Information Sciences 617 (2022) 103–132

Contents lists available at ScienceDirect

Information Sciences

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

A centralized contact-tracing protocol for the COVID-19 pandemic

Francesco Buccafurri [*], Vincenzo De Angelis, Cecilia Labrini

Universitá Mediterranea di Reggio Calabria, via dell'Universitá, 25, Reggio Calabria 89124, Italy

ARTICLE INFO

Article history:
Received 13 March 2022
Received in revised form 21 October 2022
Accepted 22 October 2022
Available online 27 October 2022

Keywords:
Digital contact tracing
Privacy
Proximity-based services
COVID-19
DP-3T

ABSTRACT

Digital contact tracing (DCT) is one of the weapons to be used against the COVID-19 pandemic, especially in a post-lockdown phase, to prevent or block foci of infection. As DCT systems can handle highly private information about people, great care must be taken to prevent misuse of the system and actions detrimental to people's privacy, up to mass surveillance. This paper presents a new centralized DCT protocol, called ZE2-P3T (Zero Ephemeral Exchanging Privacy-Preserving Proximity Protocol), which relies on smartphone localization but does not give any information about the user's location and identity to the server. Importantly, the fact that no exchange of ephemeral identities among users is required is the basis of the strong security of the protocol, which is proven to be more secure than the state-of-the-art protocol DP-3T/GAEN.

1. Introduction

The role of digital contact tracing (DCT) to control the spread of the COVID-19 pandemic has recently been studied [1,2]. DCT should be considered as a complementary task with respect to traditional contact tracing. Indeed, it is able to identify contacts that escape the investigation activities carried out by contact tracers (for example, whether they regard contacts with people unknown to the index case). In addition, the specific characteristics of COVID-19 infection (variable symptoms, frequent asymptomatic carriers, and incubation times relatively short) require fast detection of at-risk contacts. DCT represents one of the weapons against the pandemic, along with other digital weapons pursuing different goals, such as early diagnosis [3] or early warning [4].

DCT systems adopted in many countries use Bluetooth Low Energy (BLE) proximity measurements and rely on the exchange of ephemeral identifiers. They are pseudo-random self-generated numbers designed to be unlinkable with each other and with the real identity of the user. At least in the European Union, the prevailing protocol is DP-3T (Decentralized Privacy-Preserving Proximity Tracing) [5], typically implemented via GAEN (Google-Apple Exposure Notification) [6]. DP-3T/GAEN is decentralized (i.e., does not delegate contact detection to a server) and does not utilize localization systems (such as GPS) to detect proximity, but only BLE. On the other hand, localization-based approaches lead to concerns about privacy, as witnessed by the recent act of the European Union that disapproves the use of GPS [7].

Privacy concerns still remain an open issue, due to the fact that DCT handles sensitive information about people, and any system misuse or detriment to people's privacy (up to mass surveillance) should be prevented.

* Corresponding author.
E-mail address: bucca@unirc.it (F. Buccafurri).

https://doi.org/10.1016/j.ins.2022.10.101
0020-0255/© 2022 Elsevier Inc. All rights reserved.

As a matter of fact, several vulnerabilities affecting DP-3T/GAEN have been reported in the literature [8–10]. They can lead to breaking protocol integrity and users' privacy. Some of these concerns GAEN [11], about how data are transmitted to back-end servers of the Google ecosystem. Privacy and security concerns are an obstacle to the massive adoption of DCT. On the other hand, if the system is not adopted by the largest number of people, it may give very limited benefits to the fight against the pandemic.

The research question addressed in this paper is whether it is possible to define an alternative approach that overcomes the drawbacks of DP-3T/GAEN. To understand this, we start with the observation that the basis of most of the attacks reported in the literature on DP-3T/GAEN is that, through BLE, smartphones exchange ephemeral identifiers. Though they are pseudonyms, they are uniquely associated with individuals. Exchanging them means that the user loses exclusive control.

It is rather intuitive that, to avoid the exchange of ephemeral identifiers, a centralized approach could be adopted in which a server collects the locations of smartphones. This way, the server itself can detect the proximity of individuals. However, this solution would be much worse from the privacy perspective, because the server would continuously track people.

In this paper, we face the challenge of centrally detecting the proximity of individuals, thus avoiding the exchange of ephemeral identifiers, without providing the server with any information useful to track people. Therefore, the goal we pursue is to design a centralized approach that, unlike those existing in the literature, does not provide the server with tracking capabilities. Intuitively, since the exchange of ephemeral identifiers is no longer necessary, the drawbacks of DP-3T/GAEN are solved and no new security and privacy issues are introduced, resulting in an advancement of the state of the art.

To achieve the above goal, we rely on the availability of an effective localization system (the technologies for this are already mature and very very near-term progress is expected with 5G and 6G). The proposed centralized protocol is called ZE2-P3T (*Zero Ephemeral Exchanging Privacy-Preserving Proximity Protocol*). Unlike DP-3T/GAEN, the proposed protocol does not rely on the exchange of (even pseudonym) identities. Despite the use of localization, by using some cryptographic obfuscation mechanisms, our protocol does not allow the server to track people. Moreover, since ZE2-P3T does not use Bluetooth, users are not exposed to Bluetooth vulnerabilities [12,13].

We prove in the paper that ZE2-P3T is more secure than DP-3T/GAEN. Specifically, the paper provides a theoretical framework that allows us to *measure* the security improvements over DP-3T/GAEN in a quantitative way and across a multidimensional domain, composed of three dimensions: (1) type of attacker, (2) type of target, and (3) range of the attack.

Finally, the implementation of the main modules of the solution has been executed (and made publicly available), to provide research demonstrators that can be used as building blocks for a software system that fully implements our solution.

Moreover, we exploited the implemented prototype to test the performance of our solution, by verifying the feasibility of the server-side computation.

The contribution given by our paper can be summarized as follows:

- A new digital contact tracing protocol based on the centralized model has been proposed,
- although there is a server that collects information about users, users' privacy is strictly preserved,
- the proposed protocol is formally proven to be more secure (against attacks on users' privacy and protocol integrity) than the state-of-the-art decentralized protocol, which is DP-3T/GAEN,
- the proposed theoretical framework, through which we performed the security analysis, provides a quantitative measure of the security increase over DP-3T,
- the feasibility of the proposal has been tested by experiments.

To give the flavor of our approach, we give an overview of how our solution works.

The territory is virtually divided into overlapping square microcells, whose size is of the magnitude of the safety distance. Thanks to microcell overlapping, we guarantee that if two users, say Alice and Bob, are at distance less than the safety distance, they occupy at least one shared microcell. Each microcell is represented by its centroid, so that Alice and Bob, autonomously (i.e., with no reciprocal interaction), identify at least one common centroid. Each smartphone sends the server, periodically, an ephemeral identifier along with the coordinates of the centroids of all the microcells to which it belongs. Ephemeral identifiers, as in DP-3T/GAEN, are changed according to a rounding protocol. However, whilst in DP-3T/GAEN the smartphones store the ephemeral identifiers of the encountered contacts (in such a way that anybody can be informed about past at-risk contacts), in our protocol, smartphones do not exchange the ephemeral IDs with other users. Therefore, Bluetooth is not exploited by our solution (with positive impacts on security). Observe that the fact that the smartphones send the coordinates to the server is not an issue from the point of view of privacy. Indeed, these coordinates are sent in a *salted hashed* form, by using a salt broadcast by the telephone service provider and rounded periodically. This way, the server cannot know where the centroid is located, because the salted hashed coordinates are not reversible and are always different. Once an infection is reported, the server is able to detect all the contact at risk, by finding groups of ephemeral identifiers associated with the same salted hashed centroid within the same time slot. Indeed, this means that the smartphones were simultaneously in the same microcell. Thus, all these ephemeral IDs are associated with users at risk of contagion. The server has to broadcast these ephemeral IDs. Every smartphone can check if some of the received ephemeral IDs are stored in the set of past ephemeral IDs (recall that ephemeral IDs are periodically changed) not earlier than the safety time window (i.e., 14 days, according to WHO). If this is the case, then the user is alerted about the risk. In the example above, if Alice tests positive, then the ephemeral IDs of Alice of the last 14 days are notified to the server. Consequently, the ephemeral used by Bob when they came into contact will be included in the list of ephemeral IDs broadcast by the server. Therefore, Bob will be alerted by his smartphone. Regarding the infection reporting, we specify that the notification of past ephemeral IDs by the infected user is done only if it is authorized by the health facility which tested the patient. This is done by using a blind-signature mechanism to prevent the health-care facility from linking the patient's real identity with the transmitted ephemeral identifiers.

Observe that the smartphone has to keep only the ephemeral IDs of the last 14 days. Actually, some small additional information is kept by the smartphones to associate the alert, if any, with a level of risk. Therefore, no significant storage overhead is required to clients.

Importantly, the fact that the ephemeral identifiers are never exchanged among smartphones is the basis of the improvements of our protocol in terms of privacy and security with respect to DP-3T/GAEN. This is clearly shown in the last part of the paper, when security aspects are addressed.

The structure of the paper is the following. In Section 2, we present the related literature. Then, in Section 3, we propose the new centralized protocol, called ZE2-P3T. A prototype of the protocol is implemented and used to perform an experimental campaign in Section 4. The findings of this campaign are presented in Section 5. In Section 6, we provide a detailed security analysis of our protocol and compare it with DP-3T/GAEN. In Section 7, we give some general discussions about the contribution given by our work and, in Section 8, we draw our conclusions.

2. Related work

The COVID-19 pandemic certainly represents one of the most difficult challenges modern society has ever faced. To counteract and slow the spread of the virus, new ways, new strategies, and solutions are being sought every day, in every sector. From the side of technological measures, researchers are investing their effort to propose digital contact tracing solutions that preserve privacy and comply with current regulations. Existing protocols and applications can be classified in many different ways.

A possible classification can be done on the basis of the model on which they rely.

The model defines how the server is used and which data are required (or stored) by it. There are three possible models: (1) decentralized, (2) centralized, and (3) hybrid [14,15].

Decentralized Approaches. Several solutions adopt a decentralized approach, with the goal of protecting users' privacy from server misbehavior. In this class of approaches, the most relevant protocol is certainly the Decentralized Privacy-Preserving Proximity Tracing (DP-3T) [5]. For this protocol, two designs are defined, namely *Low-Cost* and *Uninkable*. In both designs, the protocol is based on ephemeral pseudonyms (called *EphIDs*) sent via BLE which are registered by nearby users. Apple and Google [6] have teamed up to realize, on the respective Operating Systems (iOS and Android), an implementation of Low-Cost DP-3T, called *Google Apple Exposure Notification* (GAEN). Several adopted apps leverage the GAEN APIs, such as the Swiss SwissCoviD app, the German Corona-Warn app, and the Italian Immuni app.

Bluetooth-based decentralized systems, such as systems that rely on DP-3T/GAEN, are vulnerable to Paparazzi attack and therefore can be exploited for mass surveillance [16]. In [8], the authors proposed two decentralized systems, based on BLE, named Pronto-B2 and Pronto-C2, respectively. These systems appear to be more resistant than DP-3T against mass surveillance attacks. Both systems can optionally be implemented using blockchain technology, but Pronto-B2 is designed to be more efficient and practical. An MIT-led research collaboration has developed the Private Automated Contact Tracing protocol, called PACT (East-coast) [17], which allows the user to store extra metadata, such as location information, to increase the accuracy of the system. Researchers from the University of Washington proposed PACT (West-coast) protocol (Privacy-sensitive protocols And mechanisms for mobile Contact Tracing) [18]. Compared with PACT (East-coast), PACT (West-coast) saves storage space by storing fewer seeds than the PACT (East-coast) app. PACT (West-coast) is also susceptible to linkage and enumeration attacks [14]. CAUDHT [19] is a decentralized system based on distributed hash tables and blind signatures. Another decentralized protocol based on Bluetooth is TCN (Temporary Contact Numbers) [20]. To solve the problem of scalability, the protocol switches from purely random TCNs to TCNs generated deterministically from seed data. The price it pays for greater scalability is a reduction in privacy because the TCNs derived from the same seed can be linked together.

There are also several decentralized DCT proposals based on IoT [21], blockchain [22,23], or both [24]. The system proposed in [21] can be configured to support different models, ranging from the fully decentralized to the fully centralized one. Most decentralized approaches use Bluetooth. It is worth noting that switching on the Bluetooth interface of smartphones, can make the devices vulnerable to a variety of attacks, also tailored to GAEN-based digital contact tracing, as shown in [13].

Centralized Approaches. Several solutions choose a centralized approach, such as NTK [25] and ROBERT [26] which have been developed inside Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) [27]. Centralization has in general the advantage of providing epidemiologists with more useful data, thus allowing more effective actions to be taken to defeat the virus. However, some scholars fear that these systems could become a tool of massive surveillance in the hands of governments [28]. PEPP-PT NTK is a proximity tracing system, based on BLE [25]. ROBust and privacy-presERving proximity Tracing protocol (ROBERT) is jointly developed by researchers at INRIA (France) and Fraunhofer (Germany). Similarly to DP-3T, NTK and ROBERT are based on ephemeral pseudonyms sent via BLE that are registered by nearby users, with the difference that the secret keys for calculating EphIDs are created and handled by a back-end server and not by the user's smartphone [29]. Adopted apps, such as TraceTogether (Singapore) and CovidSafe (Australia) are based on the Bluetrace protocol [30]. The two apps have many similar features but differ mainly in the lifetime of the EphIDs, as Trace-Together uses a value of 15 min, while CovidSafe uses a value of 2 h. For this reason, CovidSafe is more vulnerable to replay attacks [31]. In [32], the authors propose a solution that resists replay attacks and, if location data are present, to relay attacks (see Section 6).

Another solution, called EPIC [33], is still based on Bluetooth technology, and offers a fine-grained human-to–human contact tracing scheme with hybrid wireless and localization technology. EPIC introduces a matching method that uses homomorphic encryption to match devices and, then, identify contacts. However, the system can suffer from serious privacy and scalability issues [34].

Hybrid Approaches. A number of approaches combine the characteristics of centralized and decentralized architectures, such as DESIRE [35], ConTra Corona [36], and EpiOne [37].

Another possible classification can be done on the basis of the ability of the system to track the user's location. Indeed, location data can be useful for epidemiologic analysis and also for applying disinfection measures to physical places. Obviously, knowing the location of users could come at a very high price in terms of privacy. All the systems considered so far do not collect exploitable users' location information. However, despite privacy issues, this is a class of techniques that is well represented in the literature.

User-location-aware approaches. In this class of techniques, some approaches combine Bluetooth technology with GPS, and others rely only on GPS. Hamagen [38] is based on a decentralized architecture and does not rely on logging encounters with other users in proximity via Bluetooth. Instead, it cross-checks (locally on the user's smartphone) the GPS history of the smartphone with the historical–geographical data of the cases identified by the Ministry of Health. If the application discovers that a user has been in the same place and at the same time as a diagnosed case, a notification is displayed on the user's phone. However, besides privacy concerns, Hamagen exposes users to the risk of data breach server-side [39]. The Aarogya Setu [40] is based on a centralized architecture and uses both Bluetooth and GPS. This protocol also collects location data (GPS coordinates) and self-assessment data. [41] offers a solution on how to make contact tracing centralized based on GPS data while preserving users' privacy. The system uses a central party (HA) and applies multi-party computation (MPC) to achieve privacy. However, these solutions are not scalable [34], due to the computational overhead required by the adopted cryptographic protocols.

Our solution starts from the above reference framework, with the aim to overcome the privacy and security issues of current decentralized solutions. Our approach is centralized and is based on privacy-preserving absolute location detection.

Concerning the location-aware class of methods, although our protocol requires that users send their localization information to the server, this information cannot be exploited by the server for anything but contact detection. This happens because the information is sent in an obscured form. Importantly, our solution only relies on localization technologies and does not require the use of Bluetooth. Therefore, it does not suffer from the threats described in [13].

Concerning localization technologies, in this paper, we do not make a specific choice, by assuming that a precise indoor-outdoor technology is available. We observe that this assumption is well-founded. Since smartphones are equipped with GPS, WLAN, gyroscope, accelerometer, magnetometer, and other sensors, it is possible to achieve high-precision localization. For example, the combination of GPS and PDR (Pedestrian Dead Reckoning) [42] enables localization for both indoor and outdoor environments.

PDR is an algorithm that allows us to estimate the movement of pedestrians, using MEMS sensors, i.e., accelerometer, gyroscope, and magnetometer, on board of the smartphone. In particular, for indoor environments, there are different measurement techniques [43] such as *Angle of arrival (AOA), Cell Identity (CI), Time of Arrival (TOA), and Signal Strength (RSSI)*, or the Earth's magnetic field.

To achieve better accuracy, new technologies such as 5G and 6G [44] can be used, also to obtain a continuous localization service between external and internal environments [45]. Positioning accuracy can be significantly increased by using the features offered by 5G, such as wider bandwidth (mmWave frequencies), data from a number of sensors and technologies (WiFi, GNSS, relative device-to-device positioning, inertial measures, etc.), and a dense infrastructure. Furthermore, in 5G, many new technologies have been proposed, such as massive Multiple Input Multiple Output (MIMO), millimeter Wave (mmWave) communication, ultra-dense network (UDN), and device-to-device (D2D) communication, which improve both communication performance but also positioning accuracy. 6G technology, following the trend initiated by 5G systems, will continue to develop towards even higher frequency ranges, wider bandwidths, and massive antenna arrays. This will also allow for localization with a degree of accuracy at the level of the centimeter [44].

The original idea underlying this paper has been presented in [46]. This paper strongly extends [46] as follows.

