https://doi.org/10.1016/j.osnem.2022.100221

A protocol for anonymous short communications in social networks and its application to proximity-based services

Francesco Buccafurri[a,*], Vincenzo De Angelis[a], Maria Francesca Idone[a], Cecilia Labrini[a]

[a] *University of Reggio Calabria, Via Università, 25, Reggio Calabria, Italy 89124*

Abstract

Several innovative applications could be advantageously placed within social networks, to be effective, attractive, and pervasive. Examples of application domains that could benefit from social networks are e-democracy, e-participation, online surveys, crowdsourcing, and proximity-based services. In all the above cases, users' anonymity could represent a considerable added value or could be even necessary to develop the service. We observe that all the above domains are characterized by the fact that only a few asynchronous messages should be exchanged. Therefore, we do not need the full communication power of anonymous communication networks, in which low-latency and connection-oriented communication should be supported. On the other hand, unlike communication networks, the threat model we have to consider assumes the presence of an adversary (represented by an honest-but-curious social network provider) able to monitor the entire flow of the exchanged messages. In this paper, we propose an anonymous communication protocol for short communications in social networks, based on a collaborative approach. The proposed solution hides from the social network provider not only the content of the messages but also the communication itself, which, per se, can result in considerable privacy leakage (think of the case of proximity testing performed between two users). This enables the implementation, within the social network, of the above-mentioned applications. To give a concrete proof of this statement, we develop a privacy-preserving proximity-based solution which provides both symmetric and asymmetric proximity testing entirely within social networks.

Keywords: Proximity-based services, Social Networks, Anonymous Communication, Privacy

1. Introduction

Social networks probably represent the most disrupting digital innovation of the last twenty years. Different kinds of applications are nowadays implemented on top of social networks. However, the power of social networks could be better exploited in various application contexts, such as e-democracy, e-participation, online surveys, crowdsourcing, proximity-based services, and so on. Often, in the above settings, the communication should happen between anonymous users or between anonymous users and explicit entities (possibly, the social network platform itself). Therefore, anonymity is a necessary feature. The problem is not trivial. Indeed, the social network provider should be seen, in a realistic threat model, as a global (at least) passive adversary, able to monitor the whole flow of messages among users.

The same threat occurs in the case of data breach. Therefore, privacy is achieved if not only the content of messages is protected against the adversary, but also the communication itself.

An important point is that the above applications are characterized by the common denominator that only a few asynchronous messages have to be anonymously exchanged. Therefore, we do not need the full communication power usually aimed in the domain of anonymous communication networks [1], supporting low-latency and connection-oriented communication.

To the best of our knowledge, there exist a few proposals regarding anonymous communication in social networks [2, 3]. However, [2] only deals with anonymous group communication and [3] does not provide sender anonymity against the global passive adversary, which is the goal we pursue in this paper.

Indeed, the aim of this paper is to achieve communication anonymity (in the case of short asynchronous messages) in social networks, against the global passive adversary. Then, we develop a detailed application of our protocol to the important setting of proximity-based services [4], when they are delivered entirely within social networks (without leveraging external communication channels). In the recent literature [5], there is growing attention towards proximity-based social networking. The problem of privacy in proximity-based services has been deeply analyzed in the literature for more than a decade [4, 6, 7, 8, 9], but never in the most severe threat model of a global passive adversary.

As a matter of fact, proximity-based services are becoming more and more important (besides meeting apps, think also of contact tracing, proximity advertising, homeland security, proximity marketing, etc.).

Current social networks offer proximity-based services. For example, Facebook supports the meeting feature *Nearby Friends*. There are also social networks founded on geospatial features, providing services based on the reciprocal proximity

*Corresponding author
Email address: bucca@unirc.it (Francesco Buccafurri)

of users like Foursquare, Jiepang, FullCircle, Tinder, Gowalla, and Facebook Places.

Unfortunately, in these services, users' privacy is threatened. Indeed, an *honest-but-curious* (also said *semi-trusted*) service provider could misuse location data, which are potentially sensitive.

If we refer to existing centralized social networks, the only way to make communications anonymous against the social network provider is to require the collaboration of social network users. This can be done by implementing an overlay network over the application layer provided by the social network itself. Indeed, an alternate approach based on a centralized party playing as anonymizer, has very limited effectiveness in our threat model, as shown in [10].

Therefore, an idea could be to translate into the domain of social networks one of the P2P approaches used in communication networks to achieve anonymity against the global passive adversary.

Existing P2P overlay network routing techniques resisting the global passive adversary always require the inclusion of cover traffic (i.e., dummy traffic to hide the actual messages) [11] and are based either on *mixnets* [12, 13, 14], or on *buses* [15, 16, 17].

It is intuitive to understand that state-of-the-art mixnet-based approaches require a high amount of cover traffic, which, in our context, would result in bandwidth and CPU overhead (thus also battery consumption) for social network users.

In the approach based on buses, anonymity is achieved by implementing routes (either deterministic [15, 16] or non-deterministic [17]) independent of the intended communication, which senders and receivers can opportunistically exploit. With this approach, cover traffic is drastically reduced with respect to mixnets (at the price of higher latency). Indeed, here, no mixing is adopted and the incoming (cover) traffic, for each node, has exactly one 1-hop source, and each node can indifferently play the role of sender, recipient, or relay node. However, both deterministic (in which the fixed route is an Eulerian path passing through all the nodes) and non-deterministic approaches (in which the latency highly increases with the number of nodes) are unrealistic in scenarios with a huge number of nodes like social networks.

Actually, our approach uses the concept of fixed deterministic routes of buses to minimize cover traffic. However, unlike buses, this mechanism is not used to hide inside both senders and receivers. Indeed, we set two disjoint deterministic cyclic routes to obtain, separately, the anonymity of senders and recipients. This allows us to modulate the size of these predetermined cyclic routes to manage the trade-off between privacy level and latency.

Recall that, we are in a specific situation in which no general-purpose connection-oriented communication is required, but only the anonymous exchange of a few short asynchronous messages. For example, in the domain of proximity-based services, we can accept that proximity testing is performed within order of magnitude of minute.

The structure of the paper is the following. In Section 2, we analyze the related literature. Some background notions useful to better understand our proposal are reported in Section 3. A new anonymity protocol for short communication in social networks is presented in Section 4. After presenting the protocol, in Section 5, we motivate the need for a new approach by comparing it with existing solutions taken from the field of anonymous communication networks. In Section 6, we apply this protocol to implement three proximity-based services. In Section 7, we perform a cost analysis and provide a prototype also used to validate experimentally our approach. The security of the protocol is analyzed in Section 8. Finally, in Section 9, we draw our conclusions.

2. Related Work

In this section, we briefly review the literature related to our work.

The main contribution of our paper is to jointly achieve the security requirements of anonymous communication networks (i.e., sender and recipient anonymity) and the privacy features required in the context of proximity-based services. Therefore, our paper can be related to two macro-topics, traditionally not intersecting with each other: (i) *anonymous communication networks* and (ii) *privacy-preserving proximity testing*, which is the chosen application domain.

(i) Regarding *anonymous communication networks*, among all the possible approaches [1], our solution can be related to anonymous P2P overlay networks. Onion [18], Crowds [19], AP3 [20], and Octopus [21] fall within this class of protocols. All the above approaches do not provide protection against the global passive adversary, which is the goal of our paper.

In alternative to secure-multi-party-based protocols, like DC-nets [2], the inclusion of cover traffic is needed to obtain protection against the global adversary [11].

P2P anonymous routing protocols including cover traffic can be referred to two main approaches: *buses* [15, 16, 17], and *mixnets* [12, 13, 22, 23], whose original definition not including cover traffic has been given in [24]. An approach similar to *buses* is proposed in [25]. However, their method [25], as clearly stated in the paper, does not provide protection against a global passive adversary, because the observation of the initiator breaks sender anonymity.

Concerning DC-nets, it is easy to realize that they are not applicable to our context (i.e., social networks), because they require that all the users are involved in every run of the protocol and leverage pairwise shared keys between the participants. This leads to high latency and low scalability.

As highlighted in the introduction, our solution relies on the approach of fixed anonymizing predetermined routes of *buses*. Recall that, according to this approach, all the nodes participating in the communication are included in an Eulerian path. A sort of bus turns in this Eulerian path. When a node A wants to communicate with a node B, first it has to wait for the bus and then it injects the message into the bus. The message will reach B when the bus reaches B. Therefore, the average latency is determined by the number of hops of half of the Eulerian path. The worst case occurs when the bus has to travel the entire Eulerian path. In the context we are considering in this paper (i.e.,

social networks), in which the number of communicating peers is potentially huge, the resulting latency would be prohibitive.

We solve the above problem by setting disjoint cyclic anonymizing routes for sender and recipient of size just sufficient to guarantee communication k-anonymity [26] (and, thus, minimizing the latency). Each anonymizing route plays the role of anonymity set. Then, the communication is enabled between these two anonymity sets, instead of two users. Unlike the general context of anonymous communication networks, in which the (anonymity-preserving) mapping between users and anonymity sets would be hard, social networks allow us to define a distributed hash table over the domain of social network identifiers to map users to anonymizing routes.

To the best of our knowledge, our paper is the first dealing with the problem of anonymous communication in social networks with protection against the social network platform (playing as a global passive adversary). [27, 3] are state-of-the-art papers proposing solutions for anonymous communication in social networks but their approach is not designed to resist global eavesdropping.

This paper is an extension of [28]. Several improvements are introduced in the current version. We illustrate next which are the improvements introduced by this paper with respect to its conference version.

First, we enrich the theoretical framework. Specifically, the main changes regard a more formal definition of the communication primitives and the introduction of the τ-safety notion. This is a probabilistic notion that allows us to expect that the required number of users in the rings is guaranteed. Moreover, we address the problem of identity management by providing an eIDAS-based authentication mechanism [29] allowing the user to obtain the IBE private key from the PKG. Finally, we implement in a more concrete form the notion of application domain only abstractly defined in [28], also by describing how domains can be built on the basis of the selected application context.

Another meaningful innovation regards the practical application of the anonymous routing protocol to the domain of proximity-based services. Indeed, in [28], such a domain is just mentioned as a possible application but no concrete implementation is provided. In Section 6 of this paper, we include a grid-based approach allowing the users to modulate the distance in which the proximity services have to be provided, and three different proximity-based services exploiting the anonymous communication protocol. In addition, in this paper, we include a cost evaluation not provided in [28], and we provide a prototype of the most complex proximity-based service treated in the paper. We choose to implement this service because, due to its complexity, it is the most significant to execute an experimental performance validation. In fact, this prototype is used for experiments in Section 7.3. We remark that, in [28], only a preliminary experimental evaluation has been provided, based on specific Twitter APIs and just concerning communication primitives (not proximity services). Finally, the security analysis of this work is more formal and detailed with respect to [28] and takes into account all the aspects related to the proximity-based protocol not included in [28].

(ii) The techniques for *privacy-preserving proximity testing* fall within the more general class of location-based services [30]. Proximity-testing approaches can be classified into three groups.

Grid-based. According to this approach, a large geographical area is organized as a grid, identifying (possibly overlapping) cells of a given shape. In the most common approach, a user who wants to disclose their proximity with another user, sends the service provider the cell identifier in which they are located in encrypted form. In the literature, several papers follow this approach [6, 31, 32].

Tag-based. This approach leverages the spatial-temporal location tags present in a specific area to allow the service provider to detect proximity. Unpredictable and unique tags can be implemented by employing a physical infrastructure or can be obtained by capturing some environmental features, such as Bluetooth IDs, WiFi IDs, military codes in GPS, audio signals, and atmospheric gases. The papers [7, 33, 34, 35, 36, 37] are good examples of this category. Among these, [36, 37, 38] are the most related with our proposal because they address the problem of proximity testing within social networks as our paper. However, [36, 38] achieve a privacy level less robust than our approach, since the service provider is aware of the fact that two users are performing a proximity test, thus resulting in considerable privacy leakage. [37] requires an external independent secure channel to exchange some information among the users. Therefore, it does not offer a solution fully lying within the social network.

Encryption-based. The papers falling in this class of approaches face the problem mostly by using secure multi-party computation or homomorphic-based protocols [8, 9, 39, 40, 7]. In this case, two users can perform the privacy-preserving proximity test without revealing to each other and to the provider their position. At the end of the execution of the protocol, the provider does not know anything about the result of the proximity test.

To the best of our knowledge, besides those described above ([36, 37, 38]), there is no relevant work studying how to provide privacy-preserving proximity-testing in social networks. Our paper achieves this goal by applying to social networks some ideas taken from the field of anonymous routing.

The idea of applying anonymous routing to proximity-based services in social networks was also explored in [41]. However, there are relevant differences between [41] and the present paper. Indeed, [41] adopts a simpler anonymous routing protocol which assumes that the social network identifier of the recipient is a priori known to the sender. Moreover, the organization of the grid in [41] is much simpler than this paper, because it does not support the modulation of the range of action of proximity testing. Conversely, in this paper, we propose a new hierarchical spatial index, called *shifted quad tree* allowing the user to choose the distance within which proximity testing is performed. Moreover, a relevant difference exists between the two papers also in terms of the number and types of supported proximity-based services. In fact, [41] just includes a rough version of symmetric and asymmetric testing. Finally, [41] includes a very brief experimental validation and does not report

any cost evaluation.

