DESIRE: Leveraging the best of centralized and decentralized contact tracing systems

Antoine Boutet, Claude Castelluccia, Mathieu Cunche, Cédric Lauradoux, Vincent Roca, Adrien Baud, Pierre-Guillaume Raverdy

DESIRE: Leveraging the best of centralized and decentralized contact tracing systems

Antoine Boutet
Univ Lyon, INSA Lyon, Inria, CITI,
Lyon, France
antoine.boutet@insa-lyon.fr

Claude Castelluccia
claude.castelluccia@inria.fr
Inria, France

Mathieu Cunche
Univ Lyon, INSA Lyon, Inria, CITI,
Lyon, France
mathieu.cunche@insa-lyon.fr

Cédric Lauradoux
cedric.lauradoux@inria.fr
Inria, France

Vincent Roca
vincent.roca@inria.fr
Inria, France

Adrien Baud
adrien.baud@inria.fr
Inria, France

Pierre-Guillaume Raverdy
pierre-guillaume.raverdy@inria.fr
Inria, France

ABSTRACT

Contact tracing in case of pandemic is becoming an essential mitigation tool for national health services to break infection chains and prevent the virus from spreading further. To support manual tracing, several countries have been developing contact tracing apps that detect nearby mobile phones using Bluetooth. Such data collection raised privacy concerns and several privacy-preserving protocols have been proposed to prevent the leakage of personal and sensitive information. These solutions are mainly divided into two categories using a centralized or a decentralized exposure score computation. However, both approaches depict limitations.

This paper presents Desire, a novel exposure notification system which leverages the best of centralized and decentralized systems. As opposed to existing contact tracing schemes, Desire leverages Private Encounter Tokens (Pets) generated locally on the device that uniquely identify an encounter between two nodes while being private and unlinkable by the server. The role of the server is merely to match PETs generated by diagnosed users with the pets provided by requesting users. Our privacy risk analysis shows that Desire drastically improves privacy against malicious users (i.e., limitation of decentralized systems) and authority (i.e., limitation of centralised systems). We implemented Desire, evaluated it in real condition, and show it feasibility.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**; **Privacy protections**; • **Applied computing → Health care information systems**.

1 INTRODUCTION

Since the beginning of the COVID-19 pandemic, the number of infections is rising and countries have been facing a second wave. Successfully containing this pandemic closely depends on the ability to quickly and reliably identify those who have been in close proximity to a contagious individual. In this context, reliable and efficient contact tracing is becoming essential in the fight against the spread of the virus. To support manual tracing in breaking infection chains and preventing the virus from spreading further, many countries have been deploying digital contact tracing applications.

To achieve its goal, contact tracing applications perform the recording of information about individuals near each other at a certain moment of time. Such data collection from an mobile application and potentially shared with some authorities raised significant privacy concerns [22, 39]. In addition, re-identification of individuals who are declared Covid-positive is highly sensitive and can lead to discrimination and harassment [5, 24]. Consequently, most of the contact tracing systems have considered privacy by design.

Different approaches have been applied to deploy privacy-preserving contact tracing. Although in most cases, short range exchanges of Bluetooth messages are leveraged to detect proximity of nearby individuals carrying a device, contact tracing systems are mainly divided into two categories using a centralized or a decentralized exposure score computation. Indeed, where the exposure score computation is performed has an important impact on the underlying scheme and privacy properties. In a decentralized scheme [45], declared Covid-positives notify their own ephemeral identifier to the system which are forwarded to all nodes to compute the exposure score locally on each device. On the opposite, in a centralized scheme [19], declared Covid-positives notify to the system the ephemeral identifiers of encountered devices that become at risk while each node periodically requests the server to know if they are at risk. Each solution comes with its pros and cons. No consensus has been found in the privacy community [14] and privacy risks have been identified for both schemes [13]. Supporters of the decentralized solutions underline the robustness against a state-level adversary and the advantage of data minimization by only notifying information from Covid-positive individuals. For instance, it is the choice adopted by Google and Apple and rapidly deployed on their mobile operating systems. Supporter of centralized solutions, in turn, stress the requirement to avoid discrimination risk of infected

Authors' addresses: Antoine Boutet, Univ Lyon, INSA Lyon, Inria, CITI, Lyon, France, antoine.boutet@insa-lyon.fr; Claude Castelluccia, claude.castelluccia@inria.fr, Inria, France; Mathieu Cunche, Univ Lyon, INSA Lyon, Inria, CITI, Lyon, France, mathieu.cunche@insa-lyon.fr; Cédric Lauradoux, cedric.lauradoux@inria.fr, Inria, France; Vincent Roca, vincent.roca@inria.fr, Inria, France; Adrien Baud, adrien.baud@inria.fr, Inria, France; Pierre-Guillaume Raverdy, pierre-guillaume.raverdy@inria.fr, Inria, France.

people, and the advantage to centrally control and adjusting the exposure score computation.

In this paper, we present Desire, a novel exposure notification system which leverages the best of centralized and decentralized systems in order to reduce the privacy risks of contact tracing applications. As opposed to existing proximity tracing schemes, we propose an exposure notification system that is based on Private Encounter Tokens (Pets). A Pet token uniquely identifies an encounter between two nodes and is private and unlinkable from the server. This Pet is locally generated by the mobile device from the Ephemeral Bluetooth Identifiers (Ebid) of both devices. The role of the server is merely to match Pets generated by diagnosed users with the Pets provided by requesting users. Furthermore, the server stores minimal pseudonymous data. All data stored on the server is encrypted using keys that are stored on the mobile devices, protecting against data breach on the server. Finally, we rely on a Trust Execution Environment (TEE) to pre-process declaration of infection and to avoid inference leakage (e.g., social graph).

The concept of Pet also enables several types of architectural variants between the centralized and the decentralized design. This unique flexibility enables each country to make a sovereign choice, choosing the deployment design that best fits its own requirements and trade-offs (e.g., epidemiological or data protection considerations, benefit-risk balance).

We analyse the privacy risk of Desire compared to other approaches. Desire drastically improves the privacy of the scheme against malicious users (i.e., limitation of decentralized systems) and authority (i.e., limitation of centralised systems). We also implemented a proof of concept of Desire and evaluated it in real condition. We show that the energy consumption of Desire is similar to other baseline approaches. Moreover, we also evaluated the scalability of the trusted proxy facing a growing number of infected node declarations. Finally, we evaluated the efficiency of the exposure score risk computed in Desire. The source code of Desire as well as the proof of concept of the mobile application are publicly available [1].

Aware that the design of the solutions that have been adopted in different countries was carried out in a hurry and suffer from limitations, we hope that this proposal and the open release of the associated software will stimulate the conception of interoperable applications on a larger scale in case of future pandemic.

This paper is organized as follows. Section 2 introduces background and related work. Section 3 presents the key concepts of Desire while Section 4 details the solution. Section 5 then contains the risk analysis of Desire and Section 6 reports the evaluation of our proof of concept. Lastly discussion and conclusion are presented in Section 7.

2 BACKGROUND AND RELATED WORK

Since the beginning of the Covid pandemic, contact tracing apps have led many debates and generated an important literature [7]. While the first tracing apps (predominantly deployed in Asia) extensively collected sensitive user information (e.g., name, address, mobile phone numbers, location), most further tracing apps consider privacy and security as a primary concern. Indeed, location data for instance are well known to be a very rich contextual information which can lead to many inferences (e.g., home and workplaces, race and gender, social network) and re-identification [15, 25, 50]. Although this rich information could be leveraged to better detect clusters and conducting multiple analysis, exploiting the location of users has been further restricted to limit the privacy threats [4], it is also one factor that limits the users' adoption [33, 40].

The data collection and the architecture of contact tracing apps has been a matter of much discussion due to both security and privacy concerns. We review in this section the centralised, the decentralised, and the hybrid systems that combine features from both the centralised and the decentralised approaches.

Decentralized approaches (e.g., DP3T [45], PACT [20]) are designed to avoid any control of a central server who could maliciously collect data and infer sensitive information by all means. While presented as keeping the privacy of individuals safe, several reports and applications have shown the limit of a such solution [2, 31, 34, 44, 47]. By relying on end nodes to compute score risk, the ephemeral ID of infected users are push to all devices in the system opening the risk to malicious users to infer sensitive information leading to an risk of discrimination and harassment for users declared Covid-positive [13].

Centralized approaches (e.g., Robert [19], BlueTrace [9], COVIDSafe [36], first version of NHS COVID-19 app [35]) are designed to avoid malicious users to infer sensitive information on other users, infected or not. By relying on a server, centralized solutions put an important trust in the server regarding both its correct and non adversarial behavior, and the protection of the data against data leakage. In case of a server more adversarial than honest but curious, although less likely than facing to tech savvy users, the server could infer sensitive inference for users whether they are declared positive or not, including part of the social graph [42].