First, we reduced the workload server-side by moving the whole risk computation from the server to the client. Then, we extensively deal with the security analysis of the proposal. Specifically, we provide a theoretical framework allowing us to represent a multiplicity of threat models and a security metric to compare the different protocols. Therefore, unlike the empirical and brief security analysis provided in [46], we prove formally the security of our approach by comparing it with the two designs of DP-3T, and by also giving a measure of the security gap between the protocols. This is per se a new relevant contribution. Moreover, the empirical framework usually adopted in the context of DCT, based on the enumeration of existing attacks, is also preserved and contextualized within the above theoretical framework. Finally, unlike [46], in this paper we provide the implementation of the main modules of the solution as research demonstrators (https://github.com/vincenzodeangelisrc/ZE2-P3T) and use it to test the performance of the solution (see Sections 4 and 5).

The novelty of our approach with respect to the existing literature is that our protocol, unlike the other proposals based on the centralized model and in favor of privacy, does not give the server any information besides the fact that some pairs of random numbers (associated with humans) are close to each other somewhere in the territory. The specific novelty is that this is done without using complex cryptographic protocols and, thus, with no server-side computational overhead. Another important difference of our solution with respect to the relevant state of the art is that Bluetooth is not required. This implies that, as observed earlier, users' smartphones are not exposed to the threats related to the usage of this interface. But the real significance of the proposal strictly concerns the security aspects of the protocol (apart from technological aspects), as it is clearly shown in Section 6. Indeed, we show that, in terms of security and privacy goals, our solution outperforms the state-of-the-art protocol based on the decentralized model, which is DP-3T/GAEN.

3. The proposed protocol

In this section, we describe the proposed protocol, named *Zero Ephemeral Exchanging Privacy-Preserving Proximity Tracing* (ZE2-P3T, for short). As anticipated earlier, the protocol relies on the presence of a sufficiently precise localization system utilizable with sensors on board of smartphones. Although this aspect is not within the scope of this paper, it is worth noting that the availability of such a system is not an abstraction. The scientific literature (see Section 2 for details) clearly shows that these technologies are already sufficiently mature to offer a precision (also seamless indoor-outdoor) which is much higher than that required for DCT, and much higher than that provided by BLE proximity measurement. Moreover, the immediate next future with 5G and, then 6G, will allow even more precision in smartphone localization techniques.

Therefore, in the description of the protocol, we generically refer to a *localization system*, without considering a specific technology.

3.1. Protocol setting

Let G be a large geographical area, for example a country, in which the contacts among users have to be tracked. In our model, G contains several *microcells* c such that: (1) they cover the entire area G and (2) if the distance between two points x and y is less than a threshold value d, then there exists a microcell which contains both x and y.

In our setting, the microcells are squares of side $2d$ organized as in Fig. 1.

Therein, we use different colors to better highlight the different microcells (13 in total). It is easy to see that one point is always, simultaneously, within two different microcells and that two points at a distance less than d have a microcell in common. For example, in the figure, the point x is within the blue and green microcells, while the point y is within the red and green microcells.

With each microcell c, we associate a point C called *centroid* that corresponds to the center of the square. The set of all the centroids is public and each user, through the localization system, is able to identify the centroids associated with the two microcells in which the user is located.

The actors involved in our protocol are:

- The *Users* moving within the territory. Each user owns a smartphone equipped with the localization system.
- The *Server* that receives the information sent by users' devices and stores the contacts in pseudonymous form (yet unlinkable).

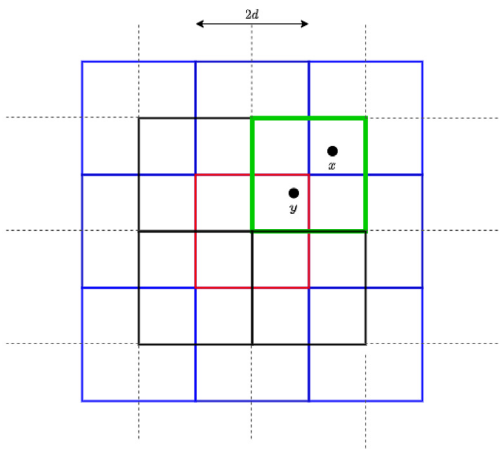


Fig. 1. Microcells organization.

- The *Health Facility* HF that tests users to diagnose the disease and allows an infected user to send the necessary information to the server for reporting the infection.
- The *Telephone Service Provider* TSP which, periodically, for each zone P (covering a number of microcells), sends a different random R_P (called *salt*) to all the users in a *zone*. The zones cover the entire territory.

The first three actors are also present in DP-3T (even though they act in a different way). We provide some details about the TSP.

Its role is to provide an unpredictable value to the users physically present in a given zone. As we will see throughout the rest of the section, this value is used for salting the hash applied to the centroid, in such a way that the digest cannot be reversed by the server (to disclose the position).

We assume that every zone contains several microcells. Since in real-life cellular systems the coverage overlap is at least 1 meter, we can argue that if two users are within the safety distance (of the magnitude of 1–2 meters, according to WHO), they receive at least one common salt if d is fixed to this value. Note that, due to the overlapping of two zones P_1 and P_2, the same user can receive two salts R_{P_1} and R_{P_2}.

R_P is rounded by TSP with a certain frequency, which is a parameter of the system (we assume that this can be set in accordance with the average permanence time of a user in a zone).

For simplicity, we assume that this service is provided by a unique TSP (to avoid dealing with the problem of coordination of multiple TSP in overlapping cells) and that the roaming mechanism can be enabled to ensure maximum coverage.

Before going into details, we provide some security assumptions. TSP, Server, and HF are assumed to be TTPs (Trusted Third Parties). TSP is assumed to be independent. Instead, HF and Server cannot be considered independent because they could be part of the same National Health System. In Section 6, concerning the adversary model, we relax the above assumptions by giving to the above parties also the role of adversary.

We distinguish five logical phases in our protocol.

3.2. User-side activity

Each τ seconds, each user U detects, through the localization system, the two centroids C_1 and C_2 of the microcells in which they are located. Moreover, U detects one or two salts provided by the TSP depending on whether the user is located just in a single zone or in the overlap between two zones. We consider the more general case of two salts R_{P_1} and R_{P_2}.

For each pair of centroid-salt, the user builds a tuple of the form $T = \langle Eph, H, (\rho, \theta), t\rangle$. *Eph* is said *ephemeral identifier*, H, when the context is clear, is called simply *digest*, (ρ, θ) are called *coordinates*, and t is called *timestamp*.

We show in detail how these tuples are built in the most complex case of four tuples:

1. $T_1 = \langle Eph_1, H_{11}, (\rho_1, \theta_1), t\rangle$,
2. $T_2 = \langle Eph_2, H_{21}, (\rho_2, \theta_2), t\rangle$,
3. $T_3 = \langle Eph_3, H_{12}, (\rho_1, \theta_1), t\rangle$,
4. $T_4 = \langle Eph_4, H_{22}, (\rho_2, \theta_2), t\rangle$.

where Eph_i $(1 \leqslant i \leqslant 4)$ is a random value, $H_{ij} = h(C_i || R_{P_j})$ $(i, j \in \{1, 2\})$ denotes the application of a cryptographic hash function on the concatenation between a centroid and a salt, (ρ_i, θ_i) (where $i \in \{1, 2\}$) are the relative polar coordinates of U with respect to the centroid C_i, and t is the current UNIX timestamp (the same for all the four tuples).

These four tuples are stored locally by U for a given time interval, that we call *retention time*, (e.g., 14 days, according to the original recommendation given by WHO). Observe that in the case a single salt R_{P_1} is retrieved by U, just the first two tuples are generated.

As discussed in Section 6, the role of the salt is to prevent dictionary attacks performed by the server with the goal of identifying the centroids (and thus the location) of users.

Now, for each tuple $T = \langle Eph, H, (\rho, \theta), t\rangle$, a *restricted* tuple $\overline{T} = \langle Eph, H, t\rangle$ is sent to the server. In other words, the restricted tuples do not include the relative coordinates of the users. Their function is to improve the accuracy of the risk computation as discussed in Section 3.6, which is performed client-side and only when necessary.

The sequence of actions performed in this phase is reported in the sequence diagram of Fig. 2.

3.3. Server-side activity

We assume a time-slot mechanism is enabled. Specifically, the time is partitioned in time slots $\tau_k = [t_k, t_{k+1}[$, where t_k and t_{k+1} are UNIX timestamps such that $t_{k+1} - t_k = \tau$. In words, the size of a time slot is a constant value equal to the period of user sending.

For each received restricted tuple $\overline{T} = \langle Eph, H, t\rangle$, the server verifies that the current UNIX timestamp is within the current time slot or within the previous one (to take into account border-line sending). Say τ_k the time slot such that $t \in \tau_k$. Then, the server builds a *clustered tuple* $\tilde{T} = \langle Eph, H, \tau_k\rangle$. The term clustered derives from the fact that the server maintains, for each

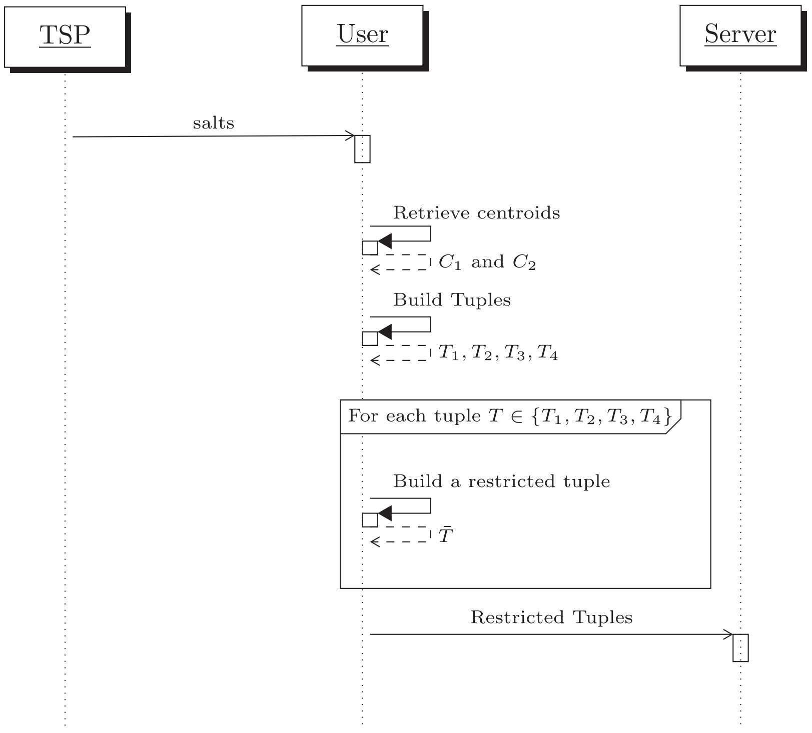


Fig. 2. Sequence diagram of the user-side activity.

time slot τ_k, a list of *contact clusters*. A *contact cluster* is a set of clustered tuples with the same time slot and same digest. A contact cluster represents a set of users who shares a centroid in the same time slot, i.e., users experienced unsafe contacts.

To avoid the inflationary growth of the server-side database size, the server maintains only the information of the last X days (with X properly greater than the retention time).

The sequence of actions performed in this phase is reported in the sequence diagram of Fig. 3.

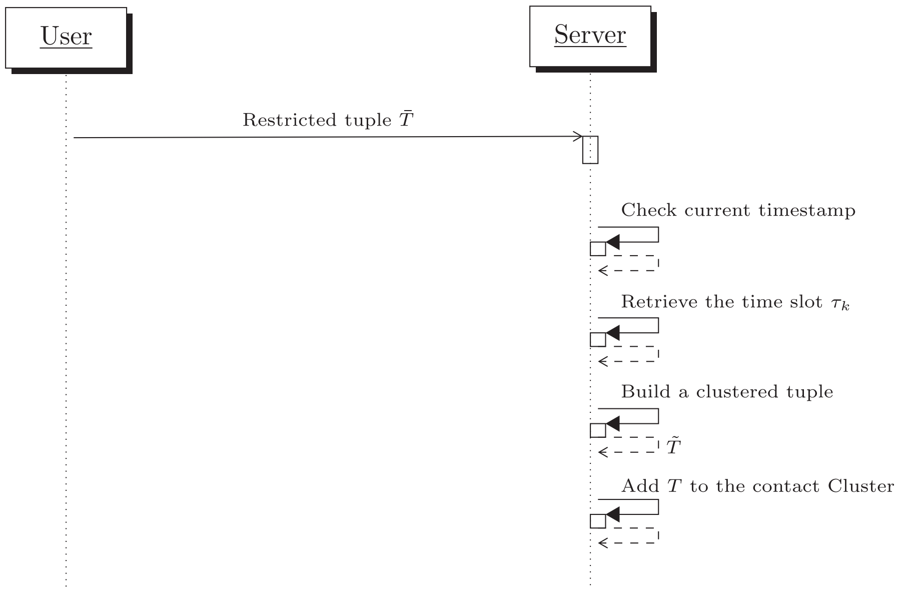

User
Server
Restricted tuple $\bar{T}$
Check current timestamp
Retrieve the time slot τ_k
Build a clustered tuple
$\tilde{T}$
Add T to the contact Cluster

Fig. 3. Sequence diagram of the server-side activity.

3.4. Infection reporting

Suppose the user U is tested positive for the infection in the health facility HF. Infection reporting requires that U sends the server the non-restricted tuples generated as described in Section 3.2 in the last X days, where X is the retention time. We denote by $\mathscr{T}$ the set of such tuples.

To avoid fake positive reports, U needs authorization by HF. We rely on a 1024 bits RSA blind signature scheme. As discussed in Section 6, the blind signature also prevents that, even if the server colludes with HF, it is not able to link all the tuples provided to the real identity of U. The procedure is the following. First, U generates a random α of $1024 - 256 = 768$ bits and obtains $M = \alpha || h(\alpha)$, where h denotes the application of a cryptographic hash function with digests of 256 bits (e.g., SHA-256). At this point, U contacts HF to obtain the RSA blind signature on M. This is done by sending HF an obfuscated message $obf(M)$ unlikable to M. Let denote by Q the message with blind signature produced by HF. U unblinds Q and obtains the signature of HF $\sigma(M)$ on M. Finally, at a later moment, U sends $\sigma(M)$ and all the non-restricted tuples they locally stored in the server.

The server verifies the signature and checks that $M = \alpha || h(\alpha)$. To avoid replay attacks, the server burns the random M, so that it cannot be used anymore.

The infection-reporting procedure is summarized in the sequence diagram of Fig. 4.

3.5. Contact detection

After receiving a set of tuples $\mathscr{T}$ by an infected user, the server performs the following actions. For each tuple $T \in \mathscr{T}$ with $T = \langle Eph, H, (\rho, \theta), t \rangle$, the server retrieves the time slot τ_k such that $t \in \tau_k$. Then, it retrieves the contact cluster $\mathscr{C}$ associated with the digest H. If two distinct tuples fall within the same time slot, the server randomly chooses one of them and skips the other. This is to prevent a malicious user from fictitiously increasing the number of ephemeral identities corresponding to that infection reporting. For each clustered tuple $\tilde{T} = \langle Eph\prime, H, \tau_k \rangle \in \mathscr{C}$ with $Eph\prime \neq Eph$, the server builds a *contact tuple* $\hat{T} = \langle Eph\prime, H, \tau_k, (\rho, \theta) \rangle$ that includes the relative coordinates of the infected user.

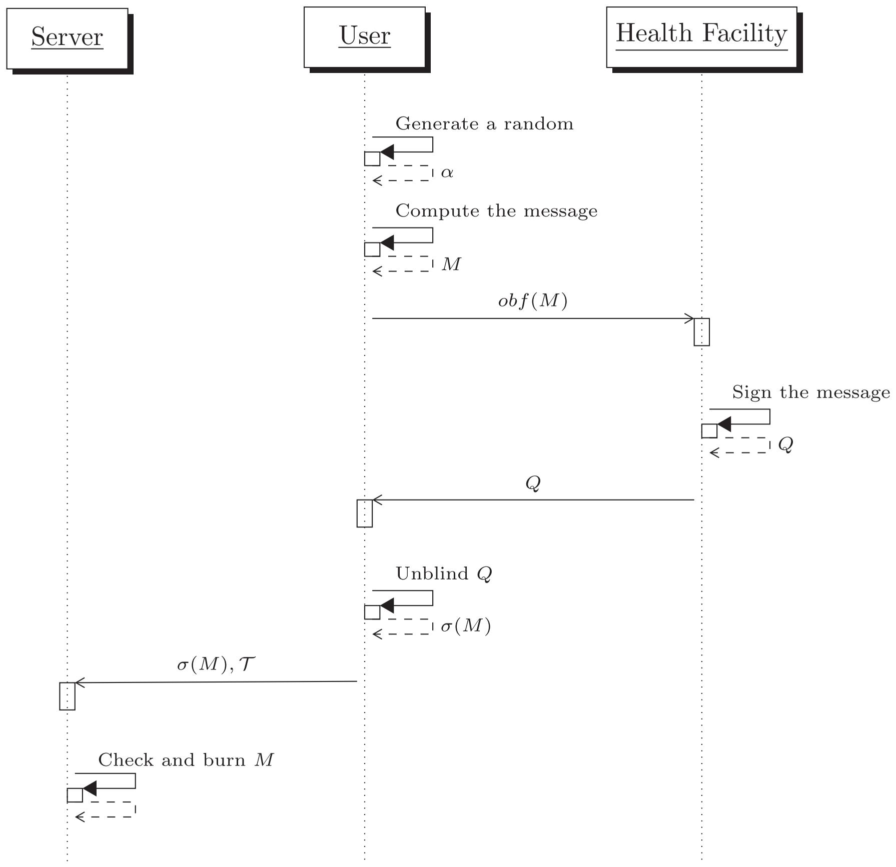


Fig. 4. Sequence diagram of the infection-reporting phase.

The set of all the contact tuples is included into a single file, called $\mathscr{CT}$ (*contact file*) collecting all the information about the infection reports regarding other infected users. The file is transmitted in broadcast periodically, and then erased in order to collect the new infection reports.

The contact-detection phase is summarized in the sequence diagram of Fig. 5.