In this paper, we focus on proximity-based services aiming to detect the proximity between users or between a user and a target. The approaches aimed to preserve privacy when people's profiles are matched to each other or to a given target [42, 43, 44] are not treated but could be combined with our approach.

3. Background

In this section, we provide some background notions about *Identity-based Encryption (IBE)* and *Anonymity* useful for the comprehension of our protocol.

Identity-based Encryption. Identity-based encryption (IBE) is a type of public-key encryption in which the public key of a user is represented by some unique information associated with the user's identity. As we will see next, in the configuration of our protocol, the identity of the user is composed of the user's name, surname, and email address. In IBE, each user may encrypt a message for another user without requiring the public key to any external party, by directly using the information associated with the identity of the other user.

Formally, an IBE scheme is composed of four algorithms:
Setup(k): it takes as input a security parameter k and outputs a master secret key MSK and master public key MPK.
Extract(MPK, MSK, ID): it takes as input the master public key MPK, the master secret key MSK, and a parameter ID representing the identity of a user. It outputs a private key d associated with the user's identity ID.
Encrypt(MPK, ID, M): it takes as input the master public key MPK, a parameter ID representing the identity of a user, and a message M. It outputs a ciphertext C intended for the user with identity ID.
Decrypt(MPK, C, d): it takes as input the master public key MPK, a ciphertext C, and a private key d. It outputs the decryption M of the ciphertext C.

These four algorithms are used as follows.

A trusted third party, called Private Key Generator (PKG), is involved. Preliminary, in the setup phase, the PKG invokes Setup(k) to obtain MPK and MSK. MPK is provided to all the users and MSK is kept secret by PKG. A user U with identity ID_u, who wants to obtain a secret key associated with ID_U, contacts the PKG, which invokes Extract(MPK, MSK, ID_U) to obtain d_U, and sends it to U. Clearly, the PKG sends d_U after verifying the identity of U, for example through the intervention of an Identity Provider [45]. Suppose another user Y wants to send a message M to U (whose identity ID_U is known to Y). Y has to invoke Encrypt(MPK, ID_U, M) to obtain the ciphertext C and then can send C to U. Eventually, U invokes Decrypt(MPK, C, d_U) to retrieve the message M. Observe that Y, during the encryption process, does not interact with any party. This works in favour of anonymity.

In our protocol, we require that the adopted IBE scheme is anonymous. This means that it is not possible from the ciphertext to retrieve the identity of the recipient. The schemes [46, 47] satisfy our requirements.

Anonymity. The notion of anonymity is widely analyzed in [48]. To ensure the anonymity of a specific element, it is necessary to refer to an appropriate set of elements with, potentially, the same attributes. This leads to the following general definition: "Anonymity of a subject means that the subject is not identifiable within a set of subjects, the anonymity set."

The elements within the anonymity set depend on the type of item under consideration. For example, if the goal is to protect the privacy of an individual performing a certain action, an anonymity set is formed by individuals that are possible actors.

In our application, the item to protect is the communication. Therefore, we have to refer to the following definitions.
Sender anonymity: the condition in which the adversary cannot *sufficiently* identify the sender in a set of potential senders.
Recipient anonymity: the condition in which the adversary cannot *sufficiently* identify the recipient in a set of potential recipients.
Relationship anonymity: the condition in which the adversary cannot *sufficiently* identify that a sender (in a set of potential senders) and a recipient (in a set of potential recipients) are communicating.

The term "sufficiently", used in the above definitions, means that there exists a suitable threshold to quantify the desired anonymity. In our paper, we adopted the notion of *communication k-anonymity* given in [26], in which the adversary can only narrow down the possible senders or recipients to a set of k users.

It is easy to see that sender anonymity or recipient anonymity implies relationship anonymity.

4. The anonymity communication protocol

The aim of this section is to provide an anonymity communication protocol in social networks implementing the following three *communication primitives*:

- **P1: anonymous sending to explicit recipient.** This primitive is used by a sender A who wants to remain anonymous when communicating with an explicit recipient B. The explicit recipient can also coincide with the social network provider itself.

- **P2: response from an explicit recipient to an anonymous sender.** This primitive is used by the explicit recipient B of Primitive **P1** to respond to the anonymous sender A by preserving the anonymity of A.

- **P3: anonymous sending to anonymous recipient.** This primitive is used by a sender A to communicate with a recipient B in such a way that both remain anonymous. Observe that the response to this primitive can be obtained by invoking the primitive itself in the opposite direction.

These three primitives represent the building blocks of the proximity services described in Section 6.

The rest of this section is devoted to the provision of a concrete mechanism implementing them.

4.1. Identity management

In our solution, the way in which identities are managed is critical, because the objective we pursue is anonymity. In this section, we treat this aspect, which is independent of how anonymous communication primitives are implemented (which we describe in Section 4.7).

Each user is associated with two *identities*. The *real identity* RI of a user is composed of three attributes: name, surname, and email address.

The *SN identity* SI of a user is obtained by applying a cryptographic hash function h_R to the real identity (i.e., $SI = h_R(RI)$).

SN identities are used as identifiers in the social network. Therefore, the URL of the profile of a user is derived from their SI.

We assume that a user A who knows B, also knows the real identity of B and, thus, can retrieve their SN identity (needed for the communication) without leveraging SN. The autonomy of the sender in this task is necessary to maintain sender anonymity.

However, this is not enough when encryption to achieve message confidentiality is enabled. Indeed, as the preliminary symmetric-key exchange among all possible pairs of users is not feasible, public-key encryption should be used. On the other hand, even a PKI to manage public keys would threaten anonymity, when the initiator of a communication contacts the PKI to obtain the public key of the recipient. To avoid this, we adopt an anonymous IBE (described in Section 3) defined on the domain of the real identities. This way, a user A just has to know the name, surname, and email address of the recipient B, to encrypt a message being sent to B, without compromising sender anonymity when contacting the PKI. We observe that the adoption of an anonymous IBE instead of a standard IBE is necessary in our case. Indeed, in a standard IBE, the identity of the recipient is in plain text in the encrypted message (this is not the case with anonymous IBEs). This would break recipient anonymity.

To obtain the private key necessary to decrypt a message encrypted for a given real identity $\langle N, S, E\rangle$ (name, surname, and email address), a user pretending to be the recipient has to prove to the PKG that they control the email address E. This could be done by sending a challenge to this email address, which the user has to solve. Observe that this could be the classical *confirmation email* received when a user registers with the most online services. Clearly, this procedure is performed just once in the set-up phase to obtain the IBE private key that will be used to decrypt all the messages intended for the user.

The way in which SN identities are obtained allows us to have a one-to-one mapping between the real identity of the recipient and their profile in the social network, provided that the email address of the real identity is kept under the control of the legitimate user. To make the email account violation not dangerous for our system, a more secure proof should be provided to the PKG to demonstrate their real identity. In a concrete scenario, we could think of adopting a secure public digital identity system, like a system compliant with eIDAS regulation in the European Union [29], or a robust Self Sovereign Identity system, like that designed in the EBSI framework [49].

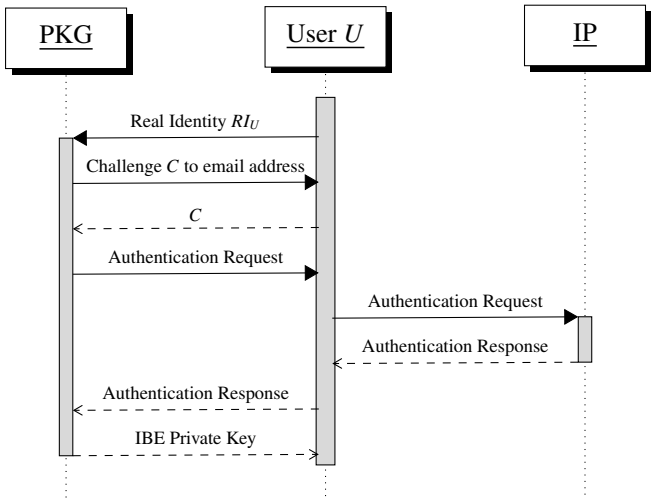

Figure 1: eIDAS-based authentication procedure with PKG to obtain the IBE private key

For the sake of clarity, we detail the solution in the case of eIDAS-based identity proof.

A user U with real identity $RI_U = \langle N_U, S_U, E_U\rangle$ contacts the PKG to obtain a private key associated with RI_U. First, the PKG sends a random number C (challenge) to the email address E_U, which U turns back to PKG by proving their control on E_U.

Furthermore, to prove the possession of N_U and S_U, an identity provider IP is required, which is a trusted third party to which U is preliminarily registered by means of an identification process with a high level of assurance. A typical identity system compliant with eIDAS leverages a federated authentication protocol like SAMLv2, in which the service provider SP requests a valid signed assertion (embedded in an authentication response) from the IP, proving the identity of the user. In our case, the PKG would play the role of service provider. The above considerations allow us to conclude that, at least in the European Union, the identity management we consider in our solution could be concretely adopted with a high security level, by considering that eIDAS enforced the Member States to have an interoperable digital identity system since 2016.

The flow of this authentication procedure is depicted in the sequence diagram reported in Figure 1

To conclude, we recall that Primitive **P1** allows the sender to communicate with an explicit recipient, possibly coincident with the social network provider. In the formal definition of such a primitive, given in Section 4.7, we pass as input the real identity of the recipient. Therefore, to allow anonymous sending to the social network provider, we assume that also the social network provider has a real identity. In practice, it simply means that it is registered with the PKG and that it obtains the IBE private key to decrypt the messages intended for it. In this case, the real identity is not composed of name, surname, and email address but it just consists of a simple public string associated with the social network (e.g., the URL of the social network).

4.2. Application domains

Anonymity is obtained through a cooperative approach, involving the users of SN. The collaboration is thought as a special feature of certain *application domains*, each forming a collaboration community. The users of each community require anonymity when a certain type of service is delivered.

The formal definition of *application domain* is the following.

Definition 4.1. *An application domain A is a tuple $\langle ID_A, N_A, k_A \rangle$, where $ID_A \subseteq \mathbb{N}$ is a finite set of the SN identities of the involved users, N_A represents the cardinality of ID_A, and $k_A \in \mathbb{N}^+$ is said privacy level.*

The meaning of the privacy level regards the objective of our protocol, which is anonymity. Anonymity regards the sender and recipient of a message (thus also relationship) and is reached by requiring a sufficient degree of uncertainty.

Given an application domain A, the privacy level k_A represents just the obtained degree of uncertainty, in the sense that the adversary can identify an item (sender or recipient) with probability not greater than $\frac{1}{k_A}$.

As introduced earlier, we are considering services in which anonymous communication between users or between users and SN is required. Observe that, even with robust anonymous communication primitives, the knowledge that some users are more likely to communicate between them rather than with other users, would lead to a break of anonymity anyway.

Therefore, we say that an application domain is *well-formed* if, from the point of view of an adversary attempting to break anonymity, all the users have the same probability to communicate between them or with SN.

To achieve this feature, the building process of the application domains has to take into account the type of delivered service and the background knowledge of the adversary (possibly, SN itself) about the users leveraging such a service.

For example, if, for the specific application, the geographical location (e.g., the IP zone) is a quasi-identifier, then the domain has to be identified on the basis of geographical information. Consider the case of a national survey not requiring more specific geographical information. We expect that the domain can include only profiles belonging to the national territory. In general, depending on the application, privacy guarantees are achieved by taking into account different constraints, possibly leveraging privacy notions like l-diversity [50] and t-closeness [51]. In Section 6, we study the application of our anonymous communication primitives to the case of proximity-based services. Therein, we describe also how, in the domain building process, the above privacy guarantees are obtained.

From now on, we assume that the application domains are well-formed.

4.3. The ring schema

In this section, we describe the structural elements of the model which the solution relies on.

We start with the definition of *ring schema*.

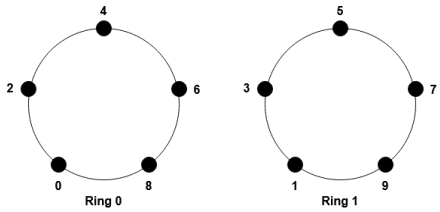

Figure 2: Ring schema for $\alpha = 2$ and $k_A = 5$

Definition 4.2. *Given an application domain $A = \langle ID_A, N_A, k_A \rangle$, a number n_A such that $n_A = \alpha \cdot k_A$ (for any $\alpha \in \mathbb{N}^+$), and the set $D_A = \{0, \ldots, n_A - 1\}$, the ($\alpha$-)ring schema (of A) is the set of the equivalence classes each containing all the elements of D_A congruent modulo α. Each class is called (α-)ring (of A). A ring is identified by the canonical representative of the equivalence class. Given an element $x \in D_A$, we denoted by $ring(x)$ (with $ring(x) \in D_A$) the canonical representative of the ring which x belongs to. The elements of a ring are called nodes.*

It is easy to see that the cardinality of an *α-ring schema* is α and that the identifiers of the classes are $0, 1, \ldots, \alpha-1$. Moreover, all the classes are of cardinality k_A. Observe that the ring schema is completely defined by the parameters N_A, k_A, and α. The parameters of the adopted ring schema are notified to all the users of the application domain.