No consensus has been found between supporters of both sides (i.e., centralized and decentralized). In this turmoil of discussion on the best approach to adopt, Google and Apple jointly agreed on a decentralized system for contact tracing interface called Exposure Notification API (GAEN) [1] that they rapidly integrated into their mobile operating systems. While many countries adopted this API to implement their solution above this API, these contact tracing applications suffer from the same limitation as DP3T on which the GAEN solution is based [3, 30]. Moreover, the underlying mechanisms of this API are not transparent and the integration of this API in the google and apple ecosystem lead to additional sensitive data collection [6, 27]. Finally, other reports have analysed the detection rules that trigger exposure notifications in the GAEN solution in complex environment (e.g., bus, tram) and show questionable correlation between exposure and real proximity [32]. But perhaps worse, Google and Apple have not lifted restrictions on the use of Bluetooth on their operating system to allow each country to develop their own solution, imposing, de facto, their solution [14].

As both centralized and decentralized solutions depict limitations, a third way has been explored [48]. These solutions try to combine advantages on both sides. For instance, Epione [43] introduces a new private set intersection cardinality or PSI-CA. This PSI-CA is used to learn the size of the intersection between the set of tokens held by a user (i.e., the application) and the set of tokens stored on a server, without the user revealing their tokens. However,

[1]https://gitlab.inria.fr/aboutet1/desire-poc

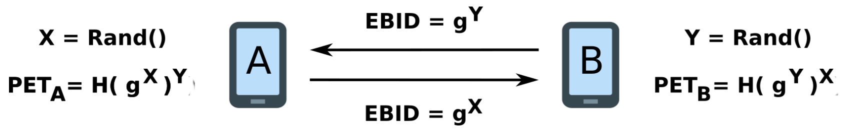

Figure 1: PET generation: device A broadcasts EBID $= g^X$. Device B broadcasts EBID $= g^Y$. Both devices compute PET $= H(g^{X.Y})$.

this solution introduces a communication overhead which could be a limitation given that a large number of users are expected in this kind of application. ConTra Corona [11], in turn, follows a decentralized approach but to better protect infected people, this solution leverages another pseudorandom identifier for the publication to all devices, instead of their broadcasted ephemeral ID. The linkability between this pseudorandom identifier and the ephemeral ID is only known by a non-colluding dedicated server.

3 DESIRE: THE BASICS

Desire aims at combining the best of centralised and the decentralised contact tracing approaches. To achieve that, DESIRE leverages the concept of Private Encounter Token (PET) that uniquely identifies an encounter between two nodes. This concept of PET also enables a third way between the centralized and decentralized categories, as the underlying protocol easily enables several types of architectural variants. This unique flexibility enables each country to make a sovereign choice, choosing the deployment design that best fits its own requirements without compromising interoperability across different national choices. Roaming across countries that potentially made different choices is therefore facilitated.

In this section, we introduce the concept of PET (Section 3.1) and the different deployment designs made possible by DESIRE (Section 3.2).

3.1 Private Encounter Tokens (PETS)

The notion of Private Encounter Token, or PET, is central to DESIRE. A PET token uniquely identifies an encounter between two nodes. This token is generated and computed locally by both nodes from their ephemeral identifiers generated at the current epoch named Ephemeral Bluetooth IDentifier (EBID), and is private and unlinkable from the server or other devices. A PET token can be generated from any *Non-Interactive Key Exchange* protocol [28]. As described in Section 4.6.1, the creation of PET tokens in DESIRE is based on the Diffie-Hellman key exchange protocol [26]. Figure 1 illustrates the generation of such PET tokens.

Device A generates and broadcasts a new EBID defined as g^X, where X is a secret generated by A. Upon the reception of an EBID, g^Y, from a device B, device A computes $(g^X)^Y$ (which is equal to $g^{X \cdot Y}$) and stores it in a local list the PET token $H(g^{X \cdot Y})$ [2], where $H()$ a cryptographic hash function such as SHA-256. Similarly, device B computes $(g^Y)^X$ (which is also equal to $g^{Y \cdot X}$) and stores it in a local list the PET token $H(g^{Y \cdot X})$.

The PET $H(g^{X \cdot Y})$ is a unique and a secret identifier for the encounter between A and B. Furthermore, the decisional Diffie–Hellman assumption guarantees that it can only be computed by A and B, and is therefore protected from eavesdroppers.

In case of infection declaration, instead of sharing all its previous EBIDS (i.e., like in decentralized approaches) or EBIDS of all devices that have been in proximity (i.e., like in centralized approaches) to inform at risk users, DESIRE only shares the associated PET tokens. PET has the advantage over EBID to reduce the attack surface and to improve privacy. Specifically, it reduces the ability of the server to link collocation information coming from different individuals. Furthermore, this method mitigates "replay attack", where a malicious individual collects the EBIDS received by an infected (or potentially infected) individual and replays them in many locations, in order to create a large number of false positives.

3.2 A Unique Flexibility and three Deployment Designs

The flexibility of DESIRE leads to three potential deployment designs.

- **(D1)** DESIRE centralized risk evaluation approach: here the central back-end server managed by the authority is in position to assess the exposure risk of a user, which means that her application needs to periodically query the server in order to know her risk.
- **(D2)** DESIRE state-less approach: in this design, the central back-end server does not maintain any state per application instance and is only used as a matching system between the encounter PET tokens uploaded by an application and a list of exposed PET tokens it maintains.
- **(D3)** DESIRE decentralized approach: here, the central back-end server is only used as a reflector in order to share widely the list of exposed PET tokens, and where the risk evaluation is performed within the device of the user.

Having three design flavors enables each country to choose the approach that best fits its requirements, in a sovereign manner. This choice depends on multiple criteria. For instance, this decision can be based on *epidemiological considerations* since it is usually recognized that an architecture with a central risk evaluation is recommended from an epidemiological point of view. This choice can also be based on *the adversary model*, since local considerations can lead to different risk assessments (each country can weigh the risk and benefits associated with each design option, with respect to the central authority, the users, and OS manufacturer). Naturally, this choice can also be based on *data protection considerations*. The DESIRE approach enables these choices to be made locally, on a per-country basis, without negatively impacting the service interoperability across different countries, even if they use different service designs.

[2]As we will see later, each node actually maintains two lists to prevent linkability between the PETS that are used to query the server and the PETS that are uploaded if the node is diagnosed positive.

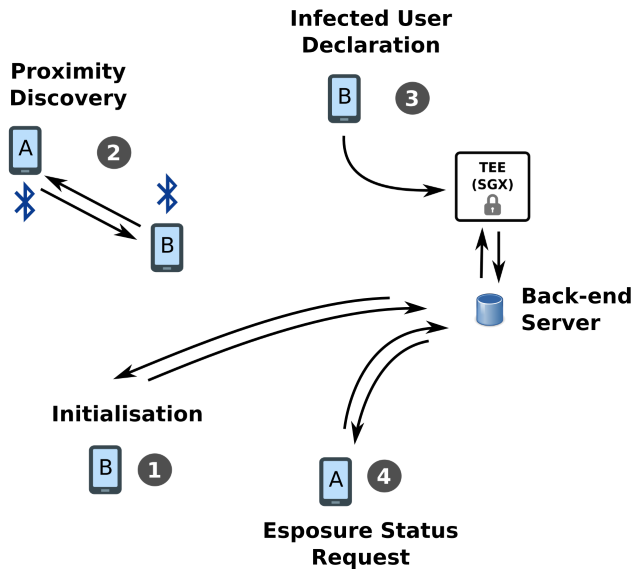

Figure 2: Overview: DESIRE follows four different phases.

4 DESIRE: A THIRD WAY FOR EUROPEAN RISK EXPOSURE NOTIFICATIONS

This section describes DESIRE in detail, focusing more particularly on the centralized design (D1). We first provide an overview of the solution (Section 4.1) before presenting each phase of DESIRE, namely the initialization phase (Section 4.2), the encounter discovery (Section 4.3), the infected user declaration (Section 4.4), and the exposure status request (Section 4.5). We then give implementation details of our publicly available proof of concept (Section 4.6). Finally, we present how to operate DESIRE as a state-less system where the server do not store information about registered users (Section 4.7) and as a fully decentralized system where the exposure score risk is computed locally on the device (Section 4.8). Lastly, we discuss interoperability considerations (Section 4.9).

4.1 Overview

In its centralized design (D1), the highlights of DESIRE are the following:

- Mobile devices generate their own identifiers, their private encounter tokens (PETS), and keep full control over them. These PET tokens are privately generated and unlinkable. This makes the scheme less vulnerable to a malicious server compared to a scheme where the pseudo-identifiers are generated by the back-end server.
- The server performs the matching between the PET tokens of infected and requesting users without having access to their actual identifiers. Relying on a central server for the matching task improves the robustness and resiliency of the scheme against malicious users, compared to a scheme where the matching is performed by the devices themselves.
- The server computes the risk score which improves the efficiency of the system as the score function can be instantly adapted by the health authority according to the evolution of the pandemics. Furthermore, this allows the server to control the number of notifications that are sent out every day and prevents uncontrollable situations where a high number of users get notified at once. Finally, this also improves the security of the scheme against malicious users since users do not get access to the proximity information but only to the resulting risk scores.