3.6. Risk computation

After receiving the contact file $\mathscr{CT}$, each user U_y performs the following actions.

For each contact tuple $\hat{T} = \langle Eph\prime, H, \tau_k, (\rho, \theta)\rangle \in \mathscr{CT}, U_y$ retrieves, if any, a tuple $T = \langle Eph\prime, H, (\rho\prime, \theta\prime), t\rangle$, i.e., a tuple with the same ephemeral identifier (and the same digest H). Then, it generates the pair $R = (d, \tau_k)$, where $d = \sqrt{\rho^2 + \rho\prime^2 - 2\rho\rho\prime cos(\theta - \theta\prime)}$. In words, R represents a contact between U_y and an infected user in the time slot τ_k at a distance d. We denote by $\mathscr{R}$, the set of all the pairs computed by U_y. The risk value r for U_y is a function of $\mathscr{R}$.

We do not focus on the function r for the computation of the risk level since it depends on several medical factors. We can say that the function increases as the cardinality of $\mathscr{R}$ increases and it decreases as the distances between users increase. Moreover, consecutive time slots represent a prolonged contact, then a higher risk.

We just remark that all the information typically used to evaluate the risk in digital contact tracing solutions is available also in our model.

The sequence of actions performed in this phase is reported in the sequence diagram of Fig. 6.

4. Implementation and Experiments

Through this section, we describe the experimental environment, the prototype we developed to validate our proposal, and the experimental procedure we followed.

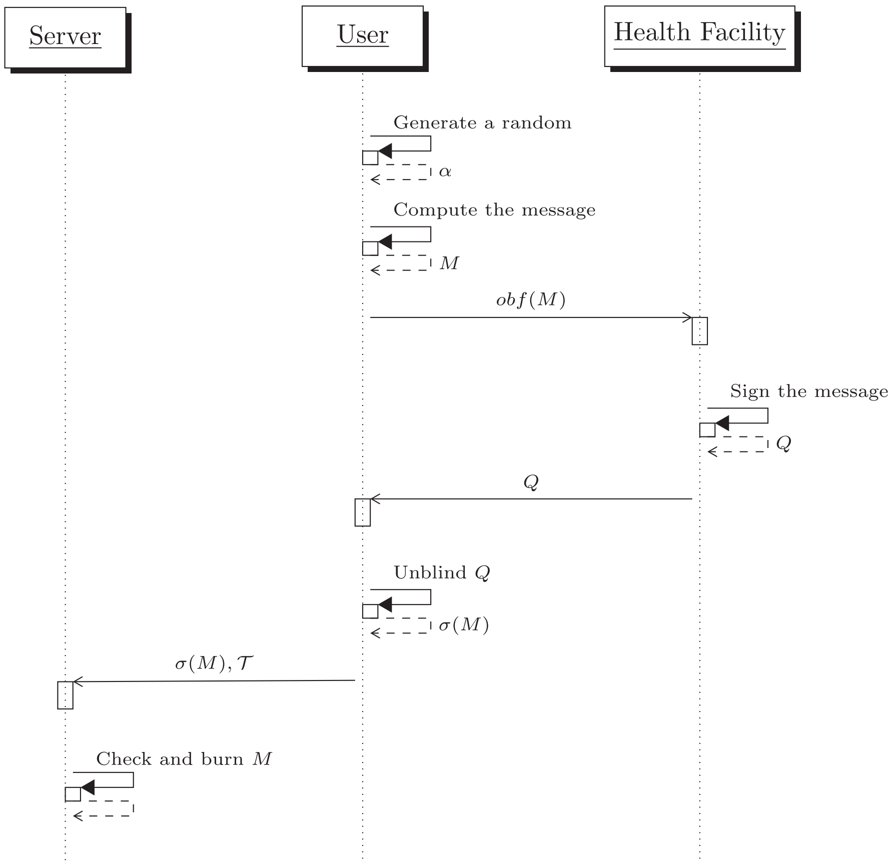


Fig. 5. Sequence diagram of the contact-detection phase.

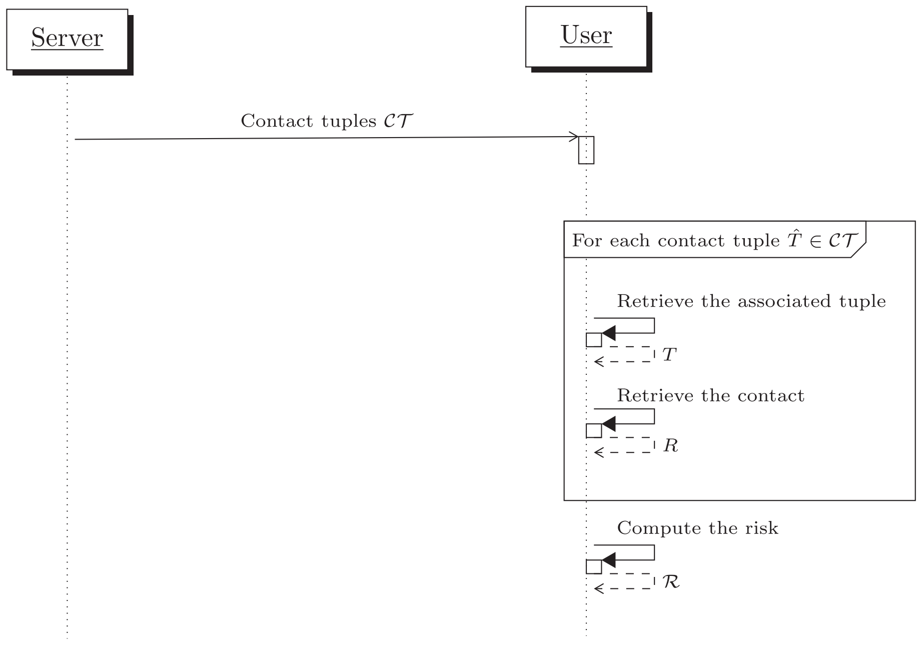

Server
User
Contact tuples $\mathcal{CT}$
For each contact tuple $\hat{T} \in \mathcal{CT}$
Retrieve the associated tuple
T
Retrieve the contact
R
Compute the risk
$\mathcal{R}$

Fig. 6. Sequence diagram of the risk-computation activity.

4.1. Experimental environment

The aim of our experiments is to analyze the performance of our protocol. The experiments were conducted by using a prototype that implements our solution whose description is given in Section 4.2. Moreover, to have realistic results, we leveraged the Thomas Brinkhoff data generator [47] to simulate n users moving in a city. As a city, we choose the default city of the simulator, which is Oldenburg (Germany). The Thomas Brinkhoff data generator uses a discrete-time model to simulate different moving patterns of objects (belonging to different classes) in real networks. To test the growth of clients we developed a java multi-threaded application in which each thread implements the part of the (android) client-side prototype responsible for the communication with the server.

Our experiments were performed by employing a personal computer equipped with an 1.8 GHz Intel i7-8850 CPU and 16 GB of RAM.

Before discussing the experimental procedure, we describe in the next section how we implemented the prototype.

4.2. Implementation

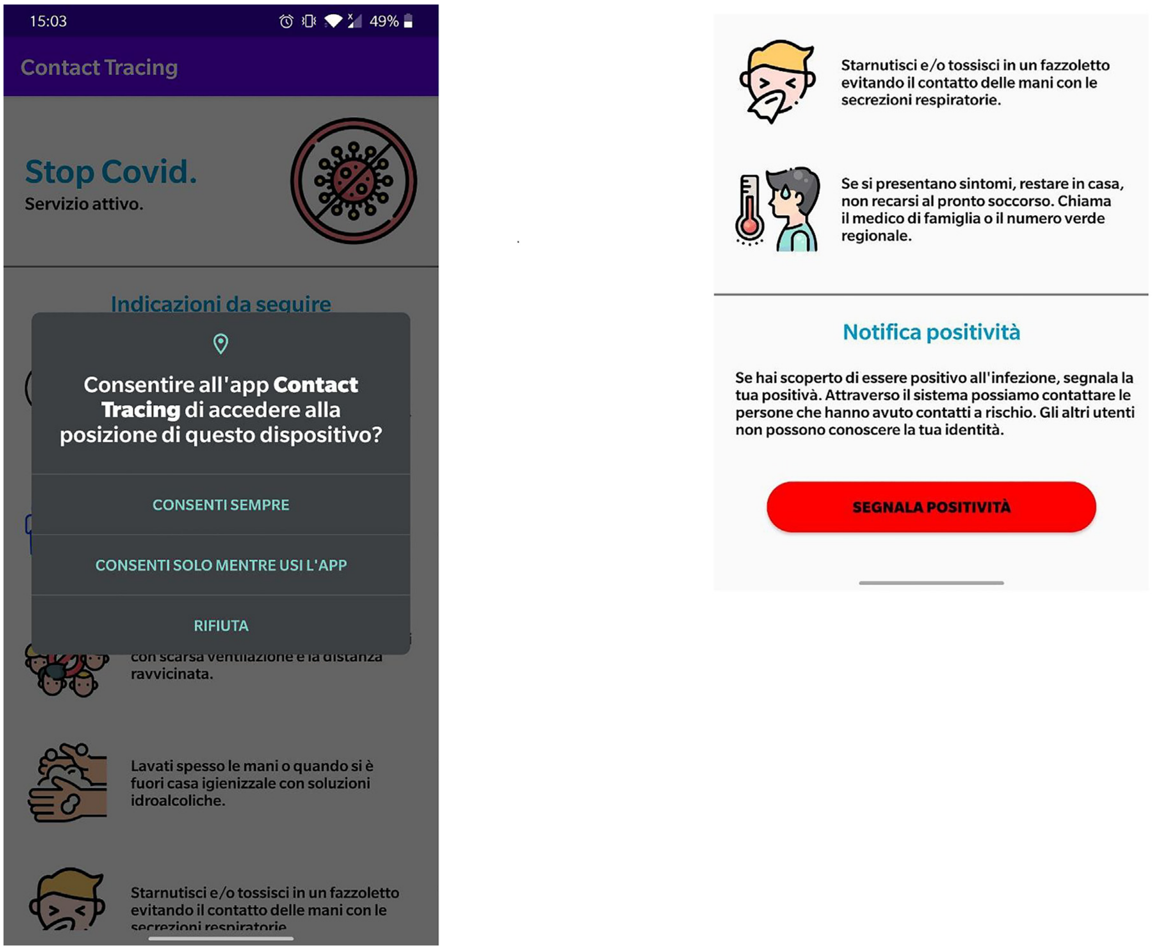
In this section, we describe the prototype we developed, which implements the main functionalities of ZE2-P3T. The source code of this implementation is available at https://github.com/vincenzodeangelisrc/ZE2-P3T. Compared with the previously presented protocol, the prototype includes some optimizations client-side (see below for detail), that should be disabled in order to refer to the security features shown in Section 6.

The implemented modules are three: (1) *Client*, (2) *Health Facility*, and (3) *Server*.

4.2.1. Client module

The Client module is a mobile Android application written in Java. In Fig. 7, we report two screenshots of the Welcome page. It is written in Italian.

The application includes two Activities. The `Welcome Activity` has only graphical functions and shows some alerting messages. The core of the module is the `Main Activity` that implements the ZE2-P3T protocol (i.e., generation of the ephemeral IDs, retrieval of the centroids, and sending information to the server). The retrieval of the salt from the TSP is simulated. A full implementation of this feature would require the collaboration of a telephone service provider to set the technological detail regarding the transmission of the salt. We plan to implement it as future work, by involving in our experimentation also industrial partners (this is the reason why, at moment, the interface of the application is developed in Italian). The application also includes a `Foreground Service` that allows the users to note that some operations are performed by the app and consume resources of the system.



Stop Covid.
Servizio attivo.

Indicazioni da seguire

Consentire all'app **Contact Tracing** di accedere alla posizione di questo dispositivo?

con scarsa ventilazione e la distanza ravvicinata.

Lavati spesso le mani o quando si è fuori casa igienizzale con soluzioni idroalcoliche.

Starnutisci e/o tossisci in un fazzoletto evitando il contatto delle mani con le secrezioni respiratorie.

Starnutisci e/o tossisci in un fazzoletto evitando il contatto delle mani con le secrezioni respiratorie.

Se si presentano sintomi, restare in casa, non recarsi al pronto soccorso. Chiama il medico di famiglia o il numero verde regionale.

Notifica positività

Se hai scoperto di essere positivo all'infezione, segnala la tua positivà. Attraverso il sistema possiamo contattare le persone che hanno avuto contatti a rischio. Gli altri utenti non possono conoscere la tua identità.

Fig. 7. Screenshots of the Welcome page.

The `Notification Exposure` component implements the client-side activity of the infection-reporting phase described in Section 3.4. The generation of the random α of 768 bits is performed through the class `SecureRandom` and the selected hash algorithm is SHA-256.

As mentioned earlier, the code of the Client module includes some optional optimization features, differentiating the implementation from the protocol presented in Section 3. These features are based on BLE and aims at reducing the effort performed client-side and server-side. In particular, the idea is to use BLE to detect the proximity of the users and to send data to the server only in this case (instead of the periodical sending). Even though some information is exchanged through BLE between users, they do not include the ephemeral IDs (thus, preserving, in principle, the security features offered by ZE2-P3T). However, no formal analysis is conducted on this aspect (planned as a possible extension of the protocol and future work). Therefore, in the current state, the BLE component should be disabled to have security guarantees. Certainly, enabling the BLE interface exposes our application to attacks such as those reported in [13].

4.2.2. Health Facility module

The HF module is a web application implemented in Java by relying on the Servlet technology. The servlet `BlindSignature` simply receives the message to sign from the client module, invokes the method `calculateSignatureOfMessage`, and returns the signed message to the client module.

The core of the HF module is the `Health Facility` class that contains the method `calculateSignatureOfMessage`. It implements the standard RSA blind signature. For the final implementation of the system, we plan to exploit tested cryptographic libraries.

4.2.3. Server module

Also the Server module is a web application implemented through the Servlet technology.

It includes two servlets (`RestrictedTuplesReport` and `InfectionReporting`) and six classes (`Tuple`, `Restricted-Tuple`, `ClusteredTuple`, `ContactTuple`,`Key`, and `ContactClusters`). The first four classes, as the names reveal, represent the four types of tuples present in our protocol. The class `Key` is an accessory class representing the key of a HashMap contained in the `ContactClusters` class. Such a key is composed of a time slot and a digest. The HashMap is of the type `ConcurrentHashMap` that is thread-safe, so that multiple parallel requests coming from the clients can be managed. This HashMap associates each `Key` with a Set of `ContactTuple` representing a contact cluster as defined in Section 3.3.

Also the Set of Contact Tuples is obtained as a particular implementation of a `ConcurrentHashMap` to be thread-safe.

The core of this module is represented by the two servlets. The servlet `RestrictedTuplesReport` implements the server-side activity of Section 3.3. It is called by the users to provide the server with the restricted tuples sent periodically. From these tuples, the servlet retrieves the contact tuples and fills the HashMap of the class `contactClusters`. In addition, periodically, at each time slot, the HashMap is emptied and its content is stored in a back-end database. In the provided implementation, we rely on MySQL 8.0. As an optimization, if a Contact Cluster contains just a restricted tuple (i.e., there is a single user in a microcell in a given time slot), it is not stored in the database.

The servlet `InfectionReporting` implements the server-side activity of Section 3.4 and Section 3.5. This servlet receives the non-restricted tuples by the positive users and detects the contacts by the database. Then all the contacts (along with those produced by other infected users) are stored in the contact file that will be downloaded periodically by the users.

4.3. Experimental procedure

In this section, we describe the experimental procedure we followed to test our protocol. The experiments aimed at validating the feasibility of our proposal. Observe that, client-side, the smartphone has to perform very cheap autonomous operations. Indeed, the smartphone, during the standard client-side activity, is required to perform: (1) detection of (at most) two centroids and salts; (2) computation of (at most) 4 digests; and (3) sending of (at most) 4 restricted tuples to the server.

Also the number of operations performed during the infection reporting phase is small. It generates a random number, computes an obfuscated message for the blind signature, unblinds the message returned by the health facility, and uploads some tuples to the server. Regarding the contact detection phase, after downloading the contact file, the client has to find its ephemeral IDs in this file to find.

Obviously, considering the standard computational capabilities of current smartphones, the above operations are definitely feasible.

A similar consideration can be done for the operations performed by HF.

Therefore, the experiments focused on the tasks performed server-side. Indeed, being our approach centralized, the server-side component of the computation could represent, in principle, an issue of the solution, also because the application should be designed for a huge number of clients. Through the performed experiments, we show that, even with limited computational resources (i.e., a standard PC), the computation can be performed also for a high number of users. On the other hand, for how they are designed, the experiments can be also viewed as an operational method to size the server-side computational and storage resources in function of the operation conditions.

In our experiments, we considered an observation window of 14 days. This value represents the retention time (see Section 3) after which the data produced by users are discarded. Then, we set the time of the time slots τ (sending period of the users) equal to 5 min, the same as DP-3T/GAEN. The size of the microcells in which the territory is partitioned is $2x2m^2$.

Two types of experiments have been performed.

In the first experiment, we analyzed the server-side computation during the standard client-side and server-side activities.

In particular, we simulated a number n of users, moving in the city of Oldengurb through the Thomas Brinkhoff generator, who, at each time slot, transmit their restricted tuples to the server.

A graphical representation of the city and users is depicted in Fig. 8.