Example 4.1. *An example of ring schema for $\alpha = 2$ and $k_A = 5$ (then $n_A = 10$) is reported in Figure 2. Therein, being $\alpha = 2$, we have 2 rings (i.e., ring 0 and ring 1) each including $k_A = 5$ nodes. Moreover, for example, $ring(2) = 0$ and $ring(9) = 1$.*

From now on, throughout the paper, assume given an application domain $A = \langle ID_A, N_A, k_A \rangle$ and its α-ring schema, for a given α.

The ring schema is the basic notion of our solution, because it allows us to identify a topological structure suitable to support a cover-message-based mechanism which *hides* senders and recipients through the mutual collaboration of the users of the application domain. To do this, the so far abstract nodes of rings have to be associated with users of the domain. This is done by uniformly mapping (through a classical hash function h) the set of SN identities of a domain ID_A to the set of nodes in D_A.

Definition 4.3. *A user mapping on the α-ring schema of A is any function $h : ID_A \rightarrow D_A$ such that the probability that $h(SI_X) = h(SI_Y)$ (for each $SI_X, SI_Y \in ID_A$, such that $SI_X \neq SI_Y$) follows the uniform distribution.*

From now on, we consider, as a user mapping, the hash function h such that $h(SI_X) = SI_X$ **mod** n_A, for each $SI_X \in ID_A$. It is well known that this function allows us to fulfil the condition required by Definition 4.3. However, a different user mapping could be adopted, also by taking into account possible specific characteristics of the set ID_A.

Also the user mapping is notified to the users.

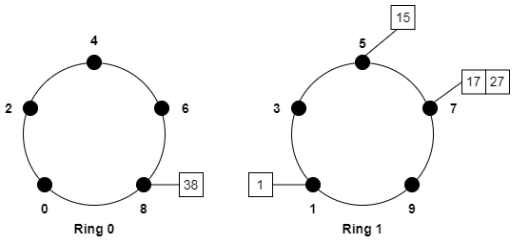

Figure 3: Example of user mapping.

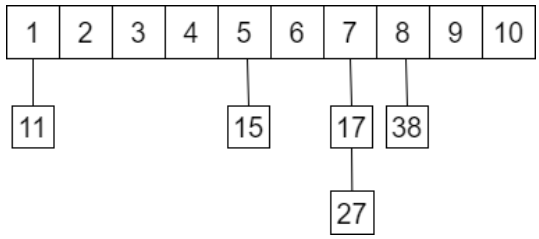

Figure 4: Example of hash table referred to the user mapping in Figure 3

Example 4.2. *By referring to Example 4.1 (in which $\alpha = 2$, $k_A = 5$), suppose $N_A = 5$ and $ID_A = \{17, 15, 38, 11, 27\}$. The users (whose SIs are in ID_A) will be mapped to the nodes of the ring* 0 *and ring* 1*, as depicted in Figure 3.*

The users with SIs 17 *and* 27 *are mapped to node* 7*, the user with SI* 15 *is mapped to node* 5*, the user with SI* 38 *is mapped to node* 8*, and the user with SI* 11 *is mapped to node* 1*. The other nodes are not associated with any user.*

We introduce now another notion, allowing us to characterize each ring as a sort of *virtual cyclic circuit* (thus motivating the name we choose for the rings).

Definition 4.4. *we define the function $f_A : D_A \rightarrow D_A$ such that $f_A(x) = x + \alpha$* **mod** *n_A.*

Example 4.3. *By referring again to Example 4.1, $f_A(2) = 2+2$* ***mod*** *$10 = 4$, $f_A(3) = 3 + 2$* ***mod*** *$10 = 5$, and $f_A(8) = 8 + 2$* ***mod*** *$10 = 0$*

On the basis of both the function f_A and the user mapping, the rings represent virtual cyclic circuits of groups of users. In other words, if the ring v_0 is $\{v_0, \ldots, v_{k_A-1}\}$, where $v_i < v_j$ for $0 \leq i < j \leq k_A - 1$, then $f_A(v_i) = v_{i+1}$, for each $0 \leq i < k_A - 1$ and $f_A(v_{k_A-1}) = v_0$. Moreover, with each v_i ($0 \leq i \leq k_A - 1$), a set of users, identifiable by reversing the function h, is associated. The multiplicity of users associated with nodes has the scope to give redundancy to these virtual cyclic circuits (as we better explain in the next subsection). Observe that, through the user mapping, each user belongs to exactly one node in a ring of a given domain.

The user mapping is not materialized by the users. As we will see later, they just could need to compute some of its values. Instead, SN stores a hash table H (based on h) materializing the user mapping in such a way that if a user requires to know the group of SN identities mapped to a given node v of a ring, then SN can efficiently provide the correct answer just by accessing the hash table at the index v.

Example 4.4. *The hash table referred to the user mapping of Example 4.2 is depicted in Figure 4.*

Therefore, a user, starting from the knowledge of a given SI, say SI_X, can determine which is the node associated with this SI just by computing $h(SI_X)$. Then, they can calculate the entire sequence of nodes of the ring (for example, the node at distance j from the node in which is mapped SI_X is obtained as $f_A^j(h(SI_X))$), and can retrieve from SN the list of SIs associated with any node of the ring. We can assume that each user always knows the SIs associated with each node of the ring which the user belongs to. Moreover, for each of these SIs, the user knows also their public keys.

This information, i.e., the set of pairs (SI-public key), is called *configuration of the ring*. Any change of the configuration of the ring is communicated by SN to all the users of the ring, as we will see in Section 4.5.

4.4. Redundancy

The ring model so far presented implicitly assumes that all the users mapped by the hash function h to a ring are alive and collaborative. Under this assumption, k_A actually represents the guaranteed privacy (i.e., anonymity) level, as a ring represents the anonymity set of a sender or a recipient. As this assumption is not realistic, in this section we relax it. Specifically, we show how to include into the ring the right level of redundancy to guarantee a given level of anonymity. Being the problem we are considering inherently probabilistic, we give to the term *guarantee* a probabilistic meaning. Therefore, we set a (suitably high) probability threshold τ, and we say that an event is *sufficiently* guaranteed if it occurs with probability not below τ.

Moreover, we assume that, if we pick a user, the probability that they are alive and collaborative is p.

It can be realized that the probability that at least one of r users is alive and collaborative is $1 - (1 - p)^r$ (corresponding to the probability of the complementary event that all the r users are not available). This supports the following definition.

Definition 4.5. *We define the redundancy level of A, denoted by r_A, as the minimum value r such that $1 - (1 - p)^{r_A} \geq \tau$ (i.e., it is sufficiently guaranteed that at least one of r_A users is alive and collaborative).*

The redundancy level is useful to introduce another definition, which takes into account the distribution of the users over the ring. The aim is to represent the fact that a given ring schema offers a level of privacy not below k_A.

Definition 4.6. *We say that the α-ring schema is τ-safe with respect to a given user mapping h, if for each α-ring y ($0 \leq y \leq \alpha - 1$), $|\{x \in ID_A : ring(h(x)) = y\}| \geq k_A \cdot r_A$.*

It is easy to see that if the α-ring schema is τ-*safe*, for any α-ring it is sufficiently guaranteed that the ring includes at least k_A users alive and collaborative. Therefore, the ring can play the role of anonymity set with privacy level k_A. The value $k_A \cdot r_A$ is called k_A-*anonymity threshold*.

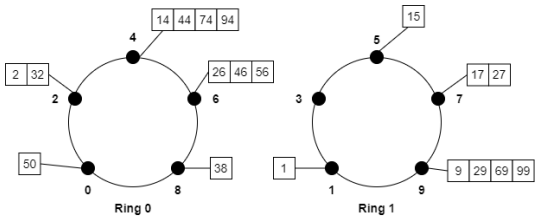

Figure 5: Ring schema τ-safe.

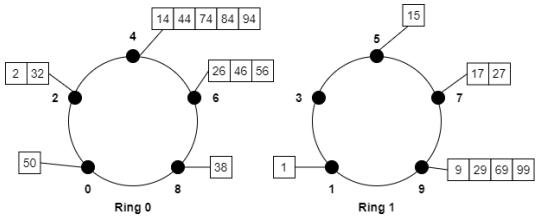

Figure 6: Ring schema after the join of the user with SI 84.

Example 4.5. *Suppose $p = 0.99$ and $\tau = 0.999$. The redundancy level (see Definition 4.6) is $r_A = 1.5$. It is easy to see that the ring schema with the users mapped as in Figure 3 is not τ-safe since the anonymity threshold is $5 \cdot 1.5 = 7.5$. Indeed, the ring 0 has just 1 user, and the ring 1 has just 4 users. An example of ring schema τ-safe is reported in Figure 5. Here, $N_A = 19$, $k_A = 5$ and $\alpha = 2$.*

4.5. System update

The previous definitions do not take into account possible updates regarding an application domain. Updates, which are joins and leaves of users, can be managed as follows.

User Join. When a new user U with real identity RI_U joins the application domain A, an SI $SI_U = h_R(RI_U)$ is assigned to this user, and it suffices for SN to include the new user in the hash table H at the index $h(SI_U)$. As this addition cannot threaten the number of expected users alive in the affected ring (i.e., the ring including the node $h(SI_U)$), the join does not impact the ring schema.

The new SI SI_U and the public key of U are included in the new configuration of the ring. The new configuration is notified, through SN, to all users in the affected ring.

No further action is required.

Example 4.6. *We take the ring schema of Figure 5. If the user U with $SI_U = 84$ joins the application domain, then SI_U is simply mapped to the node 4. The resulting τ-safe ring schema is reported in Figure 6*

User Leave. When a user U with SN identifier SI_U leaves the application domain A, SN has to remove the user from the hash table H, at the index $h(SI_U)$. The number of users is obviously updated as $N'_A = N_A - 1$. Similarly to the case of join, the users of the affected ring are notified about the changes occurred in the ring, in such a way that the local information about the configuration of the ring is kept coherent. However, the event might threaten the fact that the ring schema is τ-safe. In this case, as the ring goes below the k_A-anonymity threshold,

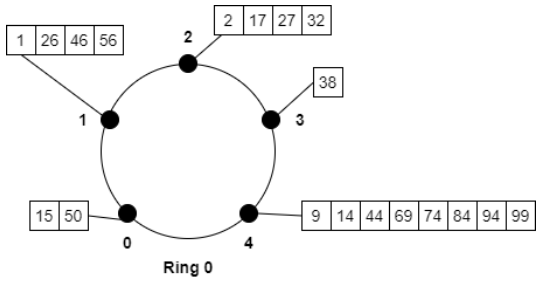

Figure 7: New ring schema after the user with SI 29 leaves the ring 1 of Figure 5

we have to restore the τ-safety property by changing the hash function h. This is done by properly decreasing n_A. Therefore, SN finds $\alpha' < \alpha$ (and, then, $n'_A < n_A$) such that, for each α'-ring y ($0 \leq y \leq \alpha' - 1$), $|\{x \in ID_A : ring(h(x)) = y\}| \geq k_A \cdot r_A$. This implies that SN has to redistribute the users in the new hash table (of size $n'_A = \alpha' \cdot k_A$), with computational cost $O(N'_A)$. SN has to notify all users in the domain the updated parameters of the ring schema (i.e., N'_A and α') and the new ring configuration. No computation overhead is required user-side.

Example 4.7. *Consider the ring schema of Figure 6. With $r_A = 1.5$, the k_A-anonymity threshold is $5 \cdot 1.5 = 7.5$ and both the rings 0 and 1 have a number of users alive exceeding this threshold.*

Suppose the user with SI 29 leaves the ring 1. In this case, the ring 1 will have $7 < 7.5$ users. The value of n_A has to be decreased. The only possibility is to choose $\alpha' = 1$ and $n'_A = k_A = 5$. The new ring schema is reported in Figure 7

Even though the worst case for leaves triggers a server-side (albeit linear, in the number of users) computational overhead, we can argue that, in real-life cases, communities tend to grow, or, at worst, joins and leaves are balanced. Therefore, with proper "safety margins" applied to the set value of α, the above worst case happens very rarely. Observe that also the growth of the number of users might trigger the resizing of the hash table even though this is not necessary for the correctness of the ring schema. Indeed, it could be opportune not to have rings with an actual privacy level much higher than the required value.

Finally, concerning system updates, one could think that drastic changes of the system not corresponding to changes of the communication characteristics can enable classical intersection attacks, thus breaking anonymity. However, this is not the case, because we only consider short communications, whose lifetime is certainly much less than the lifetime of the ring structure.

4.6. Cover-message mechanism

At this point, we describe the cover-message mechanism mentioned earlier, which is at the basis of the anonymity service provided by our solution.

Consider a ring $\{v_0, \ldots v_{k_A-1}\}$. With a certain rationale, we choose r_A users belonging to the nodes of the ring responsible

for maintaining the circulation of dummy messages called *tokens*. r_A is the value that determines the k_A-anonymity threshold (i.e., $k_A \cdot r_A$). Therefore, it is sufficiently guaranteed that at least one responsible user is alive. Now, we define how the token is built. It is a fixed-length message with three fields: $\langle \bar{M}, \bar{D}, B \rangle$, where $\bar{M}$ is a message possibly encrypted, $\bar{D}$ is an encrypted SN identity, B is a bit indicating if the token is empty ($B = 0$) or filled ($B = 1$). Observe that $\bar{M}$ may also include a dummy message.