The system is composed of users who install on their devices an exposure notification application using DESIRE, and a back-end server under the control of the health authority (Figure 2). DESIRE features four different phases:

1. **Initialization:** When a user wants to use the service, she installs the exposure notification application, APP, from an official and trusted App store. APP then registers to the server that generates a permanent identifier. The server keeps an entry for each registered ID.
2. **Encounter Discovery:** After the registration, each device periodically generates a Ephemeral Bluetooth IDentifier, EBID, and broadcasts it regularly over Bluetooth. Each device also collects EBIDS of encountered devices and generates and stores PET tokens from them. These PET tokens identify an encounter.
3. **Infected User Declaration:** When an individual is tested and *diagnosed* Covid-positive, and after an explicit user consent and authorisation (from the medical services), her smartphone's application uploads its local list of generated PET tokens to the authority server, that adds them in a global list of exposed PET tokens. To avoid social graph inference, these declarations are sent through a trusted proxy leveraging a trusted execution environment (i.e., Intel SGX) to mix PETS from several different declarations.
4. **Exposure Status Request:** To update the exposure status of the user, the APP probes regularly the server with its list of generated PET tokens. The server then checks whether

the tokens appearing in the request are declared at risk and computes the risk score in that case according to the duration and signal strength of the exposition.

We consider that all exchanges between devices and the server are secured and protected through TLS. We also consider loose time synchronization between devices (thanks to NTP or any other time synchronisation mechanism). Time is discretized into epochs of 15 minutes. This value of 15 minutes is the rotation period of the random address recommended in the Bluetooth v5.1 specification [41], Vol 3, Part C, App. A]. We assume that these epochs and rotation periods are synchronized [8, 23]. Finally, we assume a symmetric bluetooth communication channel (i.e., two nearby devices detect each other).

4.2 Initialization

A user U_A who wants to install the application on his device must download it from a trusted App store. After installing the application $\textsc{App}_A$, U_A needs to register to the back-end server. This registration phase is composed of two steps: the *authorization token generation*, during which the user obtains an anonymous authorization token, and the *user registration* where the user registers to the server anonymously. These steps are unlinkable and provide anonymity to users.

(1) *Authorization Token Generation*: In this step, the user obtains from the server an anonymous authorization token that she uses in the User Registration phase. This could be performed by using blind signatures [18] or direct anonymous attestation [17] [3].
(2) *User Registration*: Once the user, U_A, obtains an authorization token, AT_A, she can use it to register to the server. More precisely, U_A sends a registration message which includes his authorization token, AT_A. The server then verifies the authorization token, creates a unique identifier ID_A and an entry in its IDTable. The entry table contains for each registered user, the following information:
   - **ID_A ("Permanent IDentifier for A"):** an identifier that is unique for each registered App, and generated randomly (random draw process without replacement to avoid collision).
   - **UN_A ("User A Notified"):** a flag indicating if the associated user has already been notified to be at risk of exposure ("true") or not ("false"). It is initialized with the value "false".
   - **ERS_A ("Exposure Risk Score"):** Current U_A's exposure risk score.
(3) The server generates an encryption key EK_A and sends ID_A, EK_A, and the Public key of the SGX enclave PK_{sgx} to the user U_A. It then uses EK_A to encrypt all elements of $\mathrm{IDTable}[\mathrm{ID}_A]$, except for ID_A, and finally deletes EK_A.
   Note that the server also stores all the AT tokens it has received to verify that they are only used once[4].

4.3 Encounter Discovery

Devices leverage Bluetooth Low Energy advertising mechanism to discover nearby devices and exchange Ebids as illustrated on Figure 3. A Device A randomly generates an ephemeral bluetooth identifier, $\textsc{Ebid}_{A,i}$, at each epoch i and broadcasts it over Bluetooth. This Ebids is defined as g^{X_i}, where X_i is a secret generated by A at epoch i. We exploit Elliptic Curve Cryptography and more specifically the elliptic curve Curve25519 [10] to generate g^{X_i} and we assume that the devices share the same group structure (order p and generator g), which are pre-configured in the application (see Section 4.6 for implementation details).

Upon the reception of an *EBID*, g^{Y_i}, from a device B, device A computes $(g^{Y_i})^{X_i} = g^{X_i Y_i}$. If the duration of the encounter $t(A, B)$ is longer than a certain threshold (i.e., to avoid generating and storing Pet tokens associated to encounter that are very brief), A then computes a pair of unlinkable Pet tokens:

$$\textsc{Pet}^1 = H(\text{"1"} \mid g^{X_i \cdot Y_i})\, ; \, \textsc{Pet}^2 = H(\text{"2"} \mid g^{X_i \cdot Y_i}),$$

where H() a cryptographic hash function such as SHA-256. These Pet tokens are stored in two different tables: the *Exposure Token List*, $\textsc{Etl}_A$, and the *Request Token List*, $\textsc{Rtl}_A$. Specifically, if the bit-string g^{X_i} is greater than bit-string g^{Y_i}, A stores $\textsc{Pet}^1$ in $\textsc{Rtl}_A$ and the $\textsc{Pet}^2$ in $\textsc{Etl}_A$, otherwise A stores $\textsc{Pet}^2$ in $\textsc{Rtl}_A$ and $\textsc{Pet}^1$ in $\textsc{Etl}_A$.

Similarly, device B computes $(g^{X_i})^{Y_i}$ (which is also equal to $g^{X_i Y_i}$) and the two Pet tokens $\textsc{Pet}^1$ and $\textsc{Pet}^2$. However, these Pets are stored in $\textsc{Etl}_B$ and $\textsc{Rtl}_B$ in the reverse order, $\textsc{Pet}^1$ is stored in $\textsc{Etl}_B$ and $\textsc{Pet}^2$ in $\textsc{Rtl}_B$. Entries in these lists are purged after a period corresponding to the contagious period (i.e., typically 14 days). Finally, node A (respectively B) deletes g^{Y_i} (respectively g^{X_i}).

As described Section 4.4 and Section4.5, the Pet tokens in Rtl are used to query the server for exposure status and the Pet tokens in Etl are uploaded to the server if the user is diagnosed-positive. Using two unlinkable Pet tokens for the same encounter prevents the server from linking the tokens used in exposure status requests with the tokens upload to the server in case of diagnosed positive. Without this protection, the server could use these links to reconnect together the tokens of its EList that belong to A and derive A's proximity graph.

Note that the two users part of the encounter generates the same couple of Pet tokens but stores them in its lists in the reverse order (Figure 3). Therefore the server cannot link the Pet token that user A and B use in their exposure status requests for their encounter, but can still perform the Pet matching if one of the users is diagnosed-positive and uploads her Etl list.

Note also that according to various situations, devices may generate several Pet tokens for a single continuous proximity encounter.

4.4 Infected Node Declaration

If user U_A is tested and diagnosed Covid-positive at a hospital or medical office, she is proposed to upload each element of her Exposure Token List $\textsc{Etl}_A$ list to the server. Note that this upload is anonymous, in particular it does not reveal the identity of U_A nor any of its Ebids to the server.

If insufficient precautions are taken, $\textsc{Etl}_A$ could potentially be used by the server to build the de-identified social/proximity graph

[3]We do not describe the generation of these Authorization Token in this document, interested readers can refer to [18].

[4]In practice the server can generate authorization tokens that are only valid for a certain number of days (by changing his Public/Private key pair regularly), which would reduce the storage load on the server.

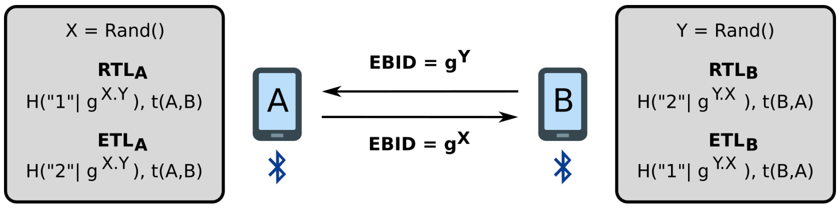

Figure 3: Device A broadcasts Ebid $= g^X$. Device B broadcasts Ebid $= g^Y$. Device A computes Pet^1 $= H(\text{"1"}|g^{X.Y})$ and Pet^2 $= H(\text{"2"}|g^{X.Y})$, stores Pet^1 in Rtl$_A$ and Pet^2 in Etl$_A$. Device B computes Pet^1 $= H(\text{"1"}|g^{Y.X})$ and Pet^2 $= H(\text{"2"}|g^{Y.X})$, stores Pet^1 in Etl$_B$ and Pet^2 in Rtl$_B$.

of the infected user. Indeed, if all the Pet tokens of the Etl are sent by batch to the server, it can link all Pets together through an anonymous node. The aggregation of many such social/proximity graphs may lead, under some conditions, to the de-anonymization of its nodes, which results in the social graphs of the users. It is therefore necessary to "break" the link between any two elements if Etl$_A$.