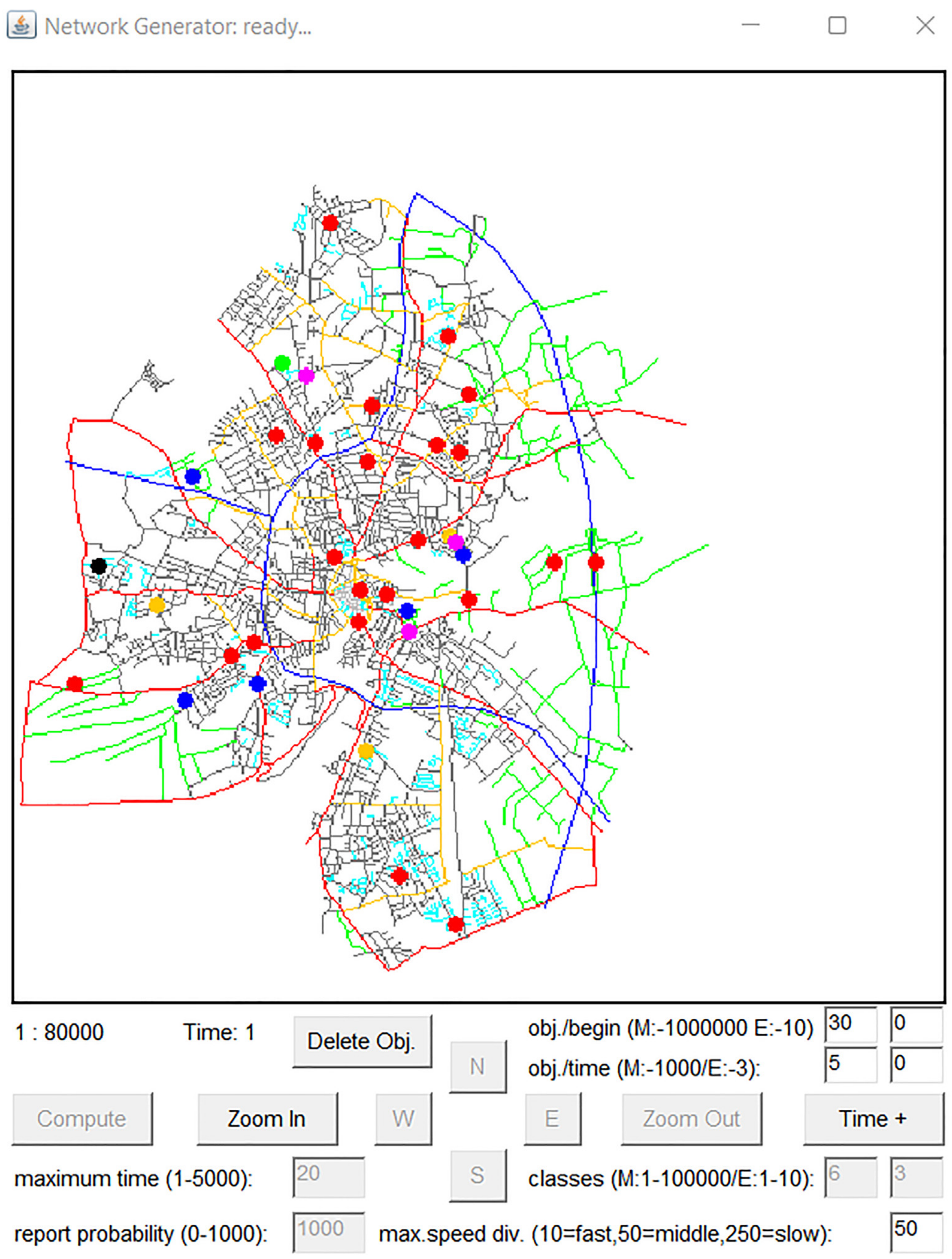
Due to the limited computation power of the employed PC, we considered n users with n ranging from 100000 to 600000. However, we show in Section 5, both analytically and experimentally, that the processing time increases linearly with n. Then, it suffices to employ a multi-core CPU to easily manage millions of users.

Through the multi-threaded application, we generated every 5 min (size of the time slot), the parallel requests coming from the n users and measured the processing time server-side.

In the second experiment, we analyzed the server-side computation time during the infection reporting phase. Specifically, we varied the number I of infection reporting requests performed by the users positive for COVID-19 in a given interval of time T and measured the processing time server-side. For uniformity, we consider I as the number of infection reporting requests performed in 5 min.

Moreover, we evaluate the number of bytes required to maintain the data in: (1) the central memory of the server, (2) the secondary memory of the server, (3) the contact file.

(3) is the most critical factor, since this file is downloaded periodically (e.g., daily) by the users. Then, we proposed an improvement applicable when the exact polar coordinates of the positive users are not stored in the file. The idea is to store an approximate version of these coordinates. In particular, we split each microcell into t subcells and include in the contact file just the identifiers of the subcells in which the positive users are located.



1 : 80000 Time: 1

obj./begin (M:-1000000 E:-10) 30 0

obj./time (M:-1000/E:-3): 5 0

maximum time (1-5000): 20

classes (M:1-100000/E:1-10): 6 3

report probability (0-1000): 1000

max.speed div. (10=fast,50=middle,250=slow): 50

Fig. 8. Users distribution in the Oldengurb city.

This way, the ephemeral identifiers of the contact tuples are concatenated with the identifiers of the subcells and can be efficiently stored in a Bloom filter [48]. A Bloom filter is a probabilistic data structure used to test whether an element is a member of a set. The probabilistic nature of the data structure regards only false positives, which are possible, Instead, false negatives are not possible. This property makes Bloom filters suitable for our application. Indeed, it works in favor of safety. However, to minimize false positives, we set the false positive probability to a very low value, namely 10^{-6}. This, in practice, means that there are no false positives. With Bloom filters, we drastically reduce the size of the contact file.

Client-side, when the contact file is downloaded, the smartphone can check if one of its ephemeral identifiers is in the contact file by testing all the t subcell identifiers. In our setting, we choose $t = 16$, i.e, we split the microcell in 4x4 subcells each of side 50cm. This value allows us not to pay a relevant price client-side (16 searches in the Bloom filter per ephemeral).

Once the subcell is retrieved, the client computes the distance between the locally stored coordinates and the center of the subcell. Observe that given the small size of the subcell, the computed distance is quite accurate.

Another point to take into consideration is whether the use of the Bloom filter introduces a server-side computational overhead during the infection reporting phase. Then, in the experimental procedure, we repeated the second experiment by including the Bloom filter. As we will see in the next section, the adoption of Boom filters does not introduce drawbacks server-side.

5. Results

The purpose of this section is twofold. First, we perform a (space and time) cost analysis of the server-side tasks considered in Section 4.3. The cost analysis allows us to foresee the scalability of our solution, since, basically, we obtain costs that grow linearly with the size of the input. Through experiments, we give the exact costs in the considered experimental setting, which confirm the above prediction.

5.1. Cost analysis

We start by considering the time required server-side to process the requests coming from the clients during the client-side and server-side activities (first experiment of Section 4.3).

Each τ seconds, each client provides the server with at most 4 restricted tuples. In the proposed implementation, each tuple is inserted in the HashMap with constant cost. By considering n users, there are (at most) $4n$ insertions per time slot. Finally, the content of the HashMap is stored in the database. We set a Hash index on the `Digest` attribute of the restricted tuple, so that both the insertion and the search of a tuple require a constant cost. Then, the time to store the HashMap in the secondary memory is proportional to the size of the HashMap (i.e., at most $4n$). We can conclude that the computational cost required server-side per time slot τ is $O(n)$.

Clearly, as we will discuss in the next section, the transfer from the HashMap to the database is the dominant operation. Furthermore, observe that this linear growth occurs until the required time is less than the time slot size (i.e., 5 min). Indeed, above this threshold, a new burst of requests reaches the server before it ends to process the previous burst and then, the performance degrades. Also this point is discussed in the next section.

Consider now the processing time required during the infection reporting phase.

When an infection reporting request reaches the server, it performs a query on the database to find the contact tuples associated with a given digest and time slot.

To improve the performance, we included the time slot in the digest itself, so that the search can be done efficiently by digest. Being the Hash index built on this attribute, the search cost of a single digest is constant.

We recall that an infected user reports the tuples which they generate in the previous 14 days. This number of tuples is constant and independent of the total number of users.

After performing the search, the query result is stored in a file (or in the Bloom filter). Even though, in principle, this cost depends on the size of the query result, we can say that this number is very small (almost constant) compared with the total number of users. Indeed, it depends on the number of users simultaneously present in a microcell of $2 \times 2\ m^2$, which is obviously very small.

Then, the search cost of a single infection reporting is $O(1)$. Then, given I infection reporting requests per time slot, the time required by the server to solve them is $O(I)$. We observe that, also by enabling the Bloom filter, the insertion cost of a single ephemeral is constant. Then, asymptotically, the cost remains $O(I)$. Experimentally, we observed a small difference by comparing the actual costs of our solution with and without the Bloom filter, respectively.

Consider now the space required to store the data of our protocol. To be general, we denote by X the observation time (i.e, 14 days) in seconds, and by τ the size of the time slot in seconds. Moreover, we denote by k the average number of users simultaneously present in a microcell.

First, in the central memory, the server loads the Hash Table, which is emptied at each time slot τ, and the contact file, which we assume is downloaded (and then emptied) daily by the users.

Regarding the size of the Hash Table, consider that, every τ seconds, at most $4 \cdot n$restricted tuples have to be inserted. If we denote by x the size (in bytes) of a single restricted tuple, the maximum space required for the Hash Table is $4 \cdot n \cdot x$ bytes.

Consider now the storage space required in the secondary memory. Since it contains the tuples stored in the last 14 days (X seconds) by the users, this space will be $4 \cdot n \cdot \frac{X}{\tau} \cdot x$. Actually, the tuples generated by users who do not share any microcell with other users, are not stored. Then, the above analysis represents the worst case.

Finally, consider the contact file. The size of this file depends on the number of contacts of the users in the last 14 days. We start by considering that a single positive user reports all the tuples they generate in X seconds i.e., $4 \cdot \frac{X}{\tau}$. For each tuple, other $k - 1$ (clustered) tuples are retrieved by the secondary memory.

Then, the contact file will contain $4 \cdot \frac{X}{\tau} \cdot (k-1)$ contact tuples per positive user.

If we denote by y the size of a contact tuple and by i the average number of positive users (per day), we have that the size of the contact file will be: $4 \cdot \frac{X}{\tau} \cdot (k-1) \cdot i \cdot y$.

If we adopt the Bloom filter, the size of the file can be computed through the standard formulas. In particular, given the (maximum) number of elements to be included in the filter and the probability of false positives, we can obtain the size in bytes of the filter (and the optimal number of hash functions to set).

Specifically, given d elements to include in the filter and a false positive probability p, the size of the filter will be [48]: $-\frac{d \cdot ln(p)}{(ln2)^2}$.

In our setting, $d = 4 \cdot \frac{X}{\tau} \cdot (k-1) \cdot i$.

As we will see in the next section, in practice, the size of the contact file is drastically reduced by the use of the Bloom filter.

5.2. Experimental results

We now describe the results we obtained through our experiments.

The plot in Fig. 9 shows the computational time required by the server to process the requests coming from n users in a time slot.

Coherently with the previous analysis, we observe that the time increases linearly with the number of users. This happens until the time reaches the time slot size (green line in the plot). After this threshold, non-linear effects are present, due to the fact that a new burst of tuples reaches the server before it processes entirely the previous burst.

During the experiment, we observed that the transfer time of the Hash table from the central memory to the secondary memory represents about 90% of the total time required by the server during this phase. The other 10% of the time is spent processing the incoming requests and storing the tuples in the Hash Table.

This plot also shows that the personal computer used for the experiments, despite its standard capabilities, is able to manage up to 300000 users. Then, by considering the linear increase, a dedicated platform (possibly in the cloud) can easily manage million of users. Actually, the above experiment, thanks to threshold analysis, provides a method to size the resources server-side.

Consider now the infection reporting phase whose results are reported in the plots of Fig. 10.

Therein, we show the computational time required by the server during an infection reporting when we enable the Bloom filter and when we disable it.

The server computational time is reported as a function of the number of infection reporting requests I arriving in a time slot. In accordance with the cost analysis, we have that until the computational capacity of the employed PC is not saturated, the growth of the time is linear in I with and without the Bloom filter.

The computer used for experiments is able to manage up to 2000 requests per time slot. Observe that, the introduction of the Bloom filter (blue line) has a minimal impact on the performance until this value is reached.

We want to highlight that 2000 requests per time slot correspond to 576000 infected users per day. By relying on the aggregate data on the pandemic available at https://github.com/CSSEGISandData/COVID-19 [49], we observe that in the peak period (January 2022), by considering just the countries with the most daily cases (in relation to the population), we have a daily rate of positively of 0.2%. This means that only a personal computer is able to manage about 288 million users. Thus, the infection reporting phase is definitively not critical.

We conclude by estimating the actual space required by our solution.

Considering the Hash Table, since each restricted tuple includes a 32-byte digest, an 8-byte ephemeral identifier, and a 16-byte time slot, the size of each restricted tuple is x = 56 bytes.

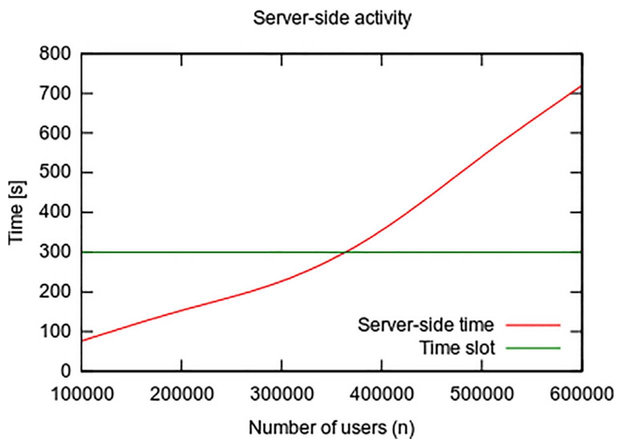


Fig. 9. Computational time required for the server-side activity.

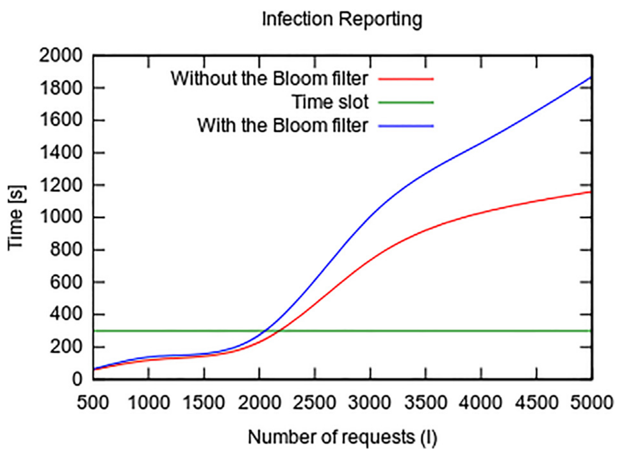


Fig. 10. Computational time required during the infection reporting phase.

For $n = 10$ million users, the Hash Table requires a space of $4 \cdot n \cdot x = 2.24$ GB that perfectly fits in the central memory of a standard personal computer. Clearly, we have to add the size of the contact file that resides in the central memory too.

Regarding the storage required in the secondary memory, it is the same as the central memory multiplied by $\frac{X}{\tau} = 4032$. This corresponds, for $n = 10$ million, to 9.03 TB (again, a feasible value for standard storage devices).

The main bottleneck is represented by the size of the contact file. Indeed, the contact file should be downloaded periodically by the users. If we do not use the Bloom filter (to include the exact polar coordinates of the infected users), we recall that the size of the contact file is $4 \cdot \frac{X}{\tau} \cdot (k-1) \cdot i \cdot y$.

We set $k = 1.5$ users per microcell simultaneously (it means that half the time a user is alone in a microcell and the other half is with another user.. Then, given $i = 0.002 \cdot n$ and $y = 16$ bytes (we include in the contact file just the ephemeral ID of 8 bytes and the two coordinates of 4 bytes), we have that for 10 million users, the size of the contact file will be of 2.62 GB. Even though it can be stored in the central memory, it is unrealistic that a user downloads such a size each day.

Then, without the adoption of the Bloom filter, a realistic number of users that can be supported is 1 million leading to a size of the contact file of 262 MB.

If we introduce the Bloom filter, we reduce this size. In particular, given a false positive report probability of 10^{-6}, we obtain that the size of the contact file for $n = 1$ million is 57,51 MB and the size of the contact file for $n = 10$ million is 575,1 MB.

These results are summarized in the table reported in Fig. 11.

6. Security analysis

In this section, we provide a security (comparative) analysis of ZE2-P3T and DP-3T protocols. We assume that the reader is familiar with the protocol DP-3T and its designs, namely Low-Cost and Unlinkable. For detail, the reader can refer to [5].

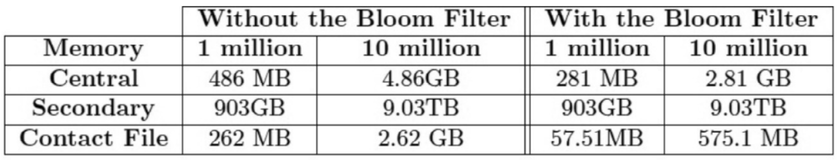
We start by introducing the following assumptions.

A0 *In the adversary model, TTPs can play as attackers.*

A1 *In the adversary model, two independent TTPs cannot collude.*

A2 *A sybil attack proliferating fake users even with a passive role cannot require any action from the side of any non-attacker TTP.*

Observe that Assumption **A1** implies that TSP and server (in ZE2-P3T) cannot collude. Indeed, they are independent TTPs. We emphasize the fact that, in order to be the Assumption **A1** applicable, the selection of the two parts must ensure real independence from each other.



	Without the Bloom Filter		With the Bloom Filter	
Memory	1 million	10 million	1 million	10 million
Central	486 MB	4.86GB	281 MB	2.81 GB
Secondary	903GB	9.03TB	903GB	9.03TB
Contact File	262 MB	2.62 GB	57.51MB	575.1 MB

Fig. 11. Space cost analysis.

On the other hand, considering that we can expect that the server is managed by a government institution, it is not realistic that a government colludes with the major telephone company of the country without risking that this attempt is discovered and results in dramatic consequences for the public opinion.