The exact meaning of the above fields will be clarified below with the description of the communication primitives. Each token turns in the ring in which it has been generated, by crossing, for each node, any alive user associated with this node. This is done according to the increasing value of the corresponding SIs. To formally describe the above mechanism, we need the following definition, introducing the notion of *next* alive user for a given user in a ring.

Definition 4.7. *Given a node v of a ring with at least one alive user, we denote by $first(v)$ the lowest SI associated with v and by $last(v)$ the highest SI associated with v. The closeness between two alive users with SIs SI_X and SI_Y belonging to a ring (denoted by $closeness_A(SI_X, SI_Y)$), is recursively defined as follows:*

- $closeness_A(SI_X, SI_Y) = 0$, *if* $SI_X = SI_Y$;
- $closeness_A(SI_X, SI_Y) = |\{SI_Z \in ID_A \mid SI_Z \neq SI_X \text{ is alive}, h(SI_Z) = h(SI_X), SI_X \leq SI_Z \leq SI_Y\}|$, *if* $h(SI_X) = h(SI_Y)$ *and* $SI_X < SI_Y$;
- $closeness_A(SI_X, SI_Y) = closeness_A(SI_X, last\ (h(SI_X))) + closeness_A\ (first(h(SI_Y)), SI_Y) + j \cdot (N_A - 1)$ *for the least* $j > 0$ *such that* $f_A^j(h(SI_X)) = h(SI_Y)$, *otherwise.*

We define the function $next_A : ID_A \rightarrow ID_A$ as follows. For any user X alive with SN identity SI_X, $next_A(SI_X)$ is the SN identity SI_Y of the user Y (alive) of the ring such that $SI_X \neq SI_Y$ and the closeness between SI_X and SI_Y is minimum.

In words, the next user of a user with SN identity SI_X is the first alive user who is encountered by moving first in the node of the ring $h(SI_X)$ in the direction of increasing SIs, and then (if there is no alive user in $h(SI_X)$ with SI higher than SI_X) to the closest node according to the function f_A with alive users and, therein, by taking the user with the lowest SI.

According to Definition 4.7, the token is sent by an alive user with SN identity SI_X who received it to the user with SN identity $next_A(SI_X)$, and proceeds in the ring with the same rule. Therefore, it is not sure that the token moves from the node v_i to the next node $f_A(v_i)$, because a jump is possible (in the case no alive user is present in the node $f_A(v_i)$).

At each hop, the token is encrypted by the current user with the public key of the next user. Thus, an external eavesdropper cannot distinguish an empty token from a filled token. For efficiency reasons, the token is encrypted with a symmetric on-the-fly key which is, in turn, encrypted with a public key and sent along with the token. When a node receives the token, first it decrypts the symmetric key and then the token. For the sake of presentation, from now on, when we refer to the public-key encryption, we mean the above procedure.

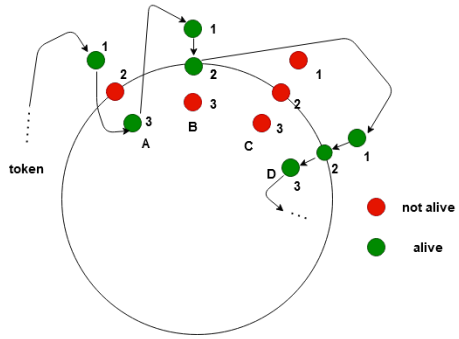

Figure 8: Example of fragment of the route of a token in a ring.

In Figure 8, an example of route fragment followed by a token is depicted. In the figure, we highlight only 4 nodes of a ring, each composed of 3 users (in general, this number could vary among nodes). Green circles represent alive users, while red circles denote non-alive users. The token turns in the ring according to the function *next* of Definition 4.7.

Periodically, SN publishes a cross-domain random $R \in \mathbb{N}^+$, with a certain rounding protocol obtained through a PRNG verifiable by the users. R, implicitly identifies a user per ring, called *bridge user*, as follows. To identify the bridge user of their ring, a user with SN identity SI_X has to find the alive user with the lowest SI belonging to the node $f_A^j(v_y)$, such that $j \geq 0$ is the minimum value and at least one alive user is in $f_A^j(v_y)$ and $v_y = ring(h(SI_X)) + R \cdot \alpha$ **mod** n_A. Observe that all the nodes of a ring identify the same bridge user. The bridge user is responsible for sending the messages outside the ring (playing the role of *exit user*) or for injecting into the ring the messages coming from outside (playing the role of *entry user*) as explained in the next section.

4.7. Communication primitives

At this point, we are ready to formally define the communication primitives supporting privacy-preserving services. We have three primitives, defined as follows.

- **P1: anonymous sending to explicit recipient.** This primitive is invoked by a user U and receives as input a message M and a real identity RI_D. The message M (possibly encrypted) is forwarded from U to the user D with real identity RI_D by keeping U anonymous.

  U knows the SN identity SI_X and the public key PK_X of the bridge user X of the ring in which U is located. In this primitive, we say that X plays the role of *exit user*.

  First, U derives the SN identity SI_D associated with RI_D i.e., $SI_D = h_R(RI_D)$. Observe that this operation does not involve SN, so anonymity of U is not compromised.

  Then, U waits for the earliest empty token of the ring (recall that a user, when receives a token, has to decrypt it to decide if forwarding or processing it, because the token is encrypted with its public key) and fills its fields $\langle \bar{M}, \bar{D}, B \rangle$ as follows: $\bar{M} = M$, $\bar{D} = E(PK_X, SI_D)$ (i.e., the encryption

for the bridge user X of the SN identity of the destination D), $B = 1$ (that represents the fact that the token is filled).

We denote by T the so obtained token. Now, U encrypts the filled token T with the public key of the user Z with SN identity $SI_Z = next_A(SI_U)$. Then, the token is sent to Z. The token turns in the ring until the user X, who is the only user able to decrypt $\bar{D}$, thus obtaining SI_D.

At this point, X forwards M to SI_D.

Finally, X sets B to 0 and $\bar{M}, \bar{D}$ to random values and forwards the token in the ring to the user with SI equal to $next_A(SI_X)$.

- **P2: response from an explicit recipient to an anonymous sender.** This primitive is invoked by the destination D of Primitive **P1** to reply to the sender U in such a way that the latter remains anonymous. The primitive receives as input a message R (possibly encrypted). In addition, we assume as implicit input the SN identity SI_X of the bridge user X acting as exit user in Primitive **P1**.

  First, D sends R to X.

  X injects the response in the ring just by waiting for the earliest empty token and filling it with R. In this case, $\bar{M} = R$, $B = 1$, and $\bar{D}$ remains undefined. Then, the filled token turns in the ring until U receives R. This is the actual recipient of the response. U does not empty the token and just forwards it. This operation is done by X (for security reasons, as discussed in Section 8) when the token reaches them again by setting B to 0 (and the other fields to random values) and by further forwarding the token in the ring.

- **P3: anonymous sending to anonymous recipient.** This primitive is invoked by a user U and receives as input a message M and a ring identifier v_k. The message M (possibly encrypted) is forwarded from U to a user D with SN identity SI_D such that $v_k = ring(h(SI_D))$,in such a way that both U and D remain anonymous.

  We denote by X the bridge user, with SN identity SI_X and public key PK_X, of the ring in which U is located.

  As in Primitive **P1**, U waits for the earliest empty token of the ring and fills it by setting its fields $\langle \bar{M}, \bar{D}, B \rangle$ as follows: $\bar{M} = M$, $\bar{D} = E(PK_X, v_k)$, $B = 1$ (representing the fact that the token is filled).

  The token turns in the ring until the user X, who retrieves v_k.

  At this point, through the collaboration of SN, X identifies the bridge user Y of the ring v_k and forwards M to Y.

  Finally, Y injects the message in the ring as in Primitive **P2**, thus eventually reaching the actual destination D. As for Primitive **P2**, D does not empty the token, which will be emptied by Y.

  Observe that a possible reply of D to the message M sent by U can be done by using the same primitive.

5. Why to design a specific protocol

After presenting the anonymous protocol, we show, in this section, that the definition of a new specific protocol represents an actual added value. In other words, by considering the attempt to apply existing approaches taken from the field of anonymous communication networks, we show why to design an original anonymous routing protocol tailored to the considered scenario.

As mentioned in the introduction, there are two possible approaches in the literature that can be used to obtain communication anonymity against the global passive adversary: buses [15, 16, 17] and mixnets [12, 24].

In deterministic buses [15, 16], the route is a path involving all the nodes of the network. This results in intolerable latency when, as the case of social networks, the number of nodes is huge. Consider that, given an application domain $\langle ID_A, N_A, k_A \rangle$, for our method, the latency time is $\Omega(k_A)$, while for deterministic buses it is $\Omega(N_A)$. In real-life applications, we expect that $N_A \gg k_A$. Coherently with the experiments shown in Section 7.3, any single hop of communication takes a time of order of magnitude 10^{-1} seconds. Therefore, for a realistic domain of just 10^4 users, the latency time for a given message communication is 10^3 seconds, which is not acceptable for the considered applications (e.g., proximity testing). Our approach allows us to modulate the cardinality of the anonymity sets in order to find a good trade-off between privacy and latency, independently of the size of the application domain. As shown in Section 7.3, for a good privacy level (i.e., the cardinality of the anonymity sets) of 10^2 users, we obtain times of order of magnitude of minute for a worst-case anonymous communication (including a number of exchanged messages).

Consider now non-deterministic buses [17]. This technique leads to very high latency times, as analytically highlighted in the paper itself. Indeed, the delivery time follows an equation of the form $\frac{K_1}{\left(1-\left(\frac{n-2}{n-1}\right)^{K_2}\right)}$, where K_1 and K_2 are suitable constants and n is the number of nodes of the network, meaning that, for large values of n, we have huge latency time. Indeed, the simulation conducted in [17], which does not take into account the emulation over social networks, in a network with only 2048 nodes (and thus a maximum privacy level of order of magnitude 10^3) produces an average latency time of 20 minutes.

Regarding mixnets, we adopt a simplified yet general model extracted from [12], in which bi-directional cover traffic over any link of the overlay network is enabled (this is necessary to hide communications from the global passive adversary). The idea is to obtain the anonymity set by mixing the traffic at each hop of the communication and by hiding the real traffic inside cover traffic. This fan-out mechanism allows us to obtain that the cardinality of the anonymity set increases exponentially with the length of the communication path. In Figure 9, we represent a simple mixnet with a degree mixing 2 (i.e., the messages of 2 senders are mixed into a receiver at each step). This way, for a communication path of length l, the anonymity set resulting from the knowledge of a given receiver, has cardinality 2^l. However, to obtain this level of uncertainty, as clearly stated in [12], bi-directional cover traffic should be injected, over all

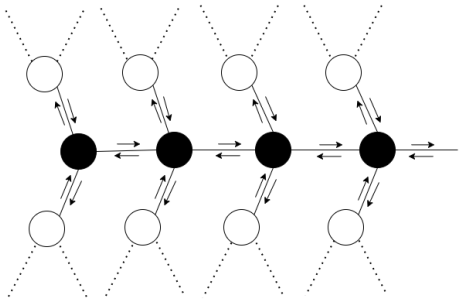

Figure 9: Mixnet with $n = 4$ and $m = 2$.

.

the links of the network. For simplicity, we assume that cover traffic is injected at a constant rate so that the amount of traffic can be represented just by the number of links in which it is injected. Let denote by m the mixing degree and by n the number of nodes participating in the mixnet. Note that, to achieve exponential growth of anonymity degree with the number of hops, we require that the mixing always involves new nodes, as depicted in Figure 9.

Hence, the number of links allowing us to protect the communication among n nodes is $(m + 1) \cdot n$ as the degree of each node is $m + 1$. For example, in Figure 9, the total number of links is $12 = 4 \cdot 3$, where $n = 4$ and $m = 2$ and the degree of each node is 3. As cover traffic is bi-directional, the estimation of the total amount of cover traffic is $2 \cdot (m + 1) \cdot n$. This means that the minimum required cover traffic is $6 \cdot n$, as, to enable the fan-out mechanism, $m \geq 2$ should hold.

In our approach, cover traffic corresponds just to tokens 1-directionally turning in the rings. Therefore, the number of links is exactly n, which is the measure we can use to represent the total amount of cover traffic. Moreover, when m is fixed to the minimum value, we reduce cover traffic of a multiplicative factor equal to 6. For higher values of m, the advantage increases.

About communication latency, we can say that, to achieve the same privacy level k_A in the mixnet, we have to set $l = log_m k_A$. Therefore, at the same privacy level k_A, the length of the communication path of our method is k_A, while the (average) length of the mixnet tunnel is $log_m k_A$. Therefore, the advantage we obtain in terms of cover traffic has a price in terms of communication latency. However, this is not critical for our application domains in which no low-latency connection oriented communication should be supported. Indeed, Section 7.3 shows that the privacy-preserving services implemented on top of our anonymous communication protocols are performed in reasonable times.