Therefore, the Etl$_A$ is uploaded on a trusted proxy equipped with an Intel SGX that mixes the elements of all infected users' Etl. This Trusted Execution Environment (TEE) ensures an execution of attested code in a trusted SGX enclave isolated from the rest of the system. The declaration of infection contains the Etl list of the user encrypted with the public key of the SGX enclave, PK$_{sgx}$. Consequently, only the SGX enclave is able to read the Etl lists sent by users.

To efficiently mix exposed Pet tokens, the SGX enclave waits for the declaration of infection from a certain number of users before mixing all Pets tokens contained in these declarations. To avoid side-channel attacks against SGX [16, 49], these Pets are shuffled using a memory oblivious permutation-based approach [37]. The trusted proxy then transmits the exposed Pet tokens via a specific API provided by the back-end server.

The server maintains a global list, EList, of all exposed $(token, t)$ tuples coming from all infected users. Thanks to the shuffle of the exposed Pet tokens coming from different users, the server is not able to reconstruct part of the social graph of the users.

4.5 Exposure Status Request

In order to check whether user U_A is at risk (i.e., if she has encountered infected and contagious users in the last CT days), the application App_A regularly sends an Exposure Status Requests (Esr_Req) to the server for ID$_A$. The server then computes a risk score value. The server replies with a Esr_Rep message that is set to "1" when the user is at risk or to "0" otherwise. More precisely, the following actions are performed:

(1) Node A periodically sends to the server an Esr_Req$_A$ message that contains ID$_A$, EK$_A$, the Pet tokens of Rtl$_A$.
(2) The server retrieves IDTable[ID$_A$], decrypts each of its elements with EK$_A$.
(3) The server verifies the UN$_A$ flag. If UN$_A$ = *true*, the server returns the same Esr_Rep$_A$ message set to "1" (at risk of exposure)[5]. The server then encrypts each element of IDTable[ID$_A$] with EK$_A$ and erases EK$_A$. Otherwise, it continues.
(4) The server then checks whether any of the Pet tokens of Rtl$_A$ appear in any $(token, t)$ tuples of EList. If yes, the server computes an exposure risk score for user U_A, stores[6] it in IDTable[ID$_A$] (field ERS$_A$).
(5) Two situations are then possible:
  (a) If the computed score indicates that the user is at risk of exposure, the server sets UN$_A$ at "*true*". An Esr_Rep$_A$ message set to "1" (at risk of exposure) is then returned to the user.
  (b) If the computed score does not indicate any significant risk, an Esr_Rep$_A$ message set to "0" is returned to the user.
(6) After sending the reply message, the server encrypts each element of IDTable[ID$_A$] with EK$_A$ and erases EK$_A$.

Note that, for a given user A, since the Pet tokens used in the uploaded list, Etl$_A$, and in the requests, Rtl$_A$ are different for the same encounters, the server can not link the elements and can not build any proximity graph. Furthermore, the only information that leaks out of a request is the number of elements, which could give some information about the number of encounters, of Rtl$_A$. One solution is to set the number of elements in the request to a fixed value, T. If the number of elements of Rtl$_A$, N_A is smaller than T, A pads its request with $(T - N_A)$ fake tokens. If N_A is larger than T, A only uses T elements of Rtl$_A$ for its requests. Other solutions could be to encode the elements of Rtl$_A$ into a Bloom or Cuckoo filter.

Note also that the number of daily requests performed by U_A can be limited.

When a node is notified at risk, its UN$_A$ flag is set to "true" in IDTable. From this point on, the server will process its Esr_Req queries as usual but will keep replying with a Esr_Rep message set to "1" regardless of the Esr_Req queries.

[5]Note that it is good practice for an application whose user is already notified "at risk" to keep on sending Esr_Req queries and receive Esr_Req messages to make this application traffic indistinguishable to any other application when observing the generated network traffic.
[6]Note that server could also store the Rtl tokens generated in the last CT in IDTable. That would save some bandwidth (since a node would only need to send its daily token in its Esr_Req queries). In addition, this would improve security on the phone that would only store the daily tokens locally.

When set "at risk", the notified node has several options:

- She is tested and diagnosed as Covid-positive: she can upload her ETL$_A$ list as described in Section 4.4.
- She is tested and diagnosed as Covid-negative: she can inform the server that her identifier ID$_A$ was tested negative via a specific protocol (not specified in this document)[7]. As a result, the UN$_A$ flag is reset to "false" on the server.
- She decides not to be tested or not to inform the server about the result of her test: in that case her "at risk" status will be reset automatically after a certain fixed period of time (3-4 days, value to be defined with epidemiologists).

In any case, an application that has been notified "at risk" continues to send and receive EBIDS and to compute PETS. This is required for instance when a user is waiting for a test result, if this latter turns out to be negative: encounters continue to be recorded and as soon as the user unlocks her status at the server, the updated exposure status can be computed without any gap in the history.

A user that was diagnosed positive should have the option to continue using the application as long as she is contagious. During this period, she must regularly upload her ETL list to the server.

4.6 Implementation

We implemented a proof of concept of DESIRE. This proof of concept including the mobile application and the server as well as monitoring tools is publicly available [8]. In this section, we give details about the implementation of both the bluetooth communications between devices (Section 4.6.1) and the computation of the exposure risk score (Section 4.6.2).

4.6.1 Bluetooth Communications. The creation of the PET tokens in DESIRE is based on the Diffie-Hellman key exchange protocol [26]. The EBIDS are accordingly defined as g^X (Section 4.3). For efficiency reasons, we implemented it using an instance of the discrete logarithm on elliptic curves (Curve25519)[9]. With Elliptic Curve Cryptography and the elliptic curve Curve25519, g^X requires 32 bytes to ensure protection against brute force attack. These EBIDS are thus larger than 16 bytes and can therefore not be carried by a Bluetooth advertisement packet alone. Consequently an EBID is split into three slices and transmitted in multiple consecutive Bluetooth advertisement packets.

We considered two Bluetooth advertisement schemes: a multi-advertising and a rotating advertising mode. In the multi-advertising mode, the device is able to broadcast multiple advertising payloads in parallel. In this mode, the different parts of the EBID are broadcasted through multiple advertising payloads that makes possible for the receivers to directly reconstruct the EBID. Although very suitable in our case and less energy consuming than the rotating mode, this feature is only available on Android 5+ and we cannot rely on it, thus currently not compatible with the greatest number of devices.

In the rotating advertising mode (also called carousel), only one advertising payload is broadcasted but the payload is changed periodically. Consequently, each slice of the EBID is advertised one after the other in a carousel fashion. To retrieve the EBID, a nearby device has to receive all the three slices. To increase the efficiency of this scheme, we introduce a fourth redundant slice which is the XOR sum of the three other slices in order to enable a device to reconstruct an EBID after receiving any subset of three slices out of the four that are transmitted.

Reliability of BLE communication is not ensured (i.e., the wireless channel can be subject to perturbations). To overcome potential packet loss due to the unreliable wireless channel, the advertising scheme of Desire addresses two kinds of perturbation. Firstly, each payload (in both the multi-advertising and the rotating mode) is advertised every second during 20 seconds. Advertising the same payload multiple times makes the reception of the payload robust to random perturbation on the wireless channel. Secondly, the introduction of a fourth slice to enable a device to reconstruct an EBID makes the reconstruction of the EBID robust to transient perturbations of the wireless channel.

Lastly, we implemented our own scan filter to identify the service UUID of our application from the BLE message as it was not well supported on several devices.

4.6.2 Exposure Risk Score. Bluetooth has not been designed to evaluate distance between two nodes. Consequently, the exposure risk score of Desire does not rely on a distance evaluation but we considered instead a probabilistic approach [29] already successfully used in TousAntiCovid, the contact tracing application developed by France. Specifically, this score is the probability to be closer than 2 meters during at least 15 minutes (i.e., an epoch). As described in the specifications, by aggregating information over different time windows inside an epoch, this exposure risk score accounts for packet loss by design.

To compute an estimator of the distance between two devices, we evaluate the power attenuation between the transmitter and the receiver. To measure an accurate power attenuation, correction gains have to be applied to take into account the antenna gain at the transmitter and at the receiver. These corrections depend on hardware (i.e., the BLE chipset and antenna) and the exploited driver. However, these correction gains are static. Consequently, without any protection, this information could be used by an adversary (i.e., the server or any user) to link together over time the different EBIDS of a same device.