To further show that Assumption **A1** conforms to the facts, observe that, while assuming that a single entity is an attacker might just mean that one or more insiders (i.e., malicious employees) break the rules, the collusion of two independent entities would require the malicious coordination between the entities, and the consequent continuous communication. Likely, some top managers (maybe CEOs or CISOs) of the two organizations should be involved. It is clear that when the scope of the malicious activity goes outside the perimeter of a single organization and crosses two independent organizations it is much more difficult to keep the activity secret for a long time. On the other hand, due to the high-level profile of the organizations we are considering in our solution, the risk (in terms of impact) of the disclosure (also *ex-post*) of malicious activities, would likely lead the involved organizations to default, and very serious legal consequences.

Assumption **A1** is not applicable to DP-3T because the two TTPs (HF and server) are not independent. Then, in our adversary model, for both protocols, the collusion between HF and the server is allowed.

Assumption **A2** means that, in ZE2-P3T, only a limited number of malicious SIM cards can be activated provided that TSP is not an attacker. The assumption is quite realistic. Indeed, without the collaboration of the TSP, for an adversary, it would be not simple and very dangerous from a legal point of view to obtain many fake SIM cards. At least in the countries in which crime is contrasted, to enable a SIM card, you should register a real-life person with an ID document. Even though a black market with stolen/fake SIM cards could exist, the TSP immediately would detect their massive utilization, thus blocking them.

For both protocols, a huge number of sybil agents can be placed without requiring any registration. For example, by using Bluetooth antennas spread over the territory.

The possible adversaries and their capabilities are:

- The user: they are the only party that operates and captures information available on the territory.
- The server S: it operates only on the basis of the information received from users and HF.
- The health facility HF: it operates only on the basis of the information provided by the user performing a test.

Observe that TSP is not included among possible adversaries because, according to Assumption **A1**, TSP could collude only with users. However, even in this case, it can access only tuples encrypted by the users with the public key of the server. Therefore, it does not have any information associable with contacts.

Now, we formalize all relevant compromises of protocols invalidating privacy and integrity requirements. These implicitly define the security properties of our security model. It is worth noting that we do not consider two kinds of compromises. The first type refers to denial of service compromises. The second type refers to *enumeration attacks* [14]. Enumeration attacks allow a malicious user to estimate the number of positive users. We do not consider this kind of compromise, because it does not appear relevant in terms of impact (indeed, in the literature, no emphasis is given to this attack).

Preliminary, we need to define the *threat-model attributes*, which are:`Attacker_Setting`, `Target`, `Range`.

For each attribute, we also define an order among its values, capturing effort or effect degrees of the compromise.

The domain of `Attacker_Setting` is $\mathscr{A} = \{OU, SU, HU, FC\}$, where

- *OU* denotes the case in which the attacker is a user not colluding with any other party;
- *SU* denotes the case in which the attacker colludes with S.
- *HU* denotes the case in which the attacker colludes with HF.
- *FC* (which stands for full collusion) denotes the case in which the attacker colludes with S and HF.

We can introduce an order relation into $\mathscr{A}$ as follows (defined in terms of transitive reduction):
$FC \preceq_A SU, FC \preceq_A HU$, $SU \preceq_A OU, HU \preceq_A OU$.
Observe that $\preceq_A$ is partial.
Intuitively, $\preceq_A$ captures the fact that the higher the number of colluding entities, the higher the required effort is.
The domain of `Target` is $\mathscr{T} = \{GV, PV\}$, where

- *GV* denotes the case in which the victim is a generic user.
- *PV* denotes the case in which the victim is a user who is tested positive for infection and reported this information to the server.

We can introduce an order relation into $\mathscr{T}$ as follows: $PV \preceq_T GV$.

Clearly, if the target is a generic user, the effect of a compromise is worse than the case in which the target is a positive user.

The domain of `Range` is $\mathscr{R} = \{SC, IC\}$, where

- *SC* denotes the case in which the compromise can realistically scale to many victims (i.e., the attacker can increase the number of victims at will with no strong barriers).

- *IC* denotes the case in which the compromise is feasible only if directed to a few victims.

We can introduce an order relation into $\mathscr{R}$ as follows: $IC \preceq_R SC$.

Similarly to `Target`, the effect of a compromise is worse if it involves many victims.

Fig. 12 summarizes the order relations defined above.

Remark 1. We observe that the three order relations above defined basically refer to set-inclusion/*is-A* hierarchies. In particular, if the attacker performs as a full collusion (FC), it can impersonate the server (SU) or the health facility (HF). Moreover, if it performs as SU or HF, it can operate as a standard user (OU). On the other hand, a positive user (PV) is a generic user (GV), and the set of victims involved in a targeted attack (IC) is not indiscriminate as in a massive attack (SC).

We focus our security analysis on the following three *Compromises*. Each compromise is defined over $(\mathscr{A}, \mathscr{T}, \mathscr{R})$.

From now on, since the ephemeral identifiers generated by the users in ZE2-P3T are equivalent to the ephemeral identifiers of DP-3T, we use the term *ephemeral ID* to refer to both these pseudonym identities.

Compromise 1: Linkage of ephemeral identifiers.

This compromise occurs when the attacker finds out that a given number of ephemeral IDs belong to the same user (not requiring that the real identities are known).

Compromise 2: Contact forgery.

This compromise occurs when the attacker is able to forge a contact that never happened between two users with the scope of generating a false alarm.

Remark 1. We remark that, as ephemeral IDs are necessarily pseudonyms unlinkable with real identities, impersonation with a colluding attacker is intrinsically possible both in DP-3T and in ZE2-P3T. Therefore, we do not consider a compromise of type 2 any fake contact forged by an attacker by communicating ephemeral IDs of the colluding attacker in the user-side activity or in the infection-reporting phase.

Compromise 3: Linkage between an ephemeral ID and extra information.

This compromise occurs when the attacker is able to link an ephemeral ID of a user with any extra information not provided by the user.

Compromises may happen in different configurations of the attributes of the threat model. A configuration is called *setting*.

Definition 6.1. A *Setting S* is a 3-tuple $s = (a, t, r)$, where $a \in \mathscr{A}, t \in \mathscr{T}$, and $r \in \mathscr{R}$. We denote by $\mathscr{S} = \mathscr{A} \times \mathscr{T} \times \mathscr{R}$ the set of all possible settings.

We have $4 \cdot 2 \cdot 2 = 16$ possible settings.

The partial order relations defined for attributes induces a partial order relation among settings.

Definition 6.2. Given two settings $S_1 = (a_1, t_1, r_1)$ and $S_2 = (a_2, t_2, r_2)$, we say that $S_1 \preceq_S S_2$ if $a_1 \preceq_A a_2 \wedge t_1 \preceq_T t_2 \wedge r_1 \preceq_R r_2$.

The order relation on $\mathscr{S}$ is represented in Fig. 13.

In the next definition, we give the notion of *vulnerability of a protocol* to a given compromise in a given setting.

Definition 6.3. Given a setting *S* and a compromise *C*, we say that a protocol *P* is *vulnerable* to *C* in *S* if there exists an attack in the setting *S* that leads to the compromise *C*.

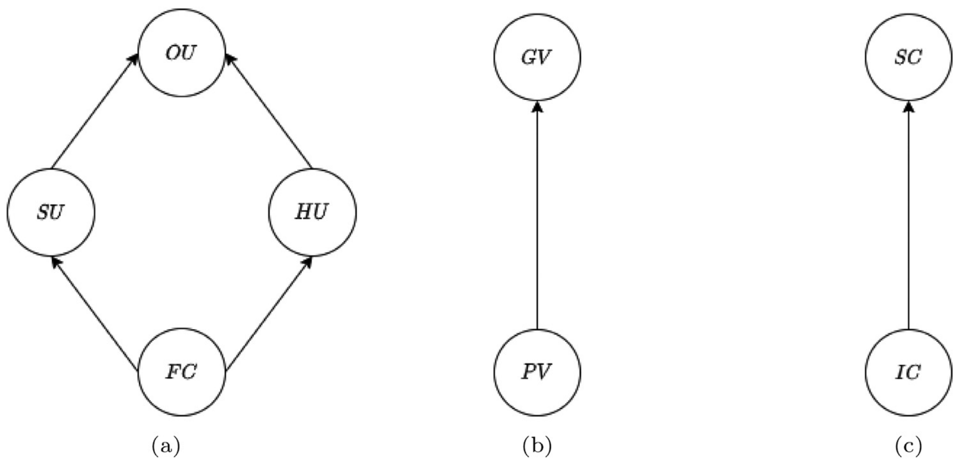


Fig. 12. Order relations on $\mathscr{A}$ (a), $\mathscr{T}$ (b), and $\mathscr{R}$ (c).

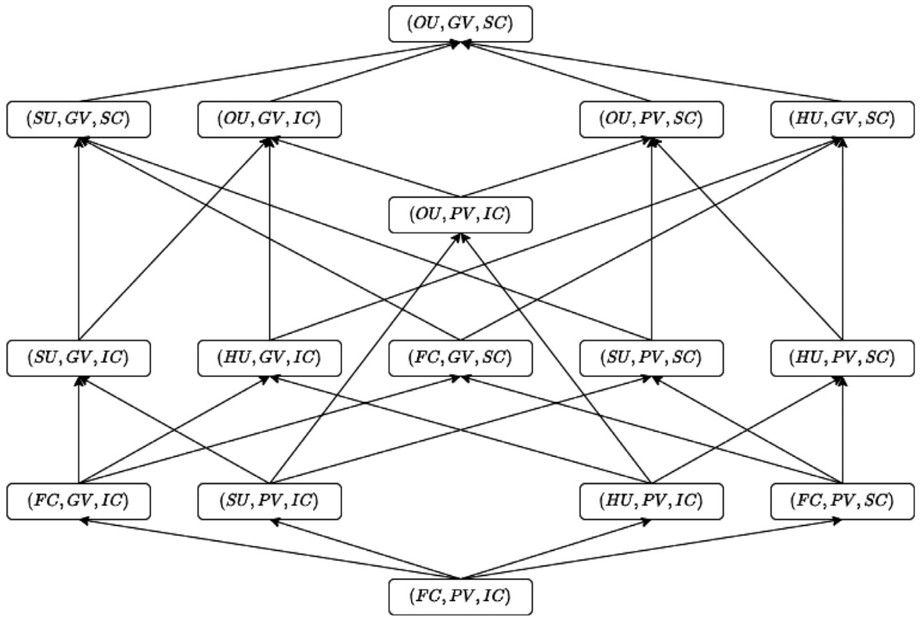


Fig. 13. Order relation on $\mathscr{S}$.

The following lemma shows that the vulnerability of a protocol to a compromise is monotone with respect to the partial order $\preceq_S$. This lemma gives value to the partial order itself, because it allows us to transitively derive the vulnerability occurrence for each considered protocol and, then, compare them. The partial order can be viewed therefore as a scale of comparison.

Lemma 6.1. *Given two settings S_1 and S_2 and a compromise C such that $S_1 \preceq_S S_2$, if a protocol P is vulnerable to C in S_2, then P is vulnerable to C in S_1.*

Proof. The proof immediately follows by **Remark 1** and the fact that $\preceq_S$ is defined on the basis of the three order relations $\preceq_A$, $\preceq_T$, and $\preceq_R$. □

As anticipated, the consequence of the above lemma is that, once we have proven the vulnerability of a protocol to a compromise C in a given setting, we have implicitly proven its *derived* vulnerability to C in a number of other settings. Obviously, to capture the whole set of settings in which a protocol is vulnerable to a compromise, we have to refer to settings that are maximal with respect to $\preceq_S$. In other words, if a protocol is vulnerable to a compromise in a maximal setting, then it is vulnerable to all the settings that are lower than this maximal, i.e., the branch of the partial order with this maximal as a top.

This notion of maximality is formalized by the following three definitions.

Definition 6.4. Given a protocol P, a setting S, and a compromise C, we say that S is a *vulnerability-maximal* of P (on C) if: (i) P is vulnerable to C in S and (ii) there not exists a setting $S_1 \neq S$ in which P is vulnerable to C such that $S \preceq_S S_1$.

Definition 6.5. Given a protocol P and a compromise C, we denote by $\mathscr{V}_{\mathscr{P}}^{\mathscr{C}}$ the set of all the vulnerability-maximals of P on C. $\mathscr{V}_{\mathscr{P}}^{\mathscr{C}}$ is called the *skyline set* of P (on C).

Definition 6.6. Given a protocol P and a compromise C, we denote by $\mathscr{I}_{\mathscr{P}}^{\mathscr{C}}$ the set $\{S \in \mathscr{S} : \exists S\prime \in \mathscr{V}_P^C \text{ s.t. } S \preceq_S S'\}$. $\mathscr{I}_{\mathscr{P}}^{\mathscr{C}}$ is called the *induced set by* $\mathscr{V}_{\mathscr{P}}^{\mathscr{C}}$.

In words, the induced set by $\mathscr{V}_{\mathscr{P}}^{\mathscr{C}}$ is the set of settings less or equal to any maximal occurring in $\mathscr{V}_{\mathscr{P}}^{\mathscr{C}}$.

The next proposition states that all the vulnerabilities of a protocol P to a compromise C are induced by the skyline set of P on C. In other words, while so far we referred to the correctness of the vulnerability implication over the partial order, now we state that this mechanism is complete. Therefore, the vulnerabilities of a protocol are all and only those induced by the maximals, i.e., the skyline set.

Proposition 6.1. *Given a protocol P and a compromise C, P is vulnerable to C in a setting S if and only if $S \in \mathscr{I}_{\mathscr{P}}^{\mathscr{C}}$.*

Proof. (**if-part**) Given a set $S \in \mathscr{I}_{\mathscr{P}}^{\mathscr{C}}$, by definition $S \preceq_S S\prime$ for some $S\prime \in \mathscr{V}_{\mathscr{P}}^{\mathscr{C}}$. Since P is vulnerable to C in $S\prime$, due to Lemma 6.1, P is vulnerable to C in S too.

(**only-if-part**) By contradiction suppose that there exists a setting S_x such that P is vulnerable to C in S_x and $S_x \neg \in \mathscr{I}_{\mathscr{P}}^{\mathscr{C}}$.

Obviously S_x is such that neither $S_x \preceq_S S_y$ nor $S_y \preceq_S S_x$, for any $S_y \in \mathscr{V}_{\mathscr{P}}^{\mathscr{C}}$. Otherwise either S_y would be not a vulnerability-maximal (if $S_y \preceq_S S_x$) or $S_x \in \mathscr{I}_{\mathscr{P}}^{\mathscr{C}}$ (if $S_x \preceq_S S_y$). As $\mathscr{S}$ is finite, this implies that there exists a vulnerability-maximal not included in $\mathscr{V}_{\mathscr{P}}^{\mathscr{C}}$. Therefore, we reached a contradiction. □

So far we compared only settings, thanks to the partial order $\preceq_S$. Now, we want to compare protocols in terms of vulnerability degree to a compromise. We say *degree*, by meaning that our comparison should include some *quantitative* feature, to give a measure of how much a protocol is more vulnerable than another protocol. To do this, we first introduce a *weighted* order relation among protocols.

Definition 6.7. Given two protocols P_1 and P_2, a compromise C, and $k \geq 0$ we say that $P_1 \preceq_k P_2$ (on C), if for each setting $S_x \in \mathscr{V}_{P_2}^{C}$, there exists a setting $S_y \in \mathscr{V}_{P_1}^{C}$ such that $S_x \preceq_S S_y$ and $|\mathscr{I}_{P_1}^{C}| - |\mathscr{I}_{P_2}^{C}| \geq k$.

With the next corollary, we highlight that even though the above definition is based on skyline sets, actually it regards the whole set of settings in which protocols are vulnerable. Again, this is strictly related to the meaning of maximality, allowing us to express properties at the level of the skyline set, instead of the whole set of settings.

Corollary 6.1. *Given two protocols P_1 and P_2, a compromise C, and $k \geq 0$, it holds that $P_1 \preceq_k P_2$ on C if and only if for each setting S in which P_2 is vulnerable to C, P_1 is vulnerable to C in S too, and there exist at least k distinct settings in which P_1 is vulnerable to C and P_2 not.*

Proof. (**if-part**) By Proposition 6.1, $\mathscr{I}_{P_1}^{C}$ contains all the settings in which P_1 is vulnerable to C. By hypothesis and Proposition 6.1, P_1 is vulnerable to C in each setting $S \in \mathscr{I}_{P_2}^{C}$. Therefore, $\mathscr{I}_{P_2}^{C} \subseteq \mathscr{I}_{P_1}^{C}$. As a consequence, $\mathscr{V}_{P_2}^{C} \subseteq \mathscr{I}_{P_1}^{C}$ as $\mathscr{V}_{P_2}^{C} \subseteq \mathscr{I}_{P_2}^{C}$.

Therefore, by the definition of $\mathscr{I}_{P_1}^{C}$, for each setting $S_x \in \mathscr{V}_{P_2}^{C}$ there exists a setting $S_y \in \mathscr{V}_{P_1}^{C}$ such that $S_x \preceq_S S_y$.

Furthermore, since $\mathscr{I}_{P_2}^{C} \subseteq \mathscr{I}_{P_1}^{C}$ and Proposition 6.1, $|\mathscr{I}_{P_1}^{C}| - |\mathscr{I}_{P_2}^{C}|$ is the number of settings in which P_2 is vulnerable to C and P_2 not. Therefore, $|\mathscr{I}_{P_1}^{C}| - |\mathscr{I}_{P_2}^{C}| \geq k$.

We conclude that $P_1 \preceq_k P_2$.

(**only-if-part**) By hypothesis, for each setting $S_x \in \mathscr{V}_{P_2}^{C}$, there exists a setting $S_y \in \mathscr{V}_{P_1}^{C}$ such that $S_x \preceq_S S_y$.