6. The proximity protocol

In this section, we show as, through the three primitives described Section 4.7, we can implement a proximity testing protocol protecting users' privacy against the global adversary. In particular, we provide three proximity-based services. The first is a service aimed to test the proximity of users who know each other. Roughly, we provide the privacy features to a service similar to *Facebook Nearby Friends*. We call this service `KN-service` (standing for *known nearby service*). The second service is used to test the proximity between users who do not know each other but make public some information (photos, preferences, etc.). The service allows detecting proximity of unknown users only on the basis of their agreement. This service extends the features given by services such as *Tinder*, by enabling the above privacy features. We call this service `UN-service` (standing for *unknown nearby service*). Finally, the last service regards proximity testing of a user with respect to a (static or moving) target. The privacy requirement is that the user remains anonymous also with respect to the target. This service extends services such as *Tripadvisor* or *BlaBlaCar*. We call this service `TN-service` (standing for *target nearby service*).

6.1. Grid organization

In this section, we provide some preliminary notions regarding the way in which our grid-based technique is organized. The standard approach used in grid-based techniques consists of the partition of the territory into *cells* of a certain shape (squares, hexagons, circles, etc.) possibly overlapping each other. We need a more sophisticated structure because we want to enable the modulation of the size of the searching area in which users want to perform their proximity test. Even though the technical detail of proximity testing is not clear, at this stage of the paper, it is intuitive to understand that the grid has a direct role in the implementation of the proximity testing. In other words, the cells induced by the grid will represent the area in which people are looking for near users or services. Therefore, to have just a fixed grid, whatever its shape and organization are, would not allow us to have flexibility in the size of the searching area. As a matter of fact, existing proximity services such as Tinder, allow the user to choose the size of the searching area between some meters to some kilometers (with a given granularity). To obtain the same feature, we design a hierarchical spatial index based on the concept of *quad tree* [52], in which also overlapping is enabled. We call this structure *shifted quad tree* (SQT). A quad tree is a tree in which each internal node has exactly four children. It can be used to partition a 2-dimensional area into regions of different sizes. Specifically, the entire area is associated with the root of the tree and it is partitioned into four regions, each associated with a child of the root. Recursively, each region is partitioned into four regions and so on. The last obtained regions are associated with the leaves of the tree.

An SQT implements, at each level, an overlapping mechanism of the square cells of the type described in Figure 10, obtained by taking two square grids (suppose, one black and the other red) initially coincident and by shifting the red one across the left-bottom diagonal for half diagonal of the square. This way, each user belongs exactly to two squares (one black square and one red square), and two users at a distance less than half of the length of the side of the square have at least one cell in common.

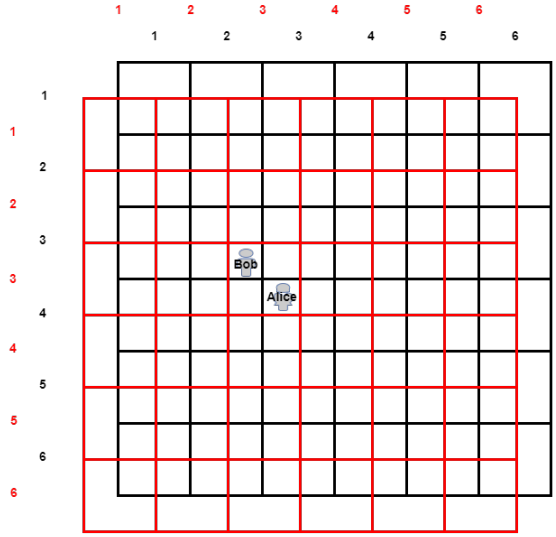

Figure 10: Overlapping mechanism.

.

For example, still referring to Figure 10, Alice belongs to the black cell $(4, 3)$ and the red cell $(3, 3)$, Bob belongs to the black cell $(3, 2)$ and the red cell $(3, 3)$. As they are at distance less than half of the length of the side of the square, they both belong to the red cell $(3, 3)$. Observe that, in the figure, the coordinates of the cells are replicated with different colors. Actually, the replication of coordinates indicated above reflects the fact that the red grid is thought as shifted from the position of the black grid. So, the red cell (x, y) is the *shifted cell* associated with the black cell (x, y).

We want to keep the same mechanism hierarchically, at each level of a quad tree, thus obtaining an SQT, which basically is a standard quad tree that includes a *shifted node* for each quad-tree node. The shifted node s of a quad-tree node n indexes, in the territory, a square that is the shifted cell of the square indexed by n.

This mechanism is resumed in Figure 11. In this figure, a restricted interest area (thus cutting some cells) and a three-level SQT are considered. About coordinates, we need to add two dimensions, one representing the level in the tree and the other needed to make explicit the fact that the coordinates are referring to a quad tree node or to a shifted node. We assume that the numbering of the coordinates starts from 1 for a square of type quad tree entirely included in the interested area. The coordinates are of the form $\langle k, i, j, t\rangle$, where k indicates the level (0,1,2, ...), i, j the position, and t the type between *(q)uad* and *(s)hifted*. For example, the cells $\langle 0, 1, 3, q\rangle$, $\langle 0, 1, 4, q\rangle$, $\langle 0, 2, 3, q\rangle$, and $\langle 0, 2, 4, q\rangle$ (representing four 0-level quad-tree nodes) are aggregated into one 1-level quad-tree node, with coordinates $\langle 1, 1, 2, q\rangle$. This node indexes the blue square with the same coordinates. Moreover, the shifted 1-level node associated with this node has coordinates $\langle 1, 1, 2, s\rangle$ and indexes the green cell with the same coordinates. Moreover, the cells $\langle 1, 1, 2, q\rangle$, $\langle 1, 1, 3, q\rangle$, $\langle 1, 2, 2, q\rangle$, and $\langle 1, 2, 3, q\rangle$ (representing four 1-level quad-tree nodes) are aggregated into the 2-level quad-tree node coloured in pink. The shifted 2-level node associated with this node is colored in orange.

Once a given level is set, say k, a cell of level k is identified by its center, said *centroid*. Each user, through a localization system, is able to identify the centroids of the two cells of level k in which they are located. Each centroid can be identified by the coordinates of the cell. Observe that, even though a given point in the space could be the centroid of different cells (belonging to different levels), the presence in the coordinates of the level allows us to uniquely identify the centroid.

Moreover, a user, for each level k, being located in two overlapping cells, one of type q and the other of type s, detects two centroids associated with the level k.

For example, the user represented in Figure 11 with a black dot, detects, level by level, the centroids $\langle 0, 4, 3, q\rangle$, and $\langle 0, 3, 4, s\rangle$, for level 0, the centroids $\langle 1, 2, 2, q\rangle$, and $\langle 1, 2, 2, s\rangle$, for level 1, and the centroids $\langle 2, 1, 1, q\rangle$, and $\langle 2, 1, 1, s\rangle$ for level 2.

The centroids are represented in the figure with yellow dots. Sometimes, a single dot represents multiple coincident centroids (in the territory). Specifically, the centroid $\langle 0, 3, 4, s\rangle$ coincides with the centroid $\langle 1, 2, 2, q\rangle$ and the centroid $\langle 1, 2, 2, s\rangle$ coincides with the centroid $\langle 2, 1, 1, s\rangle$.

In the scope of our application, it is common that the proximity services are required by the user, inside a maximum radius r with respect to their position. This means that if a service can be provided only with a distance greater than r, it can be excluded.

Therefore, the user needs to identify just the cells for each level until a maximum level l (possibly l can coincide with the depth of the SQT). Consider a user X requiring a service inside a maximum radius corresponding to the level l. We denote by C_X^l the set of centroids detected by X at each level less or equal to l. For example, still referring to Figure 11, if the black dot represents the user X, then C_X^2 is exactly the set of centroids listed above, i.e., $\langle 0, 4, 3, q\rangle$, $\langle 0, 3, 4, s\rangle$, $\langle 1, 2, 2, q\rangle$, $\langle 1, 2, 2, s\rangle$, $\langle 2, 1, 1, q\rangle$, and $\langle 2, 1, 1, s\rangle$. This information is the basis of the proximity testing implementation, because, in principle, the provider can detect two users in proximity within a certain distance d (among the granularity set induced by the SQT), if they detect the same centroid of a level corresponding to squares of side not greater than $2d$. We will see in Section 6.2, how to exploit this basic principle to obtain a privacy-preserving result.

So far, we described how our grid is hierarchically organized. Indeed, our approach is grid-based. However, it is also tag-based, as highlighted in Section 2. We define next how the tag-based mechanism is implemented.

In principle, as a *tag*, one of the technologies identified in the literature [7], such as Bluetooth IDs, Wifi IDs, military codes in GPS, audio signals, LTE, and atmospheric gases could be used to obtain an unpredictable value, associated with a point in space and time. The purpose of this value is to allow the users to obscure the centroids when sending them to the provider, by preventing that the provider can reverse the information and then discover the actual positions. The details of this mechanism will be described in the next section. This value is called *salt*.

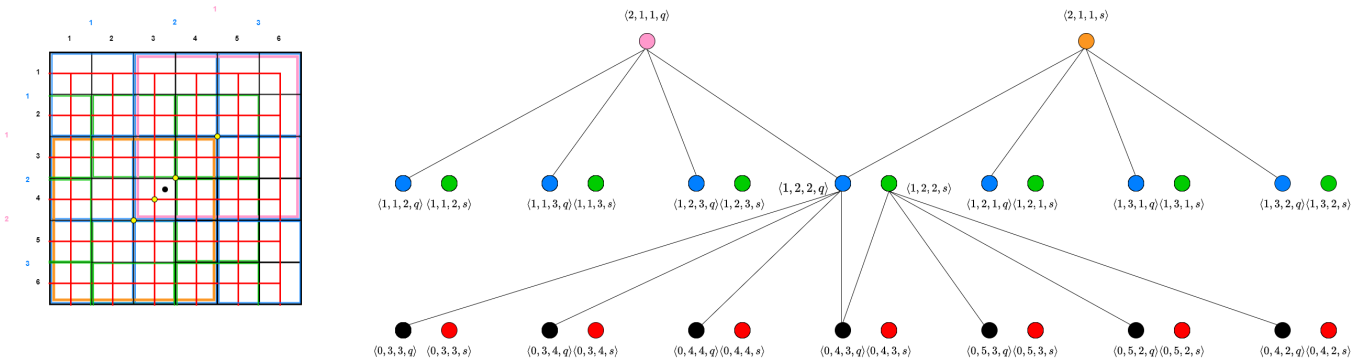

Figure 11: An example of SQT.

Among all the possibilities, a concrete way to implement salts is to rely on the collaboration of a telephone service provider (TSP), which transmits the salts through the cellular network. We can use the cellular cells to identify a region of the space in which, for a given time interval, a random salt, with a suitable rounding protocol, is periodically broadcasted to all the devices belonging to this cell. Despite the fact that the maximum space granularity we can obtain is the size of the TSP cells, we can anyway organize the tag-mechanism in a hierarchical fashion (in which the level 0 is represented by the TSP cells), needed to properly combine it with the hierarchical grid. Indeed, to implement a virtual TSP-cell of level k over the level $k-1$ (for a given $k \geq 1$), it suffices to virtually aggregate a suitable number of virtual TSP cells of level $k-1$ (observe that virtual TSP cells of level 0 are actually physical TSP cells). To be general, virtual/physical TSP cells are called *tag-cells*. An important aspect is to understand how the aggregation of tag-cells is done to obtain tag-cells higher in the hierarchy. Before discussing it, we deal with the problem of possible misalignment, at a given level of the hierarchy, between the grid and the tag-cell structure. Therefore, it can happen that a cell of the grid is cut by a tag-cell. In this case, two users belonging to the same cell might receive two different salts, and then they are not detected in proximity.

To avoid this, we enable an overlapping mechanism, which, at level 0 just relies on the physical overlapping between TSP cells adopted to manage the handover, and, at higher levels, it is suitably implemented.

First, we start from the grids of level 0. At level 0, the only requirement is that our cells are of size smaller than the overlapping area of TSP cells. This is realistic because this overlapping can be of some meters. This ensures that two users sharing the same cell (of level 0) receive at least one salt in common. For example, by referring to Figure 12, the green circles represent the tag-cells of level 0. All the users in the grey cell receive the same salt, while the users in the yellow cell may receive different salts (according to their positions), of which at least one is in common.

This overlapping mechanism has to be achieved at the higher levels. To do this, some tag-cells of level 0 are aggregated into a tag-cell of level 1. Inside this tag-cell, a new salt (of level 1) should be transmitted. Specifically, the TSP broadcasts in a tag-cell of level 0 both a salt of level 0 and a salt of level 1. The same salt of level 1 is broadcasted in the adjacent tag-cells of level 0 to form the virtual tag-cell of level 1.

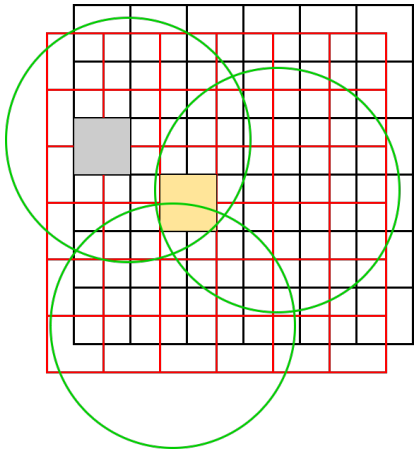

Figure 12: TSP overlapping mechanism.