To avoid this linkability, this information is never revealed in DESIRE. Instead, these information are obfuscated and their changes are synchronized with the rotation of the EBID at each epoch. Specifically, two devices exchanging their EBIDS will deduce a calibrated noise, CN, from the EBIDS (by selecting the most significant bits of the greater bit-string EBID). This shared noise between devices is then subtracting from the actual value of the corrected transmission gain (GTX) and the corrected reception gain (GRX). As the same value is removed from the transmission and the reception gain, the power attenuation used to estimate the distance remains unchanged.

The obfuscated transmission gain (GTX $-$ CN) of each PET is maintained in the ETL list and are sent to the server through the declaration of infection. For each discovering EBID, a device maintains a vector with all the received signal strength indications (RSSI) associated with all received advertisements. If the exposure with this

[7]Informing the backend server about the results of the Covid tests could be very important to define and calibrate the risk score functions.
[8]https://gitlab.inria.fr/aboutet1/desire-poc
[9]https://github.com/duerrfk/ecdh-curve25519-mobile

EBID is long enough (i.e., local filtering to only consider the relevant encounters), a PET is generated. In this case, all RSSI are corrected and obfuscated by subtracting CN from the values. All these values are then locally aggregated by time window as described below and sent to the server with the exposure status requests.

As described in [29], several operations have to be performed to get the exposure risk score. Some operations are linear transformations and can use the obfuscated reception gains and other operations cannot. Specifically, clipping, windowing and fading reduction can be performed locally on the device.

These operations lead to two vectors: the risk score (still obfuscated) and the number of received messages for each time window during the epoch (15 windows). These vectors are sent with the RTL to the server through the exposure status request. The server then checks if a PET of the ESR_REQ matches with a PET in the EList and retrieves the associated obfuscated transmission gain. If it is the case, the server is able to aggregate all risk scores by taking into account the transmission gain and return the exposure score risk to the user.

4.7 Towards a State-less DESIRE (D2-Design)

In the centralized design of DESIRE, the server stores some information about registered users. The information is minimal and is securely stored. We believe that this feature is a strength of our scheme since it mitigates some attacks (by controlling the number of registered users and limiting the request frequency). It also allows the health authority to compute and update the risk score according to the evolving situation, it provides information to the health authority about user exposures that could be very valuable to optimize the risk score function, as discussed in Section 5.

Having said that, it is possible to deploy DESIRE to operate as a state-less system where the server would be a mere "matching machine" between the PET tokens that are uploaded by infected users and the tokens that are contained in the ESR_REQ queries. The protocol would operate as follows:

- **Initialization:** Users do not need to register to the server. They however need to obtain CT different anonymous authorization tokens, AT, per day. Each of these authorization tokens should only be valid for a specific day and could be obtained, in batch, during registration[10].
- **Encounter Discovery**: Same as in the regular DESIRE protocol.
- **Infected Node Declaration**: Same as in the regular DESIRE protocol.
- **Exposure Status Request**: Users query the server with unlinkable ESR_REQ queries, where each of them contains a subset of different RTL_A's elements. Each ESR_REQ query can, for example, contain the PET tokens generated during the same day. In this case, on each day d_i, a user sends CT different and unlinkable ESR_REQ queries (one containing the tokens generated during d_i, one containing the tokens generated during d_{i-1}, …, one containing the tokens generated during d_{i-CT}).

  The server processes each of these ESR_REQ queries independently, by checking whether any tokens contained in the request appears in its list of exposed tokens, ELIST. It then computes the resulting exposure risk score of each request and sends the result back to the requested user.

Note that with this design, the server cannot compute the user's "global" exposure risk scores anymore but only unlinkable "daily" risk scores. The APP obtains CT different risk scores (for the different CT previous days) that it needs to aggregate into a global one. This also implies that all apps have to include an aggregation function that might need to be updated regularly.

4.8 Towards a fully decentralized DESIRE (D3-Design)

Lastly, it is also possible to fully decentralize the risk score assessment by publishing to all Apps the exposed PET tokens contained in ELIST of the server. In such a deployment, the exposure score risk is computed locally, within the user device. The protocol would operate as follows:

- **Initialization:** Users do not need to register to the server.
- **Encounter Discovery**: Same as in the regular DESIRE protocol.
- **Infected Node Declaration**: Same as in the regular DESIRE protocol.
- **Distribution of the global list of exposed PET tokens:** Contrarily to the design D1 and D2, here the exposure status is not computed on the back-end server. Instead, the list of exposed PET tokens (ELIST) is publicly available, in an open registry, and user devices periodically retrieve the new exposed PET tokens of this list [11]. The user device can thus check locally if any of them matches one of its own local PETS present in its RTL, and compute the associated exposure risk score by using a local risk-scoring algorithm.

In this design, DESIRE would then be very similar to other so-called "decentralized" schemes, such as DP3T [46]. However, the public registry does not identify any user device but encounters, and only the two devices that are part of a given encounter know the associated PET token. Consequently, thanks to the PETS, the decentralized design of DESIRE avoids any risk of stigmatization or discrimination of infected users unlike DP3T and Google-Apple (see Section 5).

4.9 Interoperability and Roaming

The interoperability between applications of different countries is an important aspect of contact tracing to efficiently break infection chains. One limitation of this interoperability is that each contact tracing application uses its own format to encode at-risk identifiers (e.g., ROBERT and DP3T use different EBIDs). Unfortunately, in this context, interoperability is often achieved by imposing the deployment of an unique solution over different countries to enable interoperability (e.g., GAEN) which imposes by the same way the design (and the associated privacy and risks trade-off) and reduces the sovereignty of each country. The DESIRE approach, with its

[10] A user would then obtain CT tokens per day, under CT different keys (and different every day: a token usable today for d_i, for d_{i-1}, for d_{i-2}, ... a token usable tomorrow for d_{i+1}, d_i, d_{i-1}, etc,). A specific signing key is used for each role of the token. This is in the same vein as done in on-line e-cash schemes 'a la Chaum' with blind signatures [21].

[11] The mechanism used for this distribution is out of the scope of this document.

different design flavors, enables each country to choose the design that best fits its requirements, in a sovereign manner, without negatively impacting the service interoperability across different countries. Indeed, the Pet tokens can be usable across services used in different countries, even if those services follow different design options.

Identifying the national server: For privacy purposes, a user should not learn the country of origin of encountered persons through its contact tracing application. Yet, interoperability requires forwarding PET tokens to different national back-end servers. This could be achieved either by letting the users record, manually or automatically, the visited countries (option 1), or by associating a country code to the Ebid in the broadcasted BLE message, in such a manner that privacy is not compromised (option 2).

Option 1 requires users, when they visit a different country, to continue contacting the server of the visited country during a certain time after she came back home. Similarly, in case the user is tested positive after coming back from abroad, her proximity history needs to be exchanged with the visited back-end server. On the opposite, option 2 facilitates (since information is associated to the Ebid and then the Pet tokens) and reduces the amount of traffic (since only encounters with a foreign application are forwarded to the foreign back-end server) across the various national back-end servers. Option 2 can be implemented by introducing an Encrypted Country Code as in [19] in the BLE message which carries the Ebid. During an encounter, the Country Code of the remote device is kept in the Rtl_A and Etl_A lists for the associated Pet token. This Country Code information is also uploaded to the server along with a Pet token during an infected node declaration. Consequently, when a national back-end server identifies an encounter with a foreign application, say App_B, this national server can easily forward the associated Pet token to the foreign back-end server to which App_B registered within the federation. [12]

Interoperability with foreign applications: The interoperability between App_A (national application) and App_B (foreign application), across different design options, must be considered along two dimensions: 1) when App_A is tested positive and wants to inform App_B, or 2) when App_A requests (design D1 and D2) or computes locally (design D3) its exposure risk, in case App_B has been tested Covid-positive.

The use of Desire has major benefits, because of two reasons. First of all, the interoperability between a national and foreign application, no matter their respective design options, is made possible by the single format of a Pet token: no matter the design considered, no matter the nationality of the two applications involved in an encounter, both of them will compute and store the same Pet tokens in their Etl an Rtl lists.

Secondly, interoperability is made possible because there is no difference about the nature of information uploaded in case the user is tested Covid-positive: whereas Robert uploads the Ebids of exposed users, whereas DP3T and GAEN uploads "diagnosed keys" of the user, with Desire, regardless of the design option, a Pet token that is uploaded during an infected node declaration is always a Pet token of an encounter at risk. If this Pet token involves a foreign remote application, once routed to the appropriate national server, this Pet token is added to the EList of that server. The remaining operations are then the same as before, in case of two national applications: this Pet token is either stored and used within that server (design D1 and D2) or made publicly available to all applications (design D3).

As a consequence, interoperability is maximum, each country can freely choose the design option that best fits its requirements in a sovereign manner, without any impact on interoperability and roaming.

5 PRIVACY RISK ANALYSIS

The adversary model commonly used in the literature for contact tracing is as follows:

- Users can be malicious. They can, for example, perform active and passive attacks, eavesdrop, inject bogus messages, modify messages, pollute users' contact lists and develop their own applications.
- The authority running the system can be malicious. Specifically, it can deploy spying devices or modify the protocols and the messages. It might use collected information for other purposes such as to re-identify users or to infer their contact graphs. We assume the back-end system can face data leaks.