By the definition of $\mathscr{I}_{P_2}^{C}$, for each setting $S_x \in \mathscr{I}_{P_2}^{C}$ there exists a setting $S_y \in \mathscr{V}_{P_2}^{C}$ such that $S_x \preceq_S S_y$.

Then, for each setting $S_x \in \mathscr{I}_{P_2}^{C}$, there exists a setting $S_y \in \mathscr{V}_{P_1}^{C}$ such that $S_x \preceq_S S_y$ and then $\mathscr{I}_{P_2}^{C} \subseteq \mathscr{I}_{P_1}^{C}$.

Therefore, P_1 is vulnerable to C in each S in which P_2 is vulnerable to C. Finally, since $|\mathscr{I}_{P_1}^{C}| - |\mathscr{I}_{P_2}^{C}| \geq k$, there are at least k settings in which P_1 is vulnerable to C and P_2 not. □

We are now ready to compare the security of our protocol with the two designs of DP-3T, with respect to the three compromises. We remark that this comparison implicitly includes also GAEN, because GAEN, from the protocol point of view, consists of DP-3T with Low-Cost design. This is obtained by means of the following three theorems. Therein, we denote by LD the Low-Cost design of DP-3T, by UD the Unlinkable design of DP-3T, and by ZP our protocol. We compare the protocols per compromise.

We start by considering the compromise C_1. The next theorem states that, on the compromise C_1, Low-Cost DP-3T is more vulnerable (with *degree* 6) than our protocol, and Unlinkable DP-3T is more vulnerable (with *degree* 4) than our protocol.

Theorem 6.1. *LD $\preceq_6$ ZP $\wedge$UD $\preceq_4$ ZP on the compromise C_1.*

Proof. First, we prove $\mathscr{V}_{ZP}^{C_1} = \{(SU, GV, IC), (SU, PV, SC)\}$.

To do this, first (i) we prove that ZP is vulnerable to C_1 in (SU, GV, IC) and (SU, PV, SC).

Then (ii), we prove that ZP is not vulnerable to C_1 in any setting greater than (SU, GV, IC) or (SU, PV, SC).

Finally (iii), we prove that ZP is not vulnerable to C_1 in any other incomparable setting w.r.t (SU, GV, IC) and (SU, PV, SC).

We proceed by proving (i). Consider the setting (SU, GV, IC). Regarding the compromise C_1, this setting means that the attacker colludes with the server to link the ephemeral IDs of a limited number of generic users.

Observe that the information received prevent the server from guessing the centroid. Indeed, the salt combined with the centroid in the digest remains unknown to the server. Otherwise, a successful dictionary-based attack would be possible, and then different tuples could be linked (and also included ephemeral IDs) through trajectory-based attacks. However, in the considered setting, this may happen by selecting a victim and detecting the salt sent from the TSP in the zone in which the victim is located. Therefore, ZP is vulnerable to C_1 in (SU, GV, IC).

Regarding (SU, PV, SC) (i.e., the attacker colludes with the server to link the ephemeral IDs of a scalable number of positive users), it is easy to see that ZP is vulnerable to C_1 in (SU, PV, SC). Indeed, each user who tests positive for the disease sends all their ephemeral IDs at once to the server. Therefore, the proof of (i) is concluded.

Now, consider (ii). The settings greater than (SU, GV, IC) or then (SU, PV, SC) are $\{(SU, GV, SC), (OU, GV, IC)$, (OU,GV, SC),$(OU, PV, SC)\}$.

The setting (SU, GV, SC) differs from (SU, GV, IC) only in the fact that the former involves a scalable number of victims. According to the reasoning done for the setting (SU, GV, IC), (SU, GV, SC) would be vulnerable only if a scalable number of salts given by the TSP are known to the server.

Indeed, without TSP salts, the tuples owned by the server cannot be associated with any information not sent by the user. But, receiving a scalable number of salts from different places in the territory is impossible according to Assumption **A2**. Thus, ZP is not vulnerable to C_1 in (SU, GV, SC).

In both the settings (OU, GV, IC) and (OU, PV, SC), the attacker does not collude with the server. The only way for the attacker to obtain the ephemeral IDs of other users is at the end of the contact-detection phase, when the attacker receives a list of ephemeral IDs belonging to users who entered into contact with a positive user. However, the attacker cannot distinguish if some of them belong to the same user or to different users. Therefore, ZP is not vulnerable to C_1 in (OU, GV, IC) and (OU, PV, SC).

Due to Lemma 6.1, since ZP is not vulnerable to C_1 in (SU, GV, SC), then it is not vulnerable to C_1 in (OU, GV, SC). Therefore, the proof of (ii) is concluded.

Finally, consider (iii). The settings incomparable with respect to (SU, GV, IC) and (SU, PV, SC) are $\{(OU, PV, IC), (HU, GV, SC), (HU, GV, IC), (FC, GV, SC), (HU, PV, SC), (HU, PV, IC)\}$.

Consider the setting (FC, GV, SC) i.e., the attacker colludes with the server and HF to link the ephemeral IDs of a scalable number of users. HF does not receive any ephemeral ID from users. Then, the collusion with HF does not provide any advantage to the attacker with respect to the setting (SU, GV, SC). Therefore, ZP is not vulnerable to C_1 in (FC, GV, SC).

Regarding the setting (OU, PV, IC), since the server does not collude, the attacker can obtain the ephemeral IDs only during the contact-detection phase but it is unable to link them. Therefore, ZP is not vulnerable to C_1 in (OU, PV, IC).

Finally, since the collusion with HF does not provide any advantage to the attacker, the settings $(HU, GV, SC), (HU, GV, IC), (HU, PV, SC), (HU, PV, IC)$ are equivalent to the settings $(OU, GV, SC), (OU, GV, IC), (OU, PV, SC)$, (OU, PV, IC), respectively, in which ZP is not vulnerable. This concludes the proof of (iii).

At this point, since $\mathscr{V}^{C_1}_{ZP} = \{(SU, GV, IC), (SU, PV, SC)\}$, the induced set by $\mathscr{V}^{C_1}_{ZP}$ is $\mathscr{I}^{C_1}_{ZP} = \{(SU, GV, IC), (SU, PV, SC)$, $(FC, GV, IC), (SU, PV, IC), (FC, PV, SC), (FC, PV, IC)\}$ and $|\mathscr{I}^{C_1}_{ZP}| = 6$.

Now, consider the protocol LD. To prove that LD $\preceq_6$ ZP, we have to show that (i) there exist $S_1, S_2 \in \mathscr{V}^{C_1}_{LD}$ such that $(SU, GV, IC) \preceq_S S_1$ and $(SU, PV, SC) \preceq_S S_2$, and (ii) $|\mathscr{I}^{C}_{LD}| - |\mathscr{I}^{C}_{ZP}| \geq 6$.

We prove (i).

LD is vulnerable to C_1 in (OU, GV, IC). Indeed, the attacker, without colluding with the server, can link the ephemeral IDs released in broadcast (in plaintext) by a number of target users.

Due to Proposition 6.1, $(OU, GV, IC) \in \mathscr{I}^{C_1}_{LD}$, then there exists $S_1 \in \mathscr{V}^{C_1}_{LD}$ such that $(SU, GV, IC) \preceq_S (OU, GV, IC) \preceq_S S_1$.

LD is also vulnerable to C_1 in (OU, PV, SC). Indeed, each user receives the secret key of each positive user and, locally, computes the ephemeral IDs of the latter.

Due to Proposition 6.1, $(OU, PV, SC) \in \mathscr{I}^{C_1}_{LD}$, then there exists $S_2 \in \mathscr{V}^{C_1}_{LD}$ such that $(SU, PV, SC) \preceq_S (OU, PV, SC) \preceq_S S_2$. Then, the proof of (i) is given.

Now, consider (ii).

The set $\begin{array}{l}\mathscr{I}^{C_1}_{LD} = \{(OU, GV, IC), (OU, PV, SC), (OU, PV, IC), (SU, GV, IC), (HU, GV, IC), (SU, PV, SC), \\ (HU, PV, SC), (FC, GV, IC), (SU, PV, IC), (HU, PV, IC), (FC, PV, SC), (FC, PV, IC)\} \subseteq \mathscr{I}^{C_1}_{LD}\end{array}$, since each element $S \in \mathscr{I}^{C_1}_{LD}$ is such that either $S \preceq_S (OU, GV, IC)$ or $S \preceq_S (OU, PV, SC)$ and $(OU, GV, IC), (OU, PV, SC) \in \mathscr{I}^{C_1}_{LD}$.

Therefore, $|\mathscr{I}^{C_1}_{LD}| \geq |\mathscr{I}^{C_1}_{LD}| = 12$ and then $|\mathscr{I}^{C_1}_{LD}| - |\mathscr{I}^{C_1}_{ZP}| \geq 6$. The proof of (ii) is given.

Finally, we consider the protocol UD. The reasoning is the same as LD.

To prove that UD $\preceq_4$ ZP, we have to show that (i) there exist $S_1, S_2 \in \mathscr{V}^{C_1}_{UD}$ such that $(SU, GV, IC) \preceq_S S_1$ and $(SU, PV, SC) \preceq_S S_2$, and (ii) $|\mathscr{I}^{C}_{UD}| - |\mathscr{I}^{C}_{ZP}| \geq 4$.

We prove (i).

Similarly to LD, UD is vulnerable to C_1 in (OU, GV, IC). Indeed, the attacker, without colluding with the server, can link the ephemeral IDs released in broadcast (in plaintext) by a number of target users.

Due to Proposition 6.1, $(OU, GV, IC) \in \mathscr{I}^{C_1}_{UD}$, then there exists $S_1 \in \mathscr{V}^{C_1}_{UD}$ such that $(SU, GV, IC) \preceq_S (OU, GV, IC) \preceq_S S_1$.

Similarly to ZP, UD is vulnerable to C_1 in (SU, PV, SC). Indeed, in UD, each positive user sends a list of seeds from which the server retrieves the ephemeral IDs and then can link them.

Due to Proposition 6.1, $(SU, PV, SC) \in \mathscr{I}^{C_1}_{UD}$, then there exists $S_2 \in \mathscr{V}^{C_1}_{UD}$ such that $(SU, PV, SC) \preceq_S S_2$. Therefore, the proof of (i) is concluded.

Consider (ii).

The set $\mathscr{I}^{C_1}_{UD} = \{(OU,GV,IC),(OU,PV,IC),(SU,GV,IC),(HU,GV,IC),(SU,PV,SC),(FC,GV,IC),(SU,PV,IC),(HU,PV,IC),(FC,PV,SC),(FC,PV,IC)\} \subseteq \mathscr{I}^{C_1}_{UD}$ since each element $S \in \mathscr{I}^{C_1}_{UD}$ is such that either $S \preceq_S (OU,GV,IC)$ or $S \preceq (SU,PV,SC)$ and $(OU,GV,IC),(SU,PV,SC) \in \mathscr{I}^{C_1}_{UD}$.

Therefore, $|\mathscr{I}^{C_1}_{UD}| \geq \mathscr{I}^{C_1}_{UD} = 10$ and then $|\mathscr{I}^{C_1}_{UD}| - |\mathscr{I}^{C_1}_{ZP}| \geq 4$. This concludes the proof of (ii) and then the proof of the Theorem too. □

Now, we compare the security of ZE2-P3T with DP-3T with respect to the compromise C_2. The next theorem states that, on the compromise C_1, Low-Cost DP-3T is more vulnerable (with *degree* 4) than our protocol, and Unlinkable DP-3T is more vulnerable (with *degree* 4) than our protocol.

Theorem 6.2. *LD $\preceq_4$ ZP $\wedge$UD $\preceq_4$ ZP on the compromise C_2.*

Proof. First, we prove $\mathscr{V}^{C_2}_{ZP} = \{(SU,GV,SC),(OU,GV,IC)\}$.

To do this, first (i) we prove that ZP is vulnerable to C_2 in (SU,GV,SC) and (OU,GV,IC).

Then (ii), we prove that ZP is not vulnerable to C_2 in any setting greater than (SU,GV,SC) or (OU,GV,IC).

Finally (iii), we prove that ZP is not vulnerable to C_2 in any other setting incomparable w.r.t (SU,GV,SC) and (OU,GV,IC).

We proceed by proving (i). Regarding the compromise C_2, the setting (SU,GV,SC) means that the attacker colludes with the server to forge fake contacts among a scalable number of generic users. In ZP, the server can forge fake contact tuples during the contact-detection phase by using the ephemeral IDs received by the users during the user-side activity. Then, ZP is vulnerable to C_2 in all the settings where the first component is SU or FC. In particular, ZP is vulnerable to C_2 in (SU,GV,SC).

Consider now the setting (OU,GV,IC). It means that the attacker tries to forge (without colluding with the server) fake contacts among a restricted number of generic users. Considering a single victim, the attacker can physically follow the victim (without entering into contact), capture the salt from the TSP, and compute the same digest of the victim (by using the same centroid). At this point, the attacker is able to build a restricted tuple that matches (same time slot and same digest) with that provided by the victim. Then, ZP is vulnerable to C_2 in (OU,GV,IC).

The proof of (i) is given.

Now, consider (ii). The only setting greater than (SU,GV,SC) or (OU,GV,IC) is (OU,GV,SC) (that does not include collusion with the server). To forge a fake contact, as the server is trusted, the attacker should be able to send the server fake information that will determine a (forged) contact. Recall that a contact between two ephemeral IDs is detected on the basis of the timestamp and digest. Therefore, the attacker should tamper with one of the above pieces of information. First, consider the ephemeral IDs. As they are not exchanged among users, the only possibility is that two colluding attackers agree by switching their ephemeral IDs with the aim to jeopardize either the user-side activity or infection-reporting phase by reporting the ephemeral of the other user. However, due to **Remark 2**, we are not in the case of the compromise C_2.

Consider now the remaining information (i.e., timestamp and digest). Since the server checks the sent timestamp is within the current time slot or within the previous one, it is not possible to tamper with such information. Regarding the digests, as observed in Theorem 6.1, they cannot be tampered with without knowing the salts provided by the TSP. Then, by Assumption **A2**, and by considering that, when an infection is reported, the server takes into consideration only one ephemeral per time slot, it is not possible to forge digests on a large scale even in case of high concentration of people (for which a few salts would be enough for the attacker). Therefore, ZP is not vulnerable to C_2 in (OU,GV,SC). This concludes the proof of (ii).

Finally, consider (iii). The settings incomparable with respect to (SU,GV,SC) and (OU,GV,IC) are $\{(OU,PV,SC),(HU,GV,SC),(HU,PV,SC)\}$. Since the server does not collude, by applying similar reasoning to that used in the proof of (ii), by Assumption **A2**, it is not possible to forge contact on a large scale. Then ZP is not vulnerable to C_2 in (OU,PV,SC), (HU,GV,SC), and (HU,PV,SC). Therefore, the proof of (iii) is concluded.

At this point, since $\mathscr{V}^{C_2}_{ZP} = \{(SU,GV,SC),(OU,GV,IC)\}$, the induced set by $\mathscr{V}^{C_2}_{ZP}$ is $\mathscr{I}^{C_2}_{ZP} = \{(SU,GV,SC),(OU,GV,IC),(OU,PV,IC),(SU,GV,IC),(HU,GV,IC),(FC,GV,SC),(SU,PV,SC),(FC,GV,IC),(SU,PV,IC),(HU,PV,IC),(FC,PV,SC),(FC,PV,IC)\}$ and $|\mathscr{I}^{C_2}_{ZP}| = 12$.

Regarding both LD and UD, consider the setting (OU,GV,SC). The attacker can detect the ephemeral IDs released in broadcast by the users (potentially, a huge number) and send them other ephemeral IDs detected by other users through BLE antennas spread over the territory.

Then, both LD and UD are vulnerable to C_2 in (OU,GV,SC).

Each setting $S \in \mathscr{S}$ is such that $S \preceq_S (OU,GV,SC)$, therefore we have that:

$\mathscr{V}^{C_2}_{LD} = \mathscr{V}^{C_2}_{UD} = \{(OU,GV,SC)\}$ and $|\mathscr{I}^{C_2}_{LD}| = |\mathscr{I}^{C_2}_{UD}| = |\mathscr{S}| = 16$. Therefore, LD $\preceq_4$ ZP $\wedge$UD $\preceq_4$ ZP.

The proof of the theorem is concluded.□

In the next theorem, we state the superiority of ZE2-P3T on the security with respect to the compromise C_3. Indeed, on the compromise C_3, Low-Cost DP-3T is more vulnerable (with *degree* 12) than our protocol, and Unlinkable DP-3T is more vulnerable (with *degree* 12) than our protocol.

Theorem 6.3. *$LD \preceq_{12} ZP \wedge UD \preceq_{12} ZP$ on the compromise C_3.*

Proof. First, we prove $\mathscr{V}_{ZP}^{C_3} = \{(SU, GV, IC)\}$.

To do this, first (i) we prove that ZP is vulnerable to C_3 in (SU, GV, IC).

Then (ii), we prove that ZP is not vulnerable to C_3 in any setting greater than (SU, GV, IC).

Finally (iii), we prove that ZP is not vulnerable to C_3 in any other setting incomparable w.r.t (SU, GV, IC).

We proceed by proving (i). In the setting (SU, GV, IC), the server colludes with the attacker in order to associate extra information (e.g., the position) with the ephemeral IDs of a limited number of generic users.

The reasoning is the same as that done in Theorem 6.1 for the setting (SU, GV, IC). The attacker may select a victim and detect the salts sent from the TSP in the zones in which the victim is located. This way, the server can compute the digests, identify the (restricted) tuples sent by the user containing the ephemeral IDs, and associate extra information to these latter. Thus, ZP is vulnerable to C_3 in (SU, GV, IC). The proof of (i) is given.

Now, consider (ii). The settings greater than (SU, GV, IC) are $\{(SU, GV, SC), (OU, GV, IC), (OU, GV, SC)\}$.