For example, again in Figure 12, if the three green tag-cells (of level 0) are aggregated into a tag-cell of level 1, the three antennas transmit the same salt of level 1 (and a different salt of level 0 per tag-cell of level 0).

By iteratively applying this approach, the tag-cells of level i are aggregated into tag-cells of level $i+1$.

We assume that each salt is sent along with a label indicating its level so that the users can distinguish them. Given a user X, we denote by R_X^l the set of salts of level less or equal to l detected by X at a given instant.

We observe that, at a given instant, given two users X and Y and a level l, if X and Y are at a distance less than half of the length of the side of the cells of level l, then $C_X^l \cap C_Y^l \neq \emptyset$ and $R_X^l \cap R_Y^l \neq \emptyset$ i.e., the users share at least one salt and one centroid of level less or equal to l.

This observation is the basis of how proximity is detected. We will treat this aspect in the next sections, in which the three services `KN-service`, `UN-service`, `TN-service` are described.

6.2. `KN-service`: *Proximity between known users*

In this section, we describe the first proximity-based service we provide, called `KN-service`. We consider two users knowing each other who want to discover reciprocally if they are in proximity. The test is *symmetric*. This means that if a user X discovers the proximity of another user Y, then Y discovers the proximity of X.

The service is implemented as follows.

The first phase is the *handshake procedure*.

Consider a pair of users X and Y. Suppose the handshake is started by X.

First, X knows the real identity RI_Y of Y. Through the IBE scheme, X encrypts, under the identity RI_Y, the message $M_Q = Q||RI_X$, where Q is a random value and RI_X is the real identity of X. We denote by C_Q such an encrypted message. Then, X retrieves the SN identity of Y, denoted by SI_Y, and the ring which Y belongs to. They are obtained as follows: $SI_Y = h_R(RI_Y)$ and $v_k = ring(SI_Y)$, where the functions h_R and *ring* are those defined in Section 4.

Then, X invokes Primitive **P3** by passing C_Q and v_k as input. Y, who is the only user able to decrypt C_Q, retrieves $Q||RI_X$. Similarly to X, Y encrypts through IBE, under the identity RI_X, the message $\bar{M}_Q = Q||RI_Y$. We denote by $\bar{C}_Q$ such an encrypted message. Y retrieves $SI_X = h_R(RI_X)$, $v_j = ring(SI_X)$ and replies to X through Primitive **P3** by passing $\bar{C}_Q$ and v_j as input. Hence, X obtains $Q||RI_Y$ as confirmation of their request.

At this point, X (Y, resp.) generates a random value E_X (E_Y, resp.), called *Ephemeral ID*, an on-the-fly key K_X (K_Y, resp.) for the response, builds a message $M_X = E_X||Q||K_X$ ($M_Y = E_Y||Q||K_Y$, resp.), and encrypts it, through IBE, under the real identity RI_{SN} of SN. We denote by C_X (C_Y, resp.) this encrypted message. Now, X (Y, resp.) sends anonymously the message C_X (C_Y, resp.) to SN. This is done by X (Y, resp.) by invoking **P1** with input RI_{SN} and C_X (C_Y, resp.).

SN, through Q, links E_X and E_Y, computes $H_{XY} = h(E_X \oplus E_Y)$, and invokes **P2** by passing H_{XY} encrypted with K_X (K_Y, resp.) as input to respond to both X and Y.

This ends the handshake procedure. As a result of this procedure, the two users declare reciprocally their intention to test proximity. Moreover, SN is aware of this fact but knows only the one-time pseudonymous (i.e., the ephemeral IDs) of these users and links them thanks to the value Q.

At this point, the *position notification* procedure starts. Suppose that X wants to detect the proximity of Y if they are at a distance corresponding to the level l_X and that Y wants to detect the proximity of X if they are at a distance corresponding to the level l_Y.

First X (Y, resp.) retrieves the set of centroids $C_X^{l_X}$ ($C_Y^{l_Y}$, resp.) and the set of salts $R_X^{l_X}$ ($R_Y^{l_Y}$, resp.), as defined at the end of Section 6.1.

For each level $0 \leq i \leq l_X$ ($0 \leq i \leq l_Y$, resp.), for each centroid $C_X \in C_X^{l_X}$ ($C_Y \in C_Y^{l_Y}$, resp.) of level i, and for each salt $R_X \in R_X^{l_X}$ ($R_Y \in R_Y^{l_Y}$, resp.) of level i, X (Y, resp.) computes the digest $h_X = h(C_X||R_X)$ ($h_Y = h(C_Y||R_Y)$, resp.). We denote by H_X (H_Y, resp.) the list of the obtained digests.

At this point, X (Y, resp.), encrypts with IBE, under RI_{SN}, $H_X||E_X||\bar{K}_X$ ($H_Y||E_Y||\bar{K}_Y$, resp.), where $\bar{K}_X$ and $\bar{K}_Y$ are on-the-fly keys to encrypt the response. We denote by $\bar{C}_X$ ($\bar{C}_Y$, resp.) this encrypted message. Finally, X (Y, resp.) invokes **P1** by passing $\bar{C}_X$ ($\bar{C}_Y$, resp.) and RI_{SN} as input.

This ends the position notification.

The next phase of the protocol, called *proximity detection*, is performed SN side. This is done by simply computing the intersection $H_X \cap H_Y$ and by checking whether this intersection is not empty. If this is the case, then SN detects that X and Y are in proximity. Observe that the intersections performed in this phase are done only between the pairs of users whose ephemeral IDs are linked through the same random Q.

Furthermore, the proximity is detected only if X and Y are at a distance less than half of the length of the side of the square of the minimum level between l_X and l_Y. W.l.o.g, suppose that $l_X < l_Y$. This means that, even though Y selects a greater distance than X, the proximity will be detected only until a distance corresponding to the level l_X. This happens because the set H_X contains only the obscured centroids detected by X until the level l_X and does not provide any information about the higher levels.

Suppose now that $H_X \cap H_Y \neq \emptyset$ (i.e., proximity between X and Y has been detected).

In this case, SN should report this result to X and Y. This is done through a phase of the protocol called *proximity notification*. In this phase, SN invokes **P2** by passing the encryption with key $\bar{K}_X$ ($\bar{K}_Y$, resp.) of $H_{XY}||H_X \cap H_Y$ as input to reply to both Primitives **P1** invoked by X and Y.

The last phase is the *ephemeral confirmation* procedure. Once $H_{XY}||H_X \cap H_Y$ is obtained, X (Y, resp.), for each $hr \in H_X \cap H_Y$, computes $\bar{hr} = h(hr)$. We denote by H_R the list of the obtained digests. To be sure that SN does not forge a fake contact, X (Y, resp.) invokes Primitive **P3** by passing $E_X||H_R$ ($E_Y||H_R$, resp.), encrypted with IBE under RI_Y (RI_X, resp.), and v_k (v_j, resp.) as input. Through E_Y and E_X, X and Y respectively, are able to compute H_{XY} and to check the correspondence with the value obtained from SN. Moreover, H_R allows them to identify the centroids of the cells they share. Specifically, X (Y, resp.) checks that the list H_R computed locally coincides with the list provided by Y (X, resp.).

The flow of the proximity procedure in the case of a positive result is represented in the sequence diagram of Figure 13.

6.3. `UN-service`: *Proximity between unknown users*

In this section, we consider another symmetric case, in which the users involved in the proximity procedure do not know each other (thus, no preliminary handshake can be performed).

In current commercial systems, an example of this service is provided by Tinder, in which unknown users can come into contact if they are in proximity and match some preferences. Specifically, each user advertises a set of attributes (possibly, not identifying) and expresses a preference for the attributes of other users. To detect proximity, it is necessary that the users reciprocally show an expression of interest.

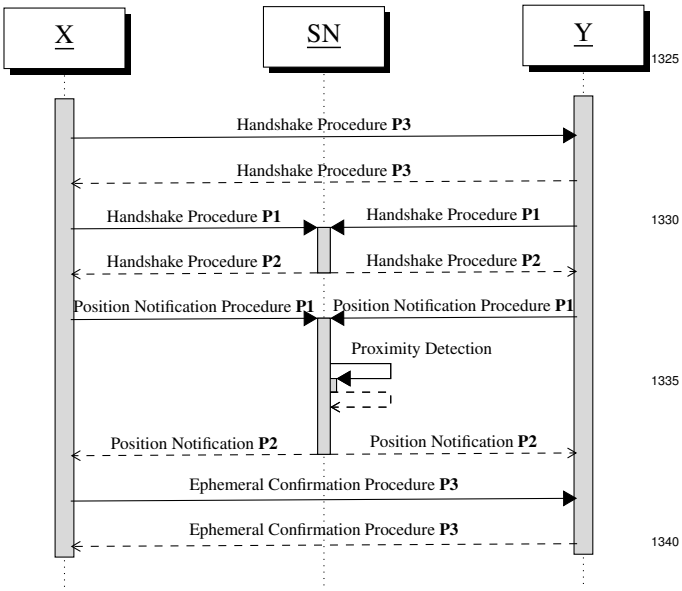

Figure 13: Sequence diagram of the `KN-Service`.

Now, we describe our privacy-preserving service, called `UN-service`, in which the whole proximity-testing activity remains unobservable by the social network provider.

The first step of the `UN-service` is the *advertising procedure*.

As in the position notification of the previous section, a given user X generates a list of digests H_X by using the salts $R_X^{l_X}$ and the centroids $C_X^{l_X}$ until the level l_X they want to set.

At this point, X creates an *advertisement* $I_X = \langle A_X, T_X, PK_X \rangle$. A_X is the set of attributes that X wants to advertise. T_X is a pair $\langle E_X, ring(SI_X) \rangle$, where E_X is an ephemeral (random) ID generated by X and $ring(SI_X)$ is the ring which X belongs to. Finally, PK_X is a public key generated by X.

For each salt $R_X \in R_X^{l_X}$, I_X is then encrypted with a symmetric key derived by R_X. We denote by C_X^R the set of all the obtained encrypted messages. This way, since the salts are distributed in a restricted area, the advertisement I_X can be decrypted only by the users in that area (who detect at least one common salt and, then, they can derive a valid key).

At this point, X is ready to send all the needed information to SN, that is, the digests (representing the obscured position), and the encrypted messages containing the advertisement. Thus, X can send them together with an on-the-fly key K_X, which SN will use for the response. The information sent by X to SN is then: $M_X = C_X^R | H_X \| K_X$, which X encrypts with IBE under the real identity of SN RI_{SN} obtaining C_X. This information is sent to SN, as before, by invoking Primitive **P1** with input C_X and RI_{SN}.

This concludes the advertising procedure.

Now, suppose that another advertising procedure is performed by a user Y in proximity who sends $M_Y = C_Y^R \| H_Y \| K_Y$ to SN.

As in the previous service, the *proximity detection* is performed SN side, and, since X and Y are in proximity, it results that $H_X \cap H_Y \neq \emptyset$. Therefore, SN has to notify the proximity (and the advertisement) to X and Y. This is done through the *proximity notification* phase. Specifically, SN encrypts C_Y^R (C_X^R, resp.) with K_X (K_Y,resp.) and passes the result of this encryption as input to **P2**, to reply to X (Y, resp.).

At this point, X (Y, resp.) is able to decrypt at least one advertisement contained in C_Y^R (C_X^R, resp.) and retrieves A_Y, T_Y, and PK_Y (A_X, T_X, and PK_X, resp.).

If both X is interested in the attributes A_Y and Y interested in the attributes A_X, then the *preference-expression* procedure starts, in which X and Y exchange with each other (through SN) their reciprocal interest.

Specifically, given $T_Y = \langle E_Y, ring(SI_Y) \rangle$, X computes $H_{XY} = h(E_X \oplus E_Y)$, encrypts $H_{XY} \| H_X$ with IBE under RI_{SN} and invokes Primitive **P1** by passing the result of this encryption and RI_{SN} as input. A dual procedure is executed by Y.

At this point, SN performs the *preference-expression detection*. Since SN receives from Y the digest $H_{YX} = h(E_Y \oplus E_X) = h(E_X \oplus E_Y) = H_{XY}$, it detects the reciprocal expression of preference between X and Y. The result of this detection is notified through the *preference-expression notification*. Therefore, SN replies to both the users by sending $H_{XY} \| H_X \cap H_Y$ through **P2** (it is encrypted, as usually, with an on-the-fly key sent by X and Y).

At the end of this phase, X and Y are informed about their reciprocal interest.

For security reasons (see Section 8), a final procedure, called *preference-confirmation*, is performed. In this procedure, the users agree on a common secret, by using Diffie-Hellman with perfect forward secrecy [53], in which the public keys included into the advertisement work as authentication of the destination. Moreover, the positions are also reciprocally checked. The common secret will be used to make the successive communication confidential. In detail, X generates a public DH parameter and encrypts it with PK_Y. We denote by CDH_X the result of this encryption. For each $hr \in H_X \cap H_Y$, X computes $\bar{hr} = h(hr)$ and obtains the list of digests H_R. Then, $CDH_X \| H_R$ and $ring(SI_Y)$ are passed as input to Primitive **P3**. Observe that both PK_Y and $ring(SI_Y)$ are contained in the advertisement of Y.