While being faced with malicious users is quite universal, considering a malicious authority highly depends on the country where the system is deployed. More precisely, the question of malicious authority is questionable in a democratic country where citizens trust their institutions – warrant of a public debate before a government can deploy any large scale surveillance system – as well as an independent Data Protection Authority (DPA) in position to audit the system specifications and its operational deployment. This notion of trust (or lack of trust) towards the institutions and a DPA is one of the main criteria for selecting a deployment design (Section 3.2).

Based on the existing privacy analysis on digital contact tracing systems [13, 42], decentralized solutions such as DP3T mainly suffer from large risks of sensitive inferences targeting infected users, and the main limitation of centralized solutions such as Robert is the risk of inference of part of the social graph in case of an adversarial server (i.e., from law enforcement agencies or a state-level adversary) and the risk of data breach/leak. In this section, we review the privacy risks with respect to Desire. This analysis shows that Desire improves privacy compared to both centralized and decentralized solutions.

Server Data Breaches: The server only stores pseudonymous data. In addition, this information is minimized (e.g., rough data is immediately removed once processed) and only used to compute the exposure risk scores. Furthermore, each entry in IDTable of a device A is encrypted using a key EK_A that is stored only by user A and provided to the server within the Esr_Req queries. As a result, in case of a data breach of the server, all useful information will be encrypted with different keys (i.e., one per user). Note that the back-end server could be equipped with some secure hardware component (e.g., Hardware Security Module) to encrypt IDTable.

[12]This document does not discuss the management of a Country Code more in detail. Note that the Encrypted Country Code defined in the Robert protocol [19] could be easily introduced here.

The risk associated with a data breach is then minimal and only concerns the centralized design D1.

Passive Eavesdropping/Tracking (by malicious users and authority): Since PET tokens are unique per encounter and are computed locally, passive eavesdroppers only get the EBIDS, that are changing randomly at every epoch and only have a temporary use. As a result of the decisional Diffie–Hellman assumption, the eavesdroppers cannot relate any EBID to any PET token. Furthermore, an attacker deploying several Bluetooth passive scanners will not be able to relate any EBID to any PET token and therefore cannot track users. DESIRE is robust in front of passive eavesdropping/tracking.

Active Eavesdropping/Tracking (by malicious users): A malicious user who actively eavesdrops, by broadcasting its own EBIDS, can compute PETS with encounters he has with both Alice and Bob if they are close enough, but he cannot determine the PET of the encounter between Alice and Bob.

It is also interesting to note that with the implementation of the rotating advertising mode (Section 4.6.1), the adversary has to collect the three slices of the EBID to build the associated PET token. As each EBID slice is advertised during 20 seconds, the adversary has to be close to the target for at least 1 minute, which makes an active eavesdropping during a short amount of time ineffective (no PET can be computed).

Active Eavesdropping/Tracking (by malicious authority): If the authority is active and deploys Bluetooth devices that also broadcast their own EBIDS, containing for example g^{Z}, the target device, A, will generate and store the PET tokens $H("1", g^{A \cdot Z})$ and $H("2", g^{A \cdot Z})$. When receiving ESR_REQ messages from a user, the authority could rebuild a mobility trace by correlating the location associated with the received EBIDS. Indeed the ESR_REQ messages of a user can be linked as they include the ID of the requesting user. Such mobility traces could then be leveraged by the authority to re-identify the user[13]. This attack can be mitigated with a state-less DESIRE (Section 4.7).

Reconstructing social interaction graphs (by malicious authority): When a user A is diagnosed Covid-positive, she uploads all elements of her ETL_A through the SGX-based proxy. As users directly contact the front-end (without using anonymous communication), an adversary controlling the front-end can run traffic analysis to monitor the ip address of users. This risk is common to all systems that use direct upload. However, only the SGX enclave is able to decrypt the infected node declaration (i.e., the PET tokens at risk contained in the list ETL) sent by users and several declarations are mixed together before to be forwarded to the back-end. Consequently, the adversary is not able to make any link, neither between the user and the uploaded PET tokens, nor between the uploaded PET tokens.

When user A queries the server for her exposure status, the server is able to link ID_A to all PET tokens contained in the ESR_REQ query. However, the PET tokens used in the ESR_REQ queries are different from the tokens uploaded to the server if she ever gets diagnosed. Consequently, the server is not able to infer any links between exposed tokens in its ELIST and tokens in requests. Furthermore two users A and B who exchanged EBIDS and built associated PET tokens, request the server with different tokens (PET^1 one of them, PET^2 for the other). Thus the server is not able to link any tokens in different requests.

Infected Node Re-identification (by malicious user): In design D1 and D2, a malicious user can not identify infected contacts since this information is kept on the server, out of reach of users who only get an exposure risk score. In design D3, as a PET token does not identify a user device but an encounter, a malicious user could only identify contacts actually encountered. It is also noteworthy that the "one entry" attack[14], that is inherent to all schemes, is still possible. This attack could be mitigated by:

- requiring users to register in order to limit Sybil attacks;
- using a probabilistic notification scheme, as presented in [18];

Replay attacks: This attack is not possible since DESIRE assumes symmetrical communications. For example, if a malicious node, Eve, replays the EBID_B to A (assumed farther away), A will compute and store the corresponding PET^1 and PET^2. However, B will not be able to compute the same PET since she cannot listen to EBID_A.

Relay attacks: In this attack, Eve replays the EBID_B to A and vice-versa, so that both of A and B can compute the same PET^1 and PET^2 even if they are too far away to listen to each other without Eve. The attack is valid but only possible within, at most, one epoch. Going further is difficult as it may require either to have accurate timing measurements to detect the added delay while relaying the EBIDS, which is hard in practice (devices are not synchronised with a sufficient accuracy), or some geolocation information (to detect physical distance between devices), which is not desired.

False Alert Injection Attacks: The pollution is an attack where a malicious node colludes with a diagnosed user to include the identifiers of some victims in his contact list. The goal of the malicious adversary is to make the app of a target victim raise false alerts. This attack requires that the colluding user and the victim interact (to compute their PETS). Such attacks are therefore only possible via a relay attack, i.e., only within an epoch.

6 EVALUATION

We implemented and evaluated DESIRE in real condition. In this section, we show that DESIRE keeps a limited battery consumption, similar to other contact tracing approaches (Section 6.1). In addition, we show that the SGX-based proxy is able to quickly process and mix together a large number of declarations of infection (Section 6.2.2). Lastly, we evaluated the exposure risk score computed by DESIRE (Section 6.3).

6.1 Energy Consumption

Energy consumption of a mobile application, in particular a contact tracing application, is a crucial aspect as it will impact its adoption and usage [38]. Here we evaluate the energy consumption of DESIRE

[13]Although this attack is technically possible, a malicious authority probably has simpler ways to track users (such as cellular network surveillance or Wi-Fi tracking).

[14]In this attack, the adversary has only one entry, corresponding to U_T, in her local lists (this can easily be achieved by keeping the Bluetooth interface off, switching it on when the adversary is next her victim and then switching it off again). When the adversary is notified "at risk", she learns that U_T was diagnosed Covid-positive.

using the rotating advertising mode (Section 4.6.1) and SwissCovid (version 1.1.0), the contact tracing applications developed by the Swiss government and based on DP3T and the GAEN API. We also consider StopCovid (version 1.1.3), the contact tracing application developed by France based on the ROBERT protocol. Figure 4 depicts the energy consumption of these solutions running on a Samsung S10+ using only Bluetooth over 12 hours compared to a baseline which corresponds to the consumption of the device without running any application. Results show that DESIRE keeps a limited energy consumption by spending 6% of the battery consumption compared to the baseline without application. This consumption is similar to the SwissCovid and the StopCovid applications.

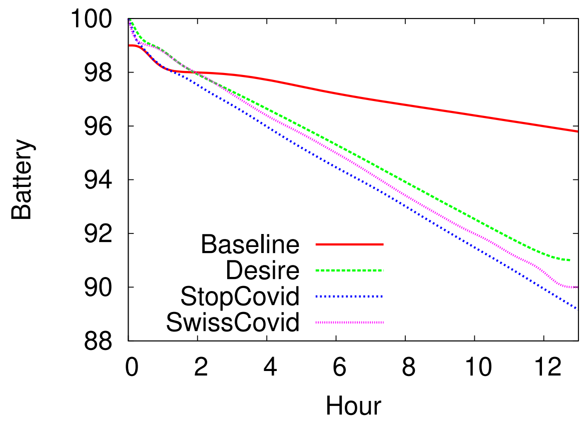

Figure 4: The energy consumption depicted by DESIRE is similar to the SwissCovid and the StopCovid applications.