The setting (SU, GV, SC) differs from (SU, GV, IC) only because the former involves a scalable number of victims. According to the reasoning done for the setting (SU, GV, IC), (SU, GV, SC) would be vulnerable only if a scalable number of salts given by the TSP are known to the server. Anyway, receiving a scalable number of salts from different places in the territory is impossible according to Assumption **A2**. Thus, ZP is not vulnerable to C_3 in (SU, GV, SC).

In the setting (OU, GV, IC), the server does not collude. As already discussed in Theorem 6.1, an attacker non-colluding with the server can obtain the ephemeral IDs of other users only at the end of the contact-detection phase but no extra information is associated with them. Then, there is no way for an attacker to link these ephemeral IDs with other external information. Thus, ZP is not vulnerable to C_3 in (OU, GV, IC).

Due to the Lemma 6.1, since ZP is not vulnerable to C_3 in (SU, GV, SC), then it is not vulnerable to C_3 in (OU, GV, SC). Therefore, the proof of (ii) is concluded.

Finally, consider (iii). The settings incomparable with respect to (SU, GV, IC) are $\{(OU, PV, SC), (HU, GV, SC), (OU, PV, IC), (HU, GV, IC), (FC, GV, SC), (SU, PV, SC), (HU, PV, SC), (HU, PV, IC), (FC, PV, SC)\}$.

Consider the setting (FC, PV, SC), where HF and the server collude with the attacker in order to associate extra information to an ephemeral ID of a scalable number of positive users. The ephemeral IDs are released by the users or (i) during the user-side activity (previously they perform the test in the HF) or (ii) during the infection-reporting phase. Regarding (i), HF is not involved and the positive users perform as generic users, then the setting (FC, PV, SC) is equivalent to (SU, GV, SC) and due to Assumptions **A2**, ZP is not vulnerable to C_3 in (SU, GV, SC)). In case (ii), the users upload their tuples containing the ephemeral IDs to the server at a non-predictable instant, and then, no information can be associated directly with them. However, some information can be associated indirectly through the collusion with HF. Even though HF knows extra information about a positive user (e.g., the identity) and communicates it to the server, there is no way to link such information with the ephemeral IDs of the user. Indeed, the blind signature scheme ensures that the credential $\sigma(M)$, used as authorization code to upload the ephemeral IDs, is unlinkable by HF with the message provided by the user to obtain the above credential. Thus, there is no way for the attacker to link the uploaded ephemeral IDs to any associated extra information. Then, ZP is not vulnerable to C_3 in (FC, PV, SC).

Due to the Lemma 6.1, since ZP is not vulnerable to C_3 in (FC, PV, SC), then it is not vulnerable to C_3 in $(FC, GV, SC), (SU, PV, SC), (HU, PV, SC), (HU, GV, SC)$, and (OU, PV, SC).

Consider the setting (HU, PV, IC). Again, an attacker non-colluding with the server can obtain the ephemeral IDs only at the end of the contact-detection phase but no extra information is associated with them. There is no way for an attacker to link these ephemeral IDs with other external information. Thus, ZP is not vulnerable to C_3 in (HU, PV, IC).

Due to the Lemma 6.1, since ZP is not vulnerable to C_3 in (HU, PV, IC), then it is not vulnerable to C_3 in (HU, GV, IC), and (OU, PV, IC). The proof of (iii) is given.

At this point, since $\mathscr{V}_{ZP}^{C_3} = \{(SU, GV, IC)\}$, the induced set by $\mathscr{V}_{ZP}^{C_3}$ is $\mathscr{I}_{ZP}^{C_3} = \{(SU, GV, IC), (FC, GV, IC), (SU, PV, IC), (FC, PV, IC)\}$ and $|\mathscr{I}_{ZP}^{C_1}| = 4$.

Regarding both LD and UD, consider the setting (OU, GV, SC). It is easy to see that both LD and UD are vulnerable to C_3 in (OU, GV, SC). Indeed, the attacker, without colluding with the server, can associate extra information with each ephemeral ID released in broadcast (in plaintext) by the users and this can be done against a scalable number of users, without violating limits imposed by Assumption **A2** (i.e., the intervention of a non-attacker TTP).

Each setting $S \in \mathscr{S}$ is such that $S \preceq_S (OU, GV, SC)$, therefore we have that:

$\mathscr{V}_{LD}^{C_3} = \mathscr{V}_{UD}^{C_3} = \{(OU, GV, SC)\}$ and $|\mathscr{I}_{LD}^{C_3}| = |\mathscr{I}_{UD}^{C_3}| = |\mathscr{S}| = 16$. Therefore, $LD \preceq_{12} ZP \wedge UD \preceq_{12} ZP$.

The proof then is concluded. □

In Figs. 14–16 we provide, coherently with Theorems 6.1, 6.2, and 6.3, a visual representation of the improvements in terms of security given by our protocol with respect to DP-3T. Therein, all the settings of S are represented and it appears evident that the sets of settings in which the latter protocol is vulnerable (in both designs Low-cost and Unlinkable) are supersets of the vulnerable settings for our protocol, for any compromise. For example, from Fig. 15, we see that both LD and UD are vulnerable to C_2 in all the settings of S. Note that, Fig. 14, corresponding to Theorem 6.1, represents the set of

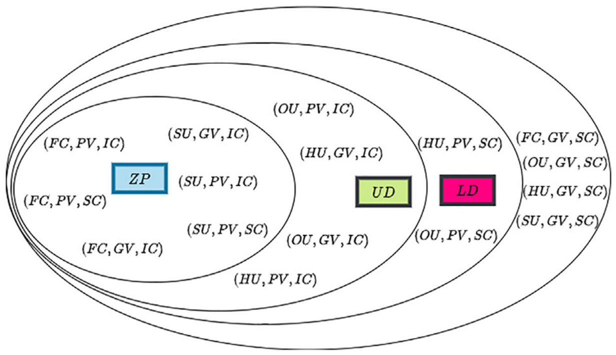


Fig. 14. Visual comparison on the compromise C_1.

all the settings in which ZP is vulnerable. Instead, for UD and LD, the theorem singles out a subset (possibly improper) of settings in which they are vulnerable. Indeed, we only need to have a *lower bound* difference in terms of security between ZP and DP-3T, not a complete characterization of the security of DP-3T. The same reason cannot be applied to Theorems 6.2, and 6.3, and then to Figs. 15 and 16, because in that case all the settings of S are covered.

In the sequel of the section, we show several attacks already reported in the literature and contextualize them within the theoretical framework introduced earlier. We remark that this description is not intended to further demonstrate anything about the security of our approach, which is completely addressed so far. However, it has practical relevance, since it witnesses, through concrete exploits, how the existing techniques perform. We remark that, given a setting, when a protocol is vulnerable to a compromise in this setting, it means that there exists at least one attack witnessing such a vulnerability. Therefore, it may be the case that for other attacks in the same setting, this protocol is not vulnerable.

Paparazzi Attack [8]: The purpose of the attack is to put on a mass tracking system on infected users. The attacker can be anyone and no collusion with the server or HF is required. The attack leads to the compromises C_1 and C_3 and the setting considered is (OU, PV, SC). It is performed as follows. First, the attacker places several passive BLE devices through the territory in order to collect the ephemeral IDs of other users located in the proximity of such devices. Moreover, it records the time and the location where such identifiers are received and, possibly, other information about the users. This attack works only on the Low-Cost Design of DP-3T. Indeed, when a user U results infected, they send a single secret seed SK to the server which, in turn, broadcasts it (together with the seeds of other positive users) to all the users, including the attacker.

Starting from SK, the attacker is able to generate all the ephemeral identifiers of U and, for each of them, the attacker retrieves the data (time, location, etc.) stored by passive devices when U was in their proximity. The linkage of these data allows the attacker to track the user.

For example, suppose the passive devices associate the coordinates of the users with their ephemeral IDs. By linking the IDs of a positive user, the attacker retrieves a list of pairs of coordinates (each pair is associated with an ephemeral ID) that allow identifying the trajectory of the positive user.

This attack does not work on the Unlinkable design of DP-3T since the infected user U sends the seeds to generate the ephemeral IDs to the server, but this latter does not broadcast such seeds to all users. Instead, the server generates all the ephemeral identifiers of U and adds them to the Cuckoo filter, so that the attacker cannot link them as belonging to the same user.

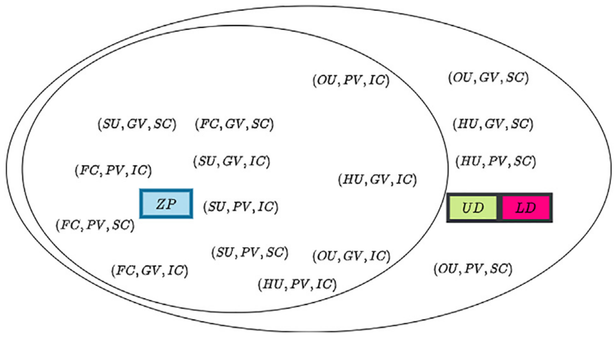


Fig. 15. Visual comparison on the compromise C_2.

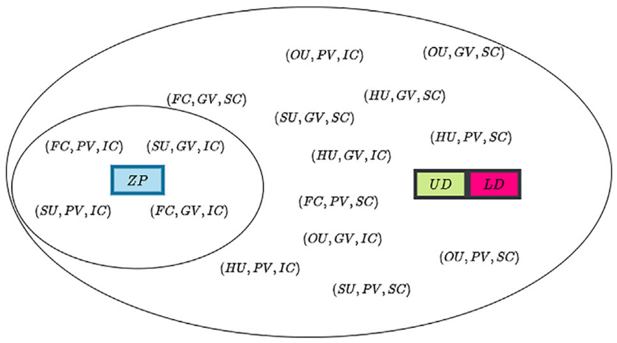


Fig. 16. Visual comparison on the compromise C_3.

Regarding ZP, the ephemeral IDs are not exchanged through BLE, but they are sent directly to the server (encrypted with its public key). Therefore, there is no way for the attacker to associate extra information with these IDs and the attack cannot be performed.

We highlight that the above discussion is perfectly aligned with Theorems 6.1 and 6.3. Indeed, the attack works with LD that is vulnerable to C_1 and C_3 in the setting (OU, PV, SC) and it does not work on ZP that is not vulnerable neither to C_1 nor to C_3 in (OU, PV, SC). Regarding UD, it is vulnerable to C_3 in the setting (OU, PV, SC) but we did not prove that it is vulnerable to C_1 in (OU, PV, SC). Note that Paparazzi can be viewed as an implementation of the most general concept of *linkage attack* [14]. Clearly, the relationship described here between Paparazzi and our security framework can be directly applied to any linkage attack.

Orwell Attack [8]: The attack is very similar to the Paparazzi attack. The difference is that the attacker colludes with the server and, consequently, they can access all the information therein stored. This makes both the Unlinkable and Low-Cost designs of DP-3T vulnerable to the Orwell attack. The attack leads to the compromises C_1 and C_3 and the setting considered is (SU, PV, SC).

Regarding the Low-Cost design, trivially, the Orwell attack without the collusion with the server is equivalent to the Paparazzi attack to which the Low-Cost design is vulnerable.

Regarding the Unlinkable design, what is missing for the attacker to perform the Paparazzi attack is the linkage between the ephemeral identifiers of the same positive user. In the Orwell attack, this linkage can be provided by the server and the attack can be performed.

In detail, the server knows the seeds uploaded by an infected user, and thus, it is able to link the ephemeral identifiers generated by such seeds. Once obtained the ephemeral IDs, the attack performs exactly as the Paparazzi attack, and the user is tracked through the information stored when they passed near the passive devices.

We show that such an attack is not possible in ZP. Indeed, as discussed in the Paparazzi attack, the ephemeral IDs (or other identifying information) are not exchanged between users but are sent directly to the server. However, from the considerations of Theorem 6.3, the server is not able to identify the tuples sent by the users in a massive fashion due to Assumption **A2**. Then, even though the attacker captures information of users in several places, it cannot associate them with their ephemeral IDs.

W.r.t Theorems 6.1 and 6.3, this attack works on LD and UD and, indeed, they are vulnerable to both C_1 and C_3 in (SU, PV, SC). Instead, ZP is not vulnerable to such an attack as we expected since it is not vulnerable to C_3 in (SU, PV, SC).

Brutus attack [8]: In this attack, the attacker colludes with the health facility HF and the server to find out the mapping between pseudonyms and real identities of infected users during the infection-reporting phase. The attack leads to the compromises C_1 and C_3 and the setting considered is (FC, PV, SC).

It is an exploit of the authorization mechanism with which infected users communicate their status to the server. DP-3T (both the designs) proposes three different authorization mechanisms but they are, essentially, based on an authorization code released by HF to the user and to the server.

Clearly, HF knows the real identity of the user performing the test and, therefore, knows the mapping between the identity and the authorization code.

On the other hand, when the infected users upload their ephemeral IDs (through a single seed for the Low-Cost design or through a set of seeds for the Unlinkable design), the server knows the mapping between such IDs and the authorization code. By merging the two mappings, the attacker is able to associate the real identity of the users with their ephemeral IDs.

In ZP, the authorization code is replaced by M which cannot be linked by HF to the message submitted by the user to obtain the signature, thanks to the blind signature mechanism.

Server-side, when the server verifies the signature, it is sure that a generic user is enabled by HF to upload its ephemeral ids, but it does not know who the user is.

Thus, both HF and the server cannot link M to the real identity of the user. In conclusion, ZP is not vulnerable to Brutus attack.

W.r.t Theorems 6.1 and 6.3, this attack works on LD and UD and, indeed, they are vulnerable to both C_1 and C_3 in (FC, PV, SC). ZP is not vulnerable to such an attack since, even though it is vulnerable to C_1 in (FC, PV, SC), it is not vulnerable to C_3 in (FC, PV, SC).

Gossip attack [8]: The objective of this attack is to provide any evidence about an encounter with an infected user before discovering their positiveness to the infection. No collusion with the server or HF is required. The attack leads to the compromises C_1 and C_3 and the setting considered is (OU, PV, IC).

The attack works on both the designs of DP-3T due to the exchange of the ephemeral IDs.

The attacker intercepts the ephemeral IDs of the other users (it can use BLE passive devices as in the Paparazzi attack) and stores them in a repository that provides sufficient guarantees on the time t in which data are stored. For example, we can think of blockchain.

If any of such users is tested positive, they upload their ephemeral IDs and the attacker is able to prove an encounter with the infected user at a given instant t before the upload of the IDs.

It can be viewed as a security flaw because it is a misuse of the system for an unintended scope, potentially threatening privacy and exploitable for disputes.

In ZP, this attack is not possible since users do not exchange any ephemeral ID.

Again, these results are compliant with Theorems 6.1 and 6.3. Indeed, LD and UD are vulnerable to both C_1 and C_3 in (OU, PV, IC) and ZP is not vulnerable neither to C_1 nor to C_3 in (OU, PV, IC).

Matteotti attack [8]: In this attack, the attacker colludes with the server in order to build a fake contact between a user victim and a positive user. The result aimed by the attacker is to damage the victim by enforcing their quarantine (or other consequent actions). The attack leads to the compromises C_2 the setting considered is (SU, GV, IC). We denote by U_v the user victim and by U_s any user which entered into contact with U_v exchanging its ephemeral IDs. The attack works only on the Unlinkable design of DP-3T.

The attacker captures (e.g., through BLE passive devices) the ephemeral IDs generated by U_s when they encounter U_v and sends them to the server. The latter inserts such identifiers in the Cuckoo filter, even if U_s is not positive, so that, when U_v checks the filter, they are wrongly alerted.

In the Low-Cost design, the ephemeral IDs of a positive user are generated directly by the other users through a single secret seed (provided by the infected user) that the attacker or the server cannot recover. Therefore, the attack does not work.

In ZP, the server, colluding with a partner located in the same zone as the victim, can retrieve the salt of the TSP, identify the ephemeral ID of the victim, and add it to the list of positive ephemeral IDs. Thus, ZP is susceptible to this attack.

In summary, this attack works on ZP and UD and they are vulnerable to C_2 in (SU, GV, IC). However, even though such an attack does not work on LD, it is anyway vulnerable to C_2 in (SU, GV, IC) since, as shown in Theorem 6.2, it is vulnerable to C_2 in (OU, GV, SC).

The following two attacks are in the setting (OU, GV, SC), and both LD and UD are vulnerable.

Missile attack [46]: Similarly to the Matteotti attack, in the Missile attack the objective is to alert other users with a fake contact with a positive user. The attack leads to the compromises C_2 the setting considered is (OU, GV, SC). This time, the attacker colludes with a positive user U to obtain their ephemeral IDs before U communicates them to the server. No collusion with the server is required.

The attacker can use a Bluetooth amplifier transmitter to send (like a *missile*) the ephemeral identifiers of U to other users even very far from the positive user and so, not at risk.

At a later moment, when such ephemeral IDs are reported to the server, the other users are wrongly alerted. Since the user U colluding with the attacker is really positive, in the Low-Cost design, U can upload the secret seed that generates their ephemeral IDs, thus this attack works on both the designs of DP-3T.

On the contrary, ZP does not suffer from this attack since no identifier is exchanged through BLE.

These results are compliant with Theorem 6.2 that shows that LD and UD are vulnerable to C_2 in all the settings, and then in (OU, GV, SC), while ZP is not vulnerable to C_2 in (OU, GV, SC).

Fregoli (or Relay) attack: This attack has been reported independently in [31,46] (with a slight temporal shift) and is one of the most serious threats of an ephemeral-based contact-tracing protocol.

In this attack, the attacker collects the ephemeral IDs of other users and sends them in broadcast in place of their own. The purpose is to simulate fake contacts between users. The attack leads to the compromises C_2 and the setting considered is (OU, PV, SC). This way, a generic user U maintains a list of ephemeral IDs belonging to users U never met. If any of such users is positive, U is wrongly alerted. This is then an impersonation attack, as its name evokes, being Fregoli one of the major *quick-change* artists of the story. This attack is more effective when a Bluetooth amplifier is used as in the Missile attack. Again, the vulnerability is the exchange of ephemeral IDs between users, thus the attack works on both the design of DP-3T but not on ZP.