On the other hand, Y performs the dual operation. At the end of this procedure, if X and Y are the real owners of PK_X and PK_Y, respectively, they are able to obtain the public DH parameter of the other user and generate a common secret that can use to communicate. Moreover, through H_R, they confirm their proximity. Specifically, X (Y, resp.) checks that the list H_R computed locally coincides with the list provided by Y (X, resp.).

The sequence diagram of the `UN-Service` is reported in Figure 14.

6.4. `TN-service`: *Proximity between an anonymous user and a target*

The last service regards proximity testing performed by users with respect to a given target. Observe that the target could be indifferently static or moving. Moreover, we consider the most challenging case in which we do not allow to make public the

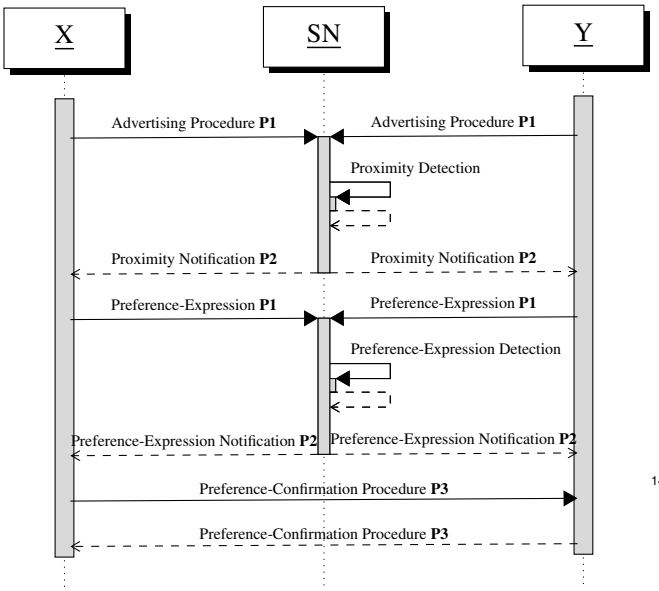

Figure 14: Sequence diagram of the `UN-Service`.

position of all the targets on a map. This prevents from massive retrieval of target positions that could be inadequate for privacy or business reasons. For example, in the context of car sharing, the targets are the available cars of a given company. However, to make public the positions of all the available cars on the whole map would give an improper advantage to competitors of this company by allowing easy benchmarking.

Other examples of moving targets are ride-sharing, crowd-shipping, or applications supporting people to obtain aids by volunteers when they are away (consider for example the case of blind people).

On the other hand, an example of static target is represented by points of interest (POIs). In this case, one could expect that the public availability of POIs in a map is not a problem. In general, this is true, but there could be some cases in which the publicity of such POIs should be limited (for example in the domain of intelligence, or sensitive targets), also for security reasons.

For the above reasons, the naive solution of providing users with a map including all the targets to allow a local privacy-preserving proximity testing cannot be adopted.

Unlike the previous two services, this service is *asymmetric*, in the sense that the user performing the test discovers the proximity of the target without disclosing their proximity to the target.

Now, we describe how this service is provided.

We consider a user X and a target T.

T performs, periodically, an *advertising procedure* simplified with respect to that described in the previous section.

Specifically, the advertisement of T, say I_T, contains only an advertising message AM_T with information purpose. As no response is enabled towards T, there is no need to include in the advertisement the information used for this purpose, i.e., key, ephemeral ID, and ring identifier.

To make the advertisement readable only for those users at distance compatible with their requirements, an encrypted version of I_T should be provided, for each involved level, by using the key derived by the corresponding salt, until a given maximum level l_T. Recall that, at a given level, due to tag-cell overlapping, multiple salts can be detected and, thus, multiple encrypted messages are needed.

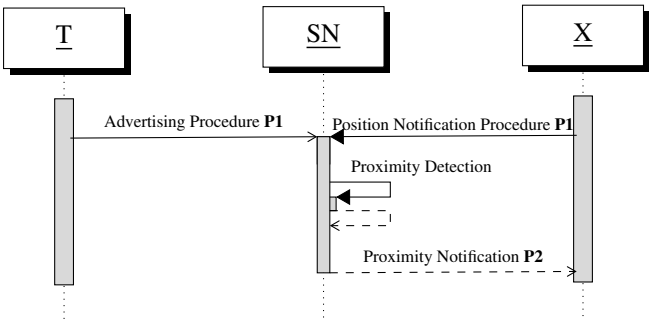

Figure 15: Sequence diagram of the `TN-Service`.

All the encrypted versions of I_T, along with the set of obscured positions H_T derived by T, are sent to SN (encrypted with IBE) through Primitive **P1**.

Due to the asymmetric nature of this service, X should not make any advertising procedure. X just generates a list of obscured positions H_X and sends them to SN through Primitive **P1** (*position notification* procedure).

As in the previous services, SN performs the *proximity detection* procedure, in which checks that $H_X \cap H_T \neq \emptyset$ and notifies the encryptions of the advertisement I_T to X through Primitive **P2** (*proximity notification* procedure).

X is able to decrypt I_T and to discover their proximity.

Observe that it is also possible to implement a client-side profile-based filtering to limit the information that the user has to process.

The flow of this proximity procedure is represented in the sequence diagram in Figure 15.

6.5. Well-formed domains

In this section, we discuss now how to build rings in such a way that the application domain is well-formed (as defined in Section 4.2). As this notion is strictly application-dependant, it is not adequate to address, in this paper, the problem of well-formed domain construction in the general case. Therefore, to be synthetic, we only consider the three services we presented earlier.

Concerning the `KN-Service`, in a real-life scenario, a relationship between the provided service and the friendship information occurring in the social network should be established to limit the scope of the preliminary handshake. For example, for a user, it could be reasonable to set that proximity testing should be allowed only with other users with maximum separation degree j in the social network (realistically, j could be 1 or 2). Otherwise, the service would degenerate into the `UN-Service`. Implicitly, we are considering a social network in which a relevant overlapping between real-life and virtual friendship exists (like in Facebook [54]).

Under the above assumption, a safe way to build rings is to define as application domain a subset of users forming a community with separation not greater than j, or, by approximation, on the basis of existing community-detection algorithms [55, 56, 57]. The above metric should be also combined with geographic information to avoid that this can be used by an adversary to infer significant bias of probability that a given pair of users are performing a proximity test (with respect to the random distribution).

Conversely, in the case of the `UN-Service`, no specific measures should be taken, besides those based on geographic information.

Finally, for the `TN-Service`, the application domain should be divided into two sub-domains: (1) the sub-domain of *clients*, and (2) the sub-domain of *targets*. Therefore, we have rings of clients and rings of targets. The measure to adopt is to ensure that both sub-domains are sufficiently homogeneous from the geographic point of view.

As mentioned earlier, the issue of building well-formed application domains is very complex and, to have a detailed and complete solution, the application of state-of-the-art social network analysis techniques should be tested in real life by measuring the reached privacy level (for example, through differential privacy). Therefore, we chose (by considering the aimed objectives of the article and its current length) not to treat this aspect here, but we argue (on the basis of the above considerations) that the problem of well-formed domain construction is feasible, at least with an acceptable degree of approximation.

7. Cost evaluation, prototype, experiments

7.1. Cost evaluation

Through this section, we evaluate the number of exchanged messages (primitives involved) and encryptions required by the three services presented in Section 6. In this analysis, we neglect other operations (such as hashing, xoring, and concatenation) since they are performed very few times and their cost is much less than even a single encryption. However, in the experimental analysis performed in Section 7.3, we take into account these operations too.

Since for all the three services, the number of exchanged messages and the number of encryptions depend on the result of the proximity test, we consider a single proximity test performed between two users (or between a user and a target) with a positive result. Indeed, this corresponds to the worst case in terms of number of operations.

We start with the `KN-service` described in Section 6.2 and analyze the exchanged messages. Each user invokes Primitive **P3** two times (one for the handshake and one for the ephemeral confirmation) and Primitive **P1** two times (one for the handshake and one for the position notification). Finally, SN invokes Primitive **P2** four times (two for the handshakes and two for the proximity notifications).

Regarding the `UN-Service` (defined in Section 6.3), each user invokes Primitive **P3** one time (for the preference confirmation) and Primitive **P1** two times (one for the advertising and one for the preference-expression). Server side, SN invokes Primitive **P2** four times (two for the proximity notifications and two for the preference-expression notifications).

Service	Number of Primitives
`KN-Service`	2 **P1** + 4 **P2** + 2 **P3**
`UN-Service`	2 **P1** + 4 **P2** + 1 **P3**
`TN-Service`	1 **P1** + 1 **P2**

Table 1: Primitives invoked by the three services of Section 6.

Service	Number of Encryptions
`KN-Service`	$4\,PK + 4S$
`UN-Service`	$3\,PK + (r+4)S$
`TN-Service`	$1\,PK + (r+1)S$

Table 2: Encryptions involved in the three services of Section 6.

Finally, regarding the `TN-Service` (defined in Section 6.4), the user and the target invoke Primitive **P1** one time (the user to perform the position notification and the target to perform the advertising procedure). On the other hand, SN invokes just Primitive **P2** one time for the proximity notification.

The result of this analysis is summarized in Table 1.

From this simple analysis, it is easy to conclude that the `KN-service` is the most onerous in terms of exchanged messages.

Now, we consider the number of encryptions. We distinguish the public-key encryptions (PK) from the symmetric encryptions (S) since the first are anyway (i.e., independently of the specific adopted cipher) more onerous than the latter.

For the sake of presentation, we omit the details about the steps in which these encryptions are performed and summarize the result of this analysis in Table 2.

Therein, we denote by r the average number of salts captured by the users when performing an advertising procedure. We refer to an average value by considering that a user may receive multiple salts due to both the multilevel grid organization and the grid-cell overlapping. We recall that, in `UN-Service` and `TN-Service`, some messages are encrypted symmetrically by using some keys derived from the salts received by the users and the targets.

Observe that the `KN-Service` requires one public-key encryption more than the `UN-Service`, but the latter requires r more symmetric encryptions. However, as public-key encryption requires much more computational effort than symmetric encryption and being r small, we conclude that the `KN-Service` is the most onerous, also regarding the number of encryptions.

Therefore, in Section 7.3, we focus our attention just on this service, which represents the worst case from a performance point of view.

7.2. Prototype

To both show the applicability of the proposed protocol and to validate it by experiments, we provide a prototype

of a *plug-in* implementing the `KN-Service`. In principle, this plug-in could be integrated into any existing social network. This prototype is available on `https://github.com/vincenzodeangelisrc/KN-Service_UNIRC`.

It is implemented in JAVA and uses the WebSocket technology [58] that offers full-duplex communication channels between a server and a client. It is used for real-time applications such as chats and real-time games.

The data are exchanged in JSON format.

The plug-in includes two modules: (1) a server-side module to integrate back-end functions of our protocol into the social network (2) a client-side module to integrate client-side functions into the social-network app of the user.

The server-side module implements the ring mechanism presented in Section 4. Specifically, the client app forwards the token to its next of the ring through the server. This module also includes the data structures necessary to detect the proximity between two users and to notify them.

The client-side module allows the client to receive a token, to fill it (if empty), and to forward it to the next client. The module also implements the role of the proxy node that empties the tokens and forwards them into the ring. To test the performance, this module includes several parameters to simulate the activity of more users, the cells in which they are located, the number of proximity tests to perform, and so on.

7.3. Experiments

By using the prototype described in the previous section, we performed an experimental evaluation of the proposed solution. We consider, as a metric of our analysis, the total time required to obtain the result of a proximity test in the `KN-service`.

Unlike experiments presented in [28, 41], we do not rely on the API of any specific social network and implement from scratch the prototype. This better simulates a real-life implementation of the proposal, which would require, as done in our prototype, the presence of a back-end module also to support quick communication between client and server. Moreover, this way, our prototype could be integrated into every existing social network, provided that our server-side module is imported from it.

In this analysis, we included hash computations, public-key encryptions, symmetric-key encryptions, xoring, and all the operations required by the service. Furthermore, to obtain realistic results, the server and the clients are remotely connected through the Internet (with a ping time of about 50 ms). Observe that, in the available social networks, lower ping times can be obtained, such as for Facebook (30 ms). Then, the actual performance of our solution, in a real-life implementation, might be also better than those experimented through this simulation.

The clients are simulated through threads running on PCs equipped with i7-8550U CPU (1.80GHz) and 16 GB of RAM. The server-side module is run on a PC equipped with i7-6500U CPU (2.50GHz) and 12 GB of RAM.

As already mentioned, the worst case corresponds to a successful proximity test. Therefore, we measured the time elapsed between the instant in which a user starts the test and the instant in which the same user receives the ephemeral confirmation from the other user. To be precise, we consider, in the analysis, also the time a user has to wait before receiving the first empty token to start the proximity test.