Activating the WiFi on the smartphone, enables exchanges with the back-end server to request the infection status for both DESIRE and StopCovid, and to download diagnosed keys for Swisscovid. This additional communication increases the battery consumption to 9% compared to the baseline without application (versus 6% without the WiFi). The environment also impacts the energy consumption, a noisy environment (i.e., with multiple additional devices communicating over Bluetooth) consumes 45% more battery than a quiet environment (i.e., only three devices communicating together).

6.2 Scalability of DESIRE

The back-end server of DESIRE has to serve a large number of users and process the multiple declarations of infection. In the design D1 and D2, its scalability is determined by the capacity to answer the exposure request status. Contrastively, in the design D3, the back-end server has to distribute the exposed PET tokens to all devices. Lastly, for all designs, the scalability of the back-end server is also determined by the capacity to protect the infected node declaration.

6.2.1 Scalability of the Status Requests to the Back-end Server. In the design D1 and D2, each application periodically requests the back-end server to know the exposure status of the associated user. These requests include the PET tokens contained in the RTL as well as metadata (i.e., corrected reception power). Although a local filtering is performed on the device in order to only keep PETS related to long enough encounters, these requests can contain a large number of PETS and thus generate significant traffic to the server. If we consider requests with an average of 100 exposure PETS and the associated metadata (i.e., around 5.5 kB once gzipped), sent twice a day by 10 Million users, we arrive at 110GB of upload traffic per day for the server.

In addition, for each request, the back-end server checks whether any of the PET tokens of the request appear in any exposed token maintained in its ELIST. If it is the case, the server retrieves the transmission gain to compute the aggregated exposure score risk (Section 4.6.2). If we consider 100.000 daily new infections with an average of 100 exposure PET tokens per declaration and a contagious period of 14 days, the ELIST will manage 140 million PET tokens.

6.2.2 Scalability of the Protection of Infected Node Declarations. To avoid the server to infer information about the social graph of users (Section 4.4), the infected node declarations are sent to a trusted proxy which mixes the exposed PET tokens from different users. To ensure trusted execution, this proxy runs inside an Intel SGX enclave. We implemented such a proxy in order to evaluate its scalability when processing the stream of infected node declarations. Figure 5 depicts the time spent by the proxy to mix the exposed PET tokens for a growing number of infected node declarations. We considered 100 PET tokens per declaration and we considered two memory oblivious shuffling schemes: a sorting and a permutation-based approach with different complexities [12]. Results report roughly a similar runtime for both approaches with a linear time spent by the enclave to process the infected node declarations. According to the current statistics of the infected node declaration through contact tracing application (e.g., around 100,000 new daily infections in the US), the scalability of the proxy is not a limitation with a fast processing of PET tokens. Indeed, Figure 5 shows that 1,000 user declarations, each with 100 PET tokens, requires 4 seconds.

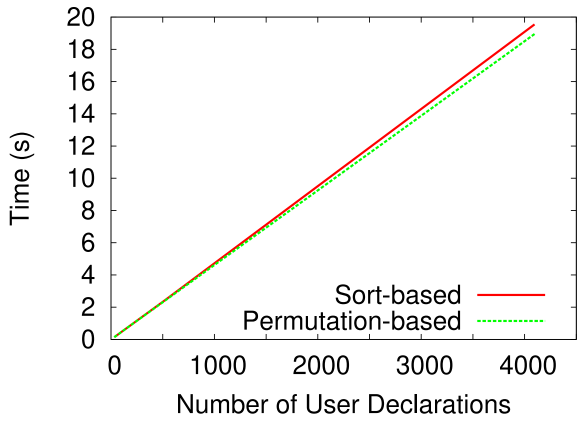

Figure 5: The trusted proxy is able to protect a growing number of infected node declarations in a linear runtime.

6.3 Exposure Score Risk

Finally, we evaluated the capacity of Desire to compute an exposure score risk. We consider two environments. First, a "quiet" environment where only three Samsung 10+ are 80 cm apart. Second, a "noisy" environment where four additional devices are placed two meters from the three Samsung 10+ that advertise BLE messages. A noisy environment with packet loss and variation of the RSSI can impact both the reconstruction of the EBID, and the score risk. In both cases, we monitored the BLE messages received by the devices and computed the exposure score risk from them including the local filtering and the computation on both the device and the server (Section 4.6.2). Our exposure score risk is computed per epoch (as the Ebid changes at every epoch) and an individual is declared at risk once its risk score reaches a certain threshold (contact threshold at 0.2 and risk threshold at 0.5 as defined in the specifications [29]) during an epoch. Figure 6 shows the evolution over the epochs of the exposure score risk of Desire. First, results show that the exposure risk score is computed after the first epoch. This means that even in a noisy environment, the EBIDs have been received and reconstructed correctly from the different slices. Second, the advertising scheme has also allowed the node to compute the risk score, and as expected, this score risk is above the threshold risk at every epoch for both a quiet and noisy environment.

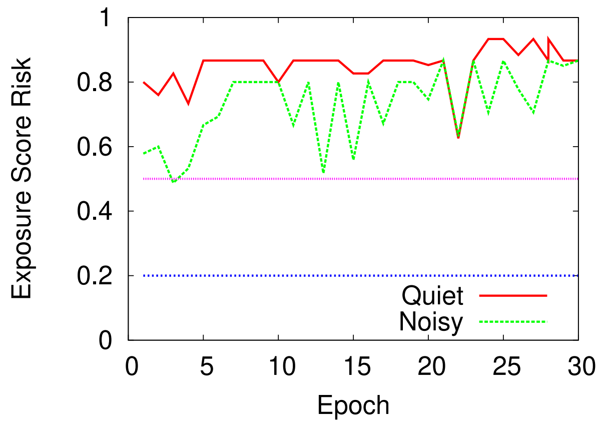

Figure 6: Devices are declared at risk with an exposure score risk above the risk threshold.

7 DISCUSSION AND CONCLUSION

Digital contact tracing is an essential tool to support manual tracing and for breaking infection chains and preventing the virus from spreading further. Two main approaches (i.e., centralized and decentralized) have been applied to deploy privacy-preserving contact tracing. However both approaches have their own advantages and limitations. To overcome these limitations, we propose Desire, a novel exposure notification system which leverages the best of centralized and decentralized systems. The key concept of Desire is the Pet token, a private and unique identifier that represents an encounter between two nodes. The concept of Pet enables a third way between the centralized and decentralized exposure notification systems, as the underlying protocol easily enables several types of architectural variants. This unique flexibility enables each country to make a sovereign choice, choosing the deployment design that best fits its own requirements without compromising interoperability across different national choices. We show that Desire drastically improves the privacy against malicious users (i.e., limitation of decentralized systems) and authority (i.e., limitation of centralised systems).

We implemented a proof of concept and showed its feasibility through evaluations in real condition. Specifically, we show that 1) the power consumption of Desire (relying on Ebids splinting into multiples advertisements) is similar to other contact tracing approaches, 2) the SGX-based proxy is scalable, and 3) the exposure risk score successfully detects proximity even in the presence of a noisy wireless channel. The source code as well as the associated software are publicly available.

REFERENCES

[1] Apple and Google. Privacy-Preserving Contact Tracing. https://www.apple.com/covid19/contacttracing, 2020.
[2] BLE contact tracing sniffer PoC. https://github.com/oseiskar/corona-sniffer, 2020.
[3] Corona Detective - Find out who gave you COVID19. https://www.coronadetective.eu, 2020.
[4] Guidelines 04/2020 on the use of location data and contact tracing tools in the context of the covid-19 outbreak. https://edpb.europa.eu/our-work-tools/our-documents/guidelines/guidelines-042020-use-location-data-and-contact-tracing_en, 2020.
[5] 'it's covid! stay away!' latin america's health workers face rising hostility. https://www.reuters.com/article/us-health-coronavirus-latinamerica-medic-idUSKCN21X2WL, 2020.
[6] Why google should stop logging contact-tracing data. https://blog.appcensus.io/2021/04/27/why-google-should-stop-logging-contact-tracing-data/, 2021.
[7] Nadeem Ahmed, Regio A Michelin, Wanli Xue, Sushmita Ruj, Robert Malaney, Salil S Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, and Sanjay K Jha. A survey of COVID-19 contact tracing apps. *IEEE Access*, 8:134577–134601, 2020.
[8] Apple and Google. Exposure Notification - Bluetooth Specification v1.2. Technical report, April 2020.
[9] J. Bay, Joel Kek, A. Tan, and Chai Sheng Hau. Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders, 2020.
[10] Daniel J Bernstein. Curve25519: new diffie-hellman speed records. In *Workshop on Public Key Cryptography*, pages 207–228, 2006.
[11] Wasilij Beskorovajnov, Felix Dörre, Gunnar Hartung, Alexander Koch, Jörn Müller-Quade, and Thorsten Strufe. Contra corona: Contact tracing against the coronavirus by bridging the centralized-decentralized divide for stronger privacy. *IACR Cryptol. ePrint Arch.*, 2020:505, 2020.
[12] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Symposium on Operating Systems Principles*, page 441–459, 2017.
[13] Antoine Boutet, Nataliia Bielova, Claude Castelluccia, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. Proximity Tracing Approaches - Comparative Impact Analysis. Research report, 2020.
[14] Antoine Boutet, Claude Castelluccia, Mathieu Cunche, Alexandra Dmitrienko, Vincenzo Iovino, Markus Miettinen, Thien Duc Nguyen, Vincent Roca, Ahmad-Reza Sadeghi, Serge Vaudenay, Ivan Visconti, and Martin Vuagnoux. Contact Tracing by Giant Data Collectors: Opening Pandora's Box of Threats to Privacy, Sovereignty and National Security. University works, EPFL, Switzerland ; Inria, France ; JMU Würzburg, Germany ; University of Salerno, Italy ; base23, Geneva, Switzerland ; Technical University of Darmstadt, Germany, December 2020.
[15] Antoine Boutet and Sébastien Gambs. Demo: Inspect what your location history reveals about you Raising user awareness on privacy threats associated with disclosing his location data. In *ACM International Conference on Information and Knowledge Management, CIKM*, pages 2861–2864, 2019.
[16] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure: SGX cache attacks are practical. In *Workshop on Offensive Technologies*, 2017.
[17] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *ACM conference on Computer and communications security*, pages 132–145, 2004.
[18] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. DESIRE: A Third Way