These results are compliant with Theorem 6.2 that shows that LD and UD are vulnerable to C_2 in all the settings, and then in (OU, GV, SC), while ZP is not vulnerable to C_2 in (OU, GV, SC).

Battleship attack [46]: This attack applies to the localization-based solutions.

The attacker, colluding with the server, tries to identify the position of a huge number of users to track them. The compromises involved are C_1 and C_3 since if the attacker knows the position of a user at a given instant, it is able to identify the tuples sent by the user and then to link the ephemeral IDs they contain and to associate them with extra information (e.g., the position of the user). The setting considered is (SU, GV, SC).

Since in DP-3T, no information about the position is sent to the server, the attack cannot be performed. On the contrary, any standard GPS-based solution is affected by this problem. Therefore, it is important to check what happens in the case of our protocol. In ZP, the only information about the position of the users is reported in the digests. The only way to reverse them (and discover the centroids in which the users are located) is to know the salts sent by the TSP.

As explained in the Orwell Attack, even though the attack can be performed on a limited number of users, putting in place a mass tracking system is infeasible.

In summary, this attack does not work on LD, UD, and ZP.

Little Thumb attack [9]: This attack is due to the practical implementation of DP-3T based on GAEN. Indeed, to avoid linkage of ephemeral IDs, they have to change simultaneously with the MAC addresses of the BLE messages sent in broadcast. In fact, if they are not perfectly synchronized, linking ephemeral IDs becomes possible when the ephemeral id changes before the MAC address. In this case, the user sends in broadcast two consecutive messages with the same MAC address and two different ephemeral IDs that can be linked. Once the ephemeral IDs are linked, we have all the drawbacks of the Paparazzi and Orwell attacks. The attack leads to the compromises C_1 and the setting considered is (OU, GV, IC).

In ZP, ephemeral IDs change randomly each τ seconds and they are not exchanged through BLE, thus the attack cannot be performed.

These results are compliant with Theorem 6.1 that shows that LD and UD are vulnerable to C_1 in (OU, GV, IC), while ZP is not vulnerable to C_1 in (OU, GV, IC).

We highlight that, although the attacks regard DP-3T, they also apply to other decentralized protocols [20,17,18] as the vulnerabilities are due to the exchange of identifiers.

The vulnerabilities of DP-3T and ZE2-P3T to the attacks are summarized in Fig. 17.

7. Discussion

History teaches us that fighting a pandemic is in general a very difficult task. Despite the scientific and technological advancements, we experimented that a pandemic can seriously threaten and damage the world, and its social and economical systems. The strategy to adopt to fight a pandemic strongly depends on the characteristic of the pathogen and the disease at the basis of the pandemic. But, in general, the strategy is a combination of different measures.

Contact tracing (both traditional and digital) is one of the weapons in the heads of governments, but it is not enough. It should be suitably combined with containment measures. They are, for example: Health checkpoints for passengers, banning air traffic from specific areas of the world, mandatory supervised quarantine, mandatory communication, red zones, suspension of public events, general lockdown or movement restrictions, suspension of all commercial activities non-indispensable for production, extension of the ban to non-indispensable activities, vaccination, etc. Contact tracing can be effective in some phases of the pandemic, and could be also restricted to specific areas of the country in which the conditions are appropriate. In particular, contact tracing can be fruitful if the incidence of the infection is below a given threshold. Indeed, the role of contact tracing is to enable early reactions by applying additional strict containment measures (such as quarantine) only to those people that are identified by the tracing activity. Conversely, when the incidence of the pathology is high, the role of contact tracing decreases and also its feasibility, due to the number of cases to manage. On the other hand, in this case, a general containment protocol can be applied. All the above strategies can encounter a number of obstacles, which are cultural, social, technological, economic, etc. The obstacles themselves may become opportunities for designing more effective strategies also by reducing their negative impact. For example, the heterogeneity of the population introduces specific issues to consider. Certain differences in intrinsic characteristics of the population may influence the dynamics of the local epidemic. Therefore, governments should consider local conditions to adopt heterogeneous containment policies. Moreover,

Attack	LD	UD	ZP
Paparazzi	✗	✓	✓
Orwell	✗	✗	✓
Brutus	✗	✗	✓
Gossip	✗	✗	✓
Matteotti	✓	✗	✗
Missile	✗	✗	✓
Fregoli/Relay	✗	✗	✓
Battleship	✓	✓	✓
Little Thumb	✗	✗	✓

Fig. 17. Vulnerabilities of DP-3T and ZE2-P3T to the attacks. **X** means vulnerable while ✓ means resistant.

the coexistence with other epidemics (such as the seasonal flu) is another factor that can hinder the management of a pandemic, especially when the symptoms may be similar.

This paper focuses on digital contact tracing, by proposing a new protocol that improves the state of the art from the perspective of security and privacy. The motivation of this paper is that the precondition for the effectiveness of digital contact tracing is not just the fact that it should be properly combined with containment measures, but also that it should be accepted by the population. Therefore, security and privacy features take a central role. Concerning security, in particular, as some attacks that we contrast (such as the relay attack – also said Fregoli attack) can threaten the stability of large communities, governments may also be seriously interested in this aspect. Consider that the relay attack could be performed on a massive scale, potentially quarantining many people. This could be exploited, for example, in the case of elections, to inhibit their regular running. Therefore, the presented research has a specific focus on security and privacy aspects, without neglecting the feasibility of the proposed solution, which is a relevant point, considering that it should be designed for a huge number of users. A final aspect to discuss is if the approach presented in this paper is generalizable to all contagious diseases. Obviously, the answer to this question should take into account the specific characteristics of the considered disease. Apart from the obvious fact that even short-term, non-close contacts could be significant only for respiratory infections, a contact tracing solution is suitable for a specific disease depending on the safety distance, incubation period, role of shared surfaces and objects, presence of paucisymptomatic or asymptomatic cases, speed of diagnostic test, and so on.

There are so many variables to consider that, as the recent scientific literature witnesses for the case of COVID-19, it is preferable to narrow the action range to a single disease, which is the choice adopted in this paper.

8. Conclusion

In this paper, a novel contact tracing protocol is presented. It is based on a centralized model, as contact detection is in charge of the server, which constantly receives the individual locations of users. However, the information is processed in such a way that the server cannot learn anything other than the fact that two random numbers (i.e., ephemeral identifiers associated with non-identifiable humans) came into contact at a given time in a non-identifiable place. The most important feature of our approach is that such ephemeral identifiers are not exchanged among users, as it happens for the state-of-the-art decentralized protocol DP-3T/GAEN. This is important because this exchange is the basis of most vulnerabilities of DP-3T/GAEN. As a consequence, the proposed approach is definitely more secure than DP-3T/GAEN. This fact represents the main result of the paper.

To show this, we define a theoretical framework able to quantitatively compare the two approaches, according to a suitably weighted ordering relation.

Interestingly, our protocol is not based on BLE. Therefore, the users are not forced to turn on the Bluetooth interface (as it happens for DP-3T/GAEN). This shields smartphones from other attacks that exploit Bluetooth (widely discussed in Section 2).

An important point to remark is that, in this paper, the proposed solution has been also analyzed experimentally, to test its feasibility and also to provide a methodology to follow in a real-life context to size the server-side resources. To this end, we developed a software prototype that implements the main functionalities of the solution.

As future work, we plan to transform the software prototype presented in this paper into a complete software system, to run it in real-life contexts. Indeed, we argue that a real-life experimentation, possibly with the partnership of industrial and government parties, would be a desirable follow-up of the present scientific study.

CRediT authorship contribution statement

Francesco Buccafurri: Conceptualization, Methodology, Formal analysis, Investigation, Validation, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Vincenzo De Angelis:** Conceptualization, Methodology, Formal analysis, Investigation, Validation, Writing – original draft, Software, Writing – review & editing. **Cecilia Labrini:** Conceptualization, Methodology, Investigation, Validation, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors are grateful to Marco Ventura for his help in implementing the Client and Health Facility modules. The experimental campaign was carried out as part of the activities of the research project “iCARE” (CUP J39J14001400007) - action 10.5.12.

References

[1] T. Jiang, Y. Zhang, M. Zhang, T. Yu, Y. Chen, C. Lu, J. Zhang, Z. Li, J. Gao, S. Zhou, A survey on contact tracing: the latest advancements and challenges, ACM Transactions on Spatial Algorithms and Systems (TSAS) 8 (2) (2022) 1–35.
[2] F. Vogt, B. Haire, L. Selvey, A.L. Katelaris, J. Kaldor, Effectiveness evaluation of digital contact tracing for covid-19 in new south wales, australia, The Lancet Public Health (2022).
[3] A. Barnawi, P. Chhikara, R. Tekchandani, N. Kumar, B. Alzahrani, Artificial intelligence-enabled internet of things-based system for covid-19 screening using aerial thermal imaging, Future Generation Computer Systems 124 (2021) 119–132.
[4] L. Ouyang, Y. Yuan, Y. Cao, F.-Y. Wang, A novel framework of collaborative early warning for covid-19 based on blockchain and smart contracts, Inf. Sci. 570 (2021) 124–143, https://doi.org/10.1016/j.ins.2021.04.021.
[5] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, et al., Decentralized privacy-preserving proximity tracing, arXiv preprint arXiv:2005.12273 (2020).
[6] Apple, Google, Apple and google's exposure notification system (2020). https://www.apple.com/covid19/contacttracing.
[7] E. eHealth Network, Mobile applications to support contact tracing in the eu's fight against covid-19 - common eu toolbox for member states, versio 1.0 (15.04.2020).
[8] G. Avitabile, V. Botta, V. Iovino, I. Visconti, Towards defeating mass surveillance and sars-cov-2: The pronto-c2 fully decentralized automatic contact tracing system, IACR Cryptol. ePrint Arch. 2020 (2020) 493.
[9] S. Vaudenay, M. Vuagnoux, Little thumb attack on swisscovid (2020). URL: https://vimeo.com/453948863.
[10] L. Baumgärtner, A. Dmitrienko, B. Freisleben, A. Gruler, J. Höchst, J. Kühlberg, M. Mezini, R. Mitev, M. Miettinen, A. Muhamedagic, et al, Mind the gap: Security & privacy risks of contact tracing apps, in: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE, 2020, pp. 458–467.
[11] D.J. Leith, S. Farrell, Contact tracing app privacy: What data is shared by europe's gaen contact tracing apps, Testing Apps for COVID-19 Tracing (TACT) (2020).
[12] Privacy and security risk evaluation of digital proximity tracing systems, https://github.com/DP-3T/documents/blob/master/Securityanalysis/PrivacyandSecurityAttacksonDigitalProximityTracingSystems.pdf (2020).
[13] H. Faria, S. Paiva, P. Pinto, An advertising overflow attack against android exposure notification system impacting covid-19 contact tracing applications, IEEE Access 9 (2021) 103365–103375.
[14] N. Ahmed, R.A. Michelin, W. Xue, S. Ruj, R. Malaney, S.S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, S.K. Jha, A survey of covid-19 contact tracing apps, IEEE Access 8 (2020) 134577–134601.
[15] B. Sowmiya, V. Abhijith, S. Sudersan, R.S.J. Sundar, M. Thangavel, P. Varalakshmi, A survey on security and privacy issues in contact tracing application of covid-19, SN computer science 2 (3) (2021) 1–11.
[16] O. Seiskari, Pble contact tracing sniffer poc, https://github.com/oseiskar/coronasniffer (2020).
[17] R.L. Rivest, J. Callas, R. Canetti, K. Esvelt, D.K. Gillmor, Y.T. Kalai, A. Lysyanskaya, A. Norige, R. Raskar, A. Shamir, et al., The pact protocol specification, Private Automated Contact Tracing Team, MIT, Cambridge, MA, USA, Tech. Rep. 0.1 (2020).
[18] J. Chan, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, S. Singanamalla, J. Sunshine, et al., Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing, arXiv preprint arXiv:2004.03544 (2020).
[19] S. Brack, L. Reichert, B. Scheuermann, Caudht: decentralized contact tracing using a dht and blind signatures, in: 2020 IEEE 45th Conference on Local Computer Networks (LCN) IEEE, 2020, pp. 337–340.
[20] T. Coalition, Tcn protocol for decentralized, privacy-preserving contact tracing. https://github.com/TCNCoalition/TCN (2020).
[21] P. Tedeschi, S. Bakiras, R. Di Pietro, Iotrace: a flexible, efficient, and privacy-preserving iot-enabled architecture for contact tracing, IEEE Commun. Mag. 59 (6) (2021) 82–88.
[22] Z. Peng, C. Xu, H. Wang, J. Huang, J. Xu, X. Chu, P2b-trace: Privacy-preserving blockchain-based contact tracing to combat pandemics, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 2389–2393.
[23] G. Avitabile, D. Friolo, I. Visconti, Terrorist attacks for fake exposure notifications in contact tracing systems, in: International Conference on Applied Cryptography and Network Security Springer, 2021, pp. 220–247.
[24] L. Garg, E. Chukwu, N. Nasser, C. Chakraborty, G. Garg, Anonymity preserving iot-based covid-19 and other infectious disease contact tracing model, Ieee Access 8 (2020) 159402–159414.
[25] P.-P. Team, Pan-european privacy-preserving proximity tracing need-to-know system. overview of pepp-pt ntk (2020).
[26] C. Castelluccia, N. Bielova, A. Boutet, M. Cunche, C. Lauradoux, D. Le Métayer, V. Roca, Robert: Robust and privacy-preserving proximity tracing (2020).
[27] P.-P. Team, Pan-european privacy-preserving proximity tracing (2020). URL: https://www.pepp-pt.org/.
[28] S. Vaudenay, Centralized or decentralized? (2020).
[29] F. AISEC, Pandemic contact tracing apps: Dp-3t, pepp-pt ntk, and robert from a privacy perspective (2020).
[30] J. Bay, J. Kek, A. Tan, C.S. Hau, L. Yongquan, J. Tan, T.A. Quy, Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders, Government Technology Agency-Singapore, Tech. Rep (2020).
[31] S. Vaudenay, Analysis of dp3t: Between scylla and charybdis, Cryptology ePrint Archive, Report 2020/399, https://eprint.iacr.org/2020/399 (2020).
[32] K. Pietrzak, Delayed authentication: Preventing replay and relay attacks in private contact tracing, in: International Conference on Cryptology in India, Springer, 2020, pp. 3–15.
[33] T. Altuwaiyan, M. Hadian, X. Liang, Epic: Efficient privacy-preserving contact tracing for infection detection, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6.
[34] A.B. Dar, A.H. Lone, S. Zahoor, A.A. Khan, R. Naaz, Applicability of mobile contact tracing in fighting pandemic (covid-19): Issues, challenges and solutions, Computer Science Review 100307 (2020).
[35] C. Castelluccia, N. Bielova, A. Boutet, M. Cunche, C. Lauradoux, D. Le Métayer, V. Roca, Desire: A third way for a european exposure notification system leveraging the best of centralized and decentralized systems (2020).
[36] W. Beskorovajnov, F. Dörre, G. Hartung, A. Koch, J. Müller-Quade, T. Strufe, Contra corona: Contact tracing against the coronavirus by bridging the centralized–decentralized divide for stronger privacy, in: M. Tibouchi, H. Wang (Eds.), Advances in Cryptology – ASIACRYPT 2021, Springer International Publishing, Cham, 2021, pp. 665–695.
[37] N. Trieu, K. Shehata, P. Saxena, R. Shokri, D. Song, Epione: Lightweight contact tracing with strong privacy, arXiv preprint arXiv:2004.13293 (2020).
[38] Hamagen app, https://github.com/MohGovIL/hamagen-react-native (2020).
[39] J. Li, X. Guo, Global deployment mappings and challenges of contact-tracing apps for covid-19, Available at SSRN 3609516 (2020).
[40] Aarogya setu app, india, https://www.mygov.in/aarogya-setu-app (2020).
[41] L. Reichert, S. Brack, B. Scheuermann, Privacy-preserving contact tracing of covid-19 patients, IACR Cryptol. ePrint Arch. 2020 (2020) 375.
[42] X. Li, D. Wei, Q. Lai, Y. Xu, H. Yuan, Smartphone-based integrated pdr/gps/bluetooth pedestrian location, Adv. Space Res. 59 (3) (2017) 877–887.
[43] B.R. Stojkoska, I.N. Kosovic, T. Jagust, A survey of indoor localization techniques for smartphones, Web Proceedings of ICT Innovations (2016) 11–22.
[44] A. Bourdoux, A.N. Barreto, B. van Liempd, C. de Lima, D. Dardari, D. Belot, E.-S. Lohan, G. Seco-Granados, H. Sarieddeen, H. Wymeersch, et al., 6g white paper on localization and sensing, arXiv preprint arXiv:2006.01779 (2020).
[45] H.S. Maghdid, I.A. Lami, K.Z. Ghafoor, J. Lloret, Seamless outdoors-indoors localization solutions on smartphones: Implementation and challenges, ACM Computing Surveys (CSUR) 48 (4) (2016) 1–34.
[46] F. Buccafurri, V. De Angelis, C. Labrini, A privacy-preserving solution for proximity tracing avoiding identifier exchanging, International Conference on Cyberworlds (CW) 2020 (2020) 235–242, https://doi.org/10.1109/CW49994.2020.00045.

[47] T. Brinkhoff, A framework for generating network-based moving objects, GeoInformatica 6 (2) (2002) 153–180.
[48] S. Tarkoma, C.E. Rothenberg, E. Lagerspetz, Theory and practice of bloom filters for distributed systems, IEEE Communications Surveys & Tutorials 14 (1) (2011) 131–155.
[49] E. Dong, H. Du, L. Gardner, An interactive web-based dashboard to track covid-19 in real time, The Lancet infectious diseases 20 (5) (2020) 533–534.