash) and iterative computation (such as training a shared neural network).

## REFERENCES

- [1] Ittai Anatiet al. 2013. Innovative technology for CPU based attestation and sealing. In *HASP*.
- [2] Johannes Götzfriedet al. 2017. Cache attacks on Intel SGX. In *Proceedings of the 10th European Workshop on Systems Security*. ACM.
- [3] Andrés Molina-Markham et al. 2010. Private memoirs of a smart meter. In *Embedded sensing systems for energy-efficiency in building*.
- [4] Craig Gentry et al. 2009. *A fully homomorphic encryption scheme*. Stanford University Stanford.
- [5] Joseph A Cruz et al. 2006. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics* (2006).
- [6] Jo Van Bulck et al. 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *27th USENIX Security Symposium*.
- [7] Ronald Cramer et al. 2015. *Secure multiparty computation*. Cambridge University Press.
- [8] Tristan Allard et al. 2016. Lightweight privacy-preserving averaging for the internet of things. In *M4IoT*.
- [9] Yisheng Lv et al. 2015. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* (2015).

<sup>2</sup>In [6] the authors present an attack against SGX which does not rely on software vulnerability and show that they can extract full cryptographic keys from enclaves and break the confidentiality and the integrity guarantees for remote computation. An expected microcode patches from intel may give solution against this attack.