for a European Exposure Notification System Leveraging the best of centralized and decentralized systems. Technical report, Inria and INSA de Lyon, May 2020. https://github.com/3rd-ways-for-EU-exposure-notification/project-DESIRE, also https://hal.inria.fr/hal-02570382/en/.

[19] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. ROBERT: ROBust and privacy-presERving proximity Tracing. Technical report, Inria and INSA de Lyon, April 2020. https://github.com/ROBERT-proximity-tracing/documents/, also https://hal.inria.fr/hal-02611265/en/.

[20] Justin Chan, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob Sunshine, et al. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing. *arXiv preprint arXiv:2004.03544*, 2020.

[21] David Chaum, Bert den Boer, Eugène van Heyst, Stig Mjølsnes, and Adri Steenbeek. Efficient offline electronic checks. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology*, pages 294–301, 1990.

[22] Hyunghoon Cho, Daphne Ippolito, and Yun William Yu. Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs, 2020.

[23] Mathieu Cunche, Antoine Boutet, Claude Castelluccia, Cédric Lauradoux, and Vincent Roca. On using Bluetooth-Low-Energy for contact tracing. Research report, Inria Grenoble Rhône-Alpes ; INSA de Lyon, June 2020.

[24] Heyam F. Dalky, Ayman M. Hamdan-Mansour, Basil H. Amarneh, Manar AlAzzam RN, Nuha Remon Yacoub, Anas H. Khalifeh, Mohammed Aldalaykeh, Alaa Fawwaz Dalky, Rana Akram Rawashdeh, Dalal Bashir Yehia, and Malek Alnajar. Social discrimination perception of health-care workers and ordinary people toward individuals with covid-19. *Social Influence*, 15(2-4):65–79, 2020.

[25] Yves-Alexandre De Montjoye, Cesar A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[26] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22:644–654, 1976.

[27] Stephen Farrell Douglas Leith. Contact tracing app privacy: What data is shared by europe's gaen contact tracing apps. In *SecureComm*, 2020.

[28] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography*, pages 254–271, 2013.

[29] Jean-Marie Gorce, Malcolm Egan, and Rémi Gribonval. An efficient algorithm to estimate Covid-19 infectiousness risk from BLE-RSSI measurements. Research Report RR-9345, Inria Grenoble Rhône-Alpes, May 2020.

[30] Yaron Gvili. Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc. Cryptology ePrint Archive, Report 2020/428, 2020.

[31] Vincenzo Iovino, Serge Vaudenay, and Martin Vuagnoux. On the effectiveness of time travel to inject covid-19 alerts. Cryptology ePrint Archive, Report 2020/1393, 2020.

[32] Douglas J. Leith and Stephen Farrell. Measurement-based evaluation of google/apple exposure notification api for proximity detection in a light-rail tram. *PLOS ONE*, 15(9):1–16, 09 2020.

[33] Tianshi Li, Camille Cobb, Jackie, Yang, Sagar Baviskar, Yuvraj Agarwal, Beibei Li, Lujo Bauer, and Jason I. Hong. What makes people install a covid-19 contact-tracing app? understanding the influence of app design and individual difference on contact-tracing app adoption intention. *arXiv preprint arXiv:2012.12415*, 2021.

[34] Tianshi Li, Jackie, Yang, Cori Faklaris, Jennifer King, Yuvraj Agarwal, Laura Dabbish, and Jason I. Hong. Decentralized is not risk-free: Understanding public perceptions of privacy-utility trade-offs in covid-19 contact-tracing apps, 2020.

[35] UK National Health Service NHSX. NHS COVID-19 App. https://www.nhsx.nhs.uk/covid-19-response/nhs-covid-19-app/, July 2020.

[36] Australian Government Department of Health. COVIDSafe app. https://www.health.gov.au/resources/apps-and-tools/covidsafe-app, April 2020.

[37] Olga Ohrimenko, Michael T. Goodrich, Roberto Tamassia, and Eli Upfal. The melbourne shuffle: Improving oblivious storage in the cloud. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming*, pages 556–567, 2014.

[38] Michael Edmund O'Callaghan, Jim Buckley, Brian Fitzgerald, Kevin Johnson, John Laffey, Bairbre McNicholas, Bashar Nuseibeh, Derek O'Keeffe, Ian O'Keeffe, Abdul Razzaq, et al. A national survey of attitudes to covid-19 digital contact tracing in the republic of ireland. *Irish Journal of Medical Science*, pages 1–25, 2020.

[39] Ramesh Raskar, Isabel Schunemann, Rachel Barbar, Kristen Vilcans, Jim Gray, Praneeth Vepakomma, Suraj Kapa, Andrea Nuzzo, Rajiv Gupta, Alex Berke, Dazza Greenwood, Christian Keegan, Shriank Kanaparti, Robson Beaudry, David Stansbury, Beatriz Botero Arcila, Rishank Kanaparti, Vitor Pamplona, Francesco M Benedetti, Alina Clough, Riddhiman Das, Kaushal Jain, Khahlil Louisy, Greg Nadeau, Vitor Pamplona, Steve Penrod, Yasaman Rajaee, Abhishek Singh, Greg Storm, and John Werner. Apps gone rogue: Maintaining personal privacy in an epidemic, 2020.

[40] Elissa M. Redmiles. User concerns & tradeoffs in technology-facilitated contact tracing, 2020.

[41] Bluetooth SIG. *Bluetooth Core Specification v5.1.* 2019. Accessed: 2019-08-30.

[42] DP3T Team. Privacy and Security Attacks on Digital Proximity Tracing Systems, April 2020.

[43] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy, 2020.

[44] Carmela Troncoso, Marios Isaakidis, George Danezis, and Harry Halpin. Systematizing Decentralization and Privacy: Lessons from 15 Years of Research and Deployments. *Privacy Enhancing Technologies*, 2017(4):307 – 329, 2017.

[45] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth Paterson, Srdjan Čapkun, David Basin, Jan Beutel, Dennis Jackson, Marc Roeschlin, Patrick Leu, Bart Preneel, Nigel Smart, Aysajan Abidin, Seda Gürses, Michael Veale, Cas Cremers, Michael Backes, Nils Ole Tippenhauer, Reuben Binns, Ciro Cattuto, Alain Barrat, Dario Fiore, Manuel Barbosa, Rui Oliveira, and José Pereira. Decentralized privacy-preserving proximity tracing, 2020.

[46] Carmela Troncoso, Mathias Payer, Marcel Salathé, James Larus, Dr Wouter Lueks, Theresa Stadler, Dr Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth Paterson, Srdjan Capkun, David Basin, Dennis Jackson, KU Leuven, Bart Preneel, Nigel Smart, Dr Dave Singelee, Dr Aysajan Abidin, TU Delft, Seda Guerses, and Cas Cremers. Decentralized Privacy-Preserving Proximity Tracing. Technical Report v1.0, April 2020.

[47] Serge Vaudenay. Analysis of dp3t. Cryptology ePrint Archive, Report 2020/399, 2020.

[48] Serge Vaudenay. Centralized or decentralized? the contact tracing dilemma. Cryptology ePrint Archive, Report 2020/531, 2020.

[49] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *IEEE Symposium on Security and Privacy*, page 640–656, 2015.

[50] Yuan Zhong, Nicholas Jing Yuan, Wen Zhong, Fuzheng Zhang, and Xing Xie. You are where you go: Inferring demographic attributes from location check-ins. In *ACM International Conference on Web Search and Data Mining, WSDM*, pages 295–304, 2015.