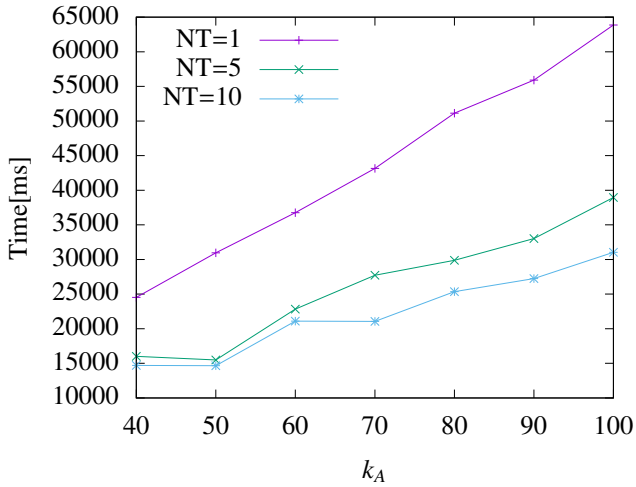

Figure 16: Total proximity-testing time vs privacy level with $\sigma = 0$.

The main parameters affecting the performance of our solution are three.

The first is the privacy level k_A. Indeed, the more users are in the ring, the higher the time required for the messages to exit from the ring or to reach the destination.

The second parameter is the number of tokens NT circulating in the ring. Indeed, the more the tokens, the higher the probability that a user receives an empty token (and then that they can transmit).

Finally, the last parameter measures the activity of the user in the ring. Indeed, if users send messages very often, then the percentage of available tokens for any user is reduced, and the total time they have to wait to receive the result is reduced too.

Regarding the latter parameter, to have a controllable environment, we define an *activity level* $\sigma \in [0, 1)$. It represents the probability that, at each turn of the ring, a given token is empty or filled. In particular, the proxy, with probability σ, sets each token filled, so that it cannot be used by other users.

At this point, we evaluate the performance of our protocol as the above three parameters vary.

The results are represented in Figures 16, 17, 18. Therein, we show as the total time required to perform a proximity test varies as the privacy level k_A varies. We consider $k_A \in [40, 100]$.

Each figure includes three plots, each associated with a different number of tokens ($NT = 1, 5, 10$) circulating in the rings.

Finally, the three figures differ in terms of activity level. Specifically, we consider three activity levels $0, 0.25, 0.5$ for the Figures 16, 17, 18, respectively.

As a first consideration, observe that, for all the possible configurations of k_A, σ, NT, the time of a proximity test results acceptable (the worst result corresponds to $k = 100, NT = 1, \sigma = 0.5$ that leads to a total time of about 90 seconds).

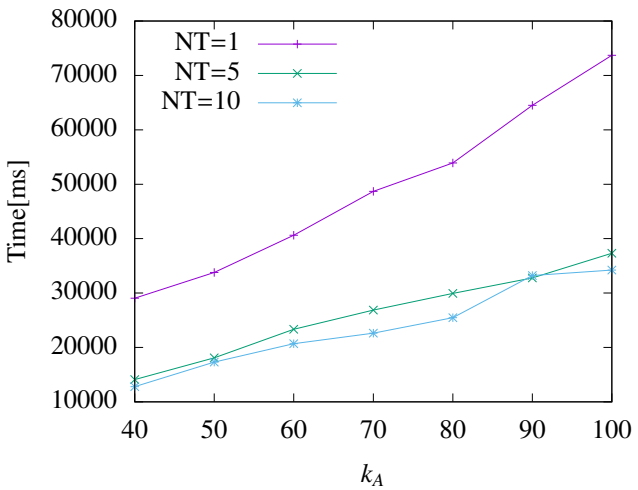

Figure 17: Total proximity-testing time vs privacy level with $\sigma = 0.25$.

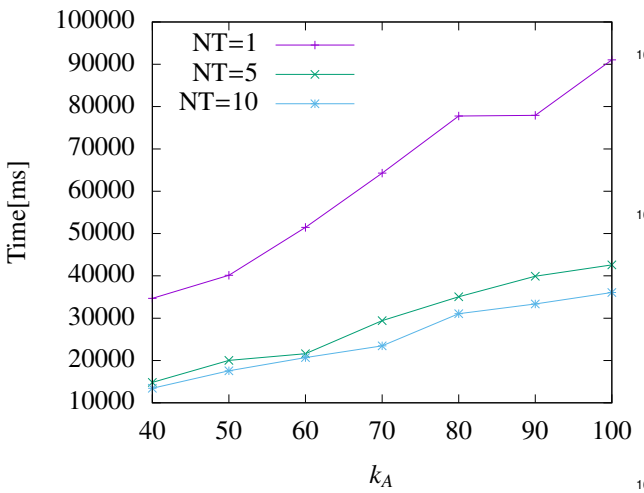

Figure 18: Total proximity-testing time vs privacy level with $\sigma = 0.50$.

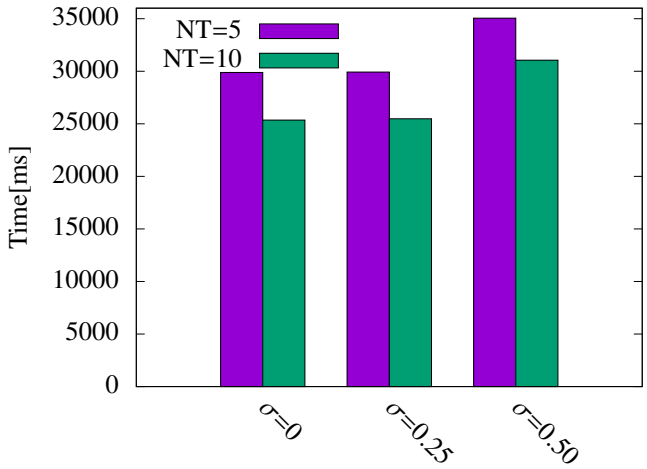

Figure 19: Total proximity-testing time with $k_A = 80$.

As expected, the total time increases with k_A (we observe an almost linear growth). Furthermore, the total time decreases with NT even though (regardless of the traffic conditions) no significant difference exists between the case with $NT = 5$ and the case with $NT = 10$. Observe that an increase of NT results in bandwidth waste and energy consumption for the devices (since they have to process more tokens). Therefore, $NT = 5$ can represent a good choice that leads to a total time of about $30 - 40$ seconds with $k = 100$ and any σ.

Finally, the total time increases as the activity level σ increases. Anyway, this effect is more evident when $NT = 1$ and less relevant for $NT = 5$ and $NT = 10$. This confirms that $NT = 5$ allows us to obtain good performance independently of the activity of the users (in the considered range).

To better highlight the performance of our solution in a realistic scenario, we report in Figure 19 the total time of a proximity test with a high privacy level ($k_A = 80$) for $NT = 5, 10$ and $\sigma = 0, 0.25, 0.5$.

We note that the total time is in the range $25 - 35$ seconds and is almost stable when σ varies.

8. Security analysis

In this section, we provide the security analysis of our solution. We start by defining the threat model **TM**. We introduce the following assumptions:

- **A1:** The application domains are well-formed.
- **A2:** The applied α-ring schema is τ-safe (with respect to a given user mapping h).

A1 can be obtained as discussed in Section 6.5. **A2** is guaranteed by ring schema updating 4.5. We recall that **A2** implies that for each α-ring, it is sufficiently guaranteed that at least k_A users are alive and collaborative.

Adversary Model. We consider as an adversary the social network provider SN. It is honest but curious in the sense that it legally performs the steps of the protocol, but attempts to break the anonymity of the user victim.

Observe that SN acts as a global passive adversary able to monitor the entire flow of messages exchanged between users. The proposed protocol offers anonymity in this severe adversary model.

Our analysis is performed at two levels. First, we show that the anonymity communication protocol described in Section 4 allows anonymous communication between users or between a user and SN. Then, we show that proximity testing (see Section 6) built on top of the anonymous communication protocol is secure.

We start by analyzing the first point. The next three Theorems show how the three Primitives of Section 4 guarantee anonymity.

Theorem 8.1. *A user U invoking **P1** can be identified with probability not greater than $\frac{1}{k_A}$ (sender anonymity).*

Proof 8.1. Since the token has a fixed size and changes hop-by-hop due to encryption, U cannot be identified as the sender when filling the token. Therefore, the only way for the adversary to identify that U is the sender is to detect a possible transition empty/filled or filled/empty of a token in another point of the ring and try to draw some information starting from this observation.

The only point of the ring from which the adversary can draw such information is the bridge user, say X. Indeed, this is the only point of the ring in which the possible transition empty/filled or filled/empty of a token could be in principle related to the observable incoming or outcoming traffic in/from the bridge. Transitions occurring in other points are not identifiable.

Therefore, we have to consider the following two cases (they are the only cases potentially helpful for the adversary). Either (1) the adversary observes incoming traffic in X (i.e., X may play the role of entry user), or (2) the adversary observes outcoming traffic from X (i.e., X plays the role of exit user). In case (1), two alternatives are possible. Either (1).a X actually injects a token into the ring by inserting the message coming from outside, or (1).b X empties a token circulating in the ring as the final step of Primitive **P2** or **P3**, and then X cannot process the incoming traffic. In case (1).a, the token cannot be filled by U, therefore we are not in the case of the hypothesis. Consider now case (1).b. The adversary knows that the token injected by X is empty. Two cases may hold. Either (1).b.1, i.e., the token, after a turn, reaches X still empty, or (1).b.2, i.e., the token, after a turn, reaches X filled. The adversary may understand which of the above cases ((1).b.1 or (1).b.2) occurs, just by observing if outcoming traffic arises from the arrival of the token (i.e., X, after the turn, plays the role of exit user). In the positive case, we are in case (1).b.2, otherwise we are in case (1).b.1. In the latter case, no sender is involved and we are not in the case of the hypothesis. In case (1).b.2, a sender (possibly, U) filled the token somewhere during the turn. But, due to Assumptions **A1** and **A2**, the adversary cannot identify the sender with probability greater than $\frac{1}{k_A}$.

In case (2) (i.e., X plays the role of exit user), the adversary can infer that the token injected by X into the ring is empty. Two cases may hold. Either (2).a, i.e., the token, after a turn, reaches X still empty, or (2).b, i.e., the token, after a turn, reaches X filled. The adversary may understand which of the above cases ((2).a or (2).b) occurs, just by observing if outcoming traffic arises from the arrival of the token (i.e., X, after the turn, plays again the role of exit user). In the positive case, we are in case (2).b, otherwise we are in case (2).a. In the latter case, no sender is involved in this turn and we are not in the case of the hypothesis. In case (2).b, a sender (possibly, U) had filled the token somewhere during the turn. But, due to the assumptions, the adversary cannot identify the sender with probability greater than $\frac{1}{k_A}$. The proof is then concluded.

Theorem 8.2. *A user U receiving a message through Primitive **P2** (as response to Primitive **P1**) can be identified with probability not greater than $\frac{1}{k_A}$ (recipient anonymity).*

Proof 8.2. This proof follows a reasoning similar to the proof of Theorem 8.1.

Since the token has a fixed size and changes hop-by-hop due to the encryption, U cannot be identified as the recipient when emptying the token. Therefore, the only way for the adversary to identify U as the recipient is to detect a possible transition empty/filled or filled/empty of a token in another point of the ring and try to draw some information from this transition. The only point of the ring from which the adversary can draw such information is the bridge user, say it X.

Therefore, we have to consider the following two cases (they are the only cases potentially helpful for the adversary). Either (1) the adversary observes incoming traffic in X (i.e., X could play the role of entry user), or (2) the adversary observes outcoming traffic from X (i.e., X plays the role of exit user). In case (1), two alternatives are possible. Either (1).a X actually injects a token in the ring inserting the message coming from outside or (1).b X empties a token circulating in the ring as the final step of Primitive **P2** or **P3**, and then X cannot process the incoming traffic. In case (1).a, the adversary can infer that a recipient (possibly, U) exists for this token. The only way to draw more information about this recipient is to follow the token and observe it when it reaches the bridge. At this point, the adversary can detect a transition filled/empty in X, but this does not give any additional information about where the message has been received. Therefore, due to Assumptions **A1** and **A2**, the adversary cannot identify the recipient with probability greater than $\frac{1}{k_A}$. Consider now case (1).b. The adversary knows that the token injected by X is empty. In this case, no recipient is present for such a token and we are not in the case of the hypothesis.

In case (2) (i.e., X plays the role of exit user), the adversary can infer that the token injected by X into the ring is empty. Again, no recipient is present for such a token and we are not in the case of the hypothesis. The proof is then concluded.

Theorem 8.3. *Let U be a user sending a message to a user D through **P3**. It holds: (1) U can be identified as the sender with probability not greater than $\frac{1}{k_A}$ and (2) D can be identified as the recipient with probability not greater than $\frac{1}{k_A}$.*

Proof 8.3. (1) can be proved as in Theorem 8.1. (2) can be proved as in Theorem 8.2.

This concludes the security analysis of the anonymity protocol described in Section 4. At this point, we show that the three services described in Section 6, leveraging the three Primitives **P1**, **P2**, and **P3**, protects the privacy of the users as specified in the next three theorems.

Theorem 8.4. *Consider two users X and Y (who know each other) leveraging the* `KN-service` *described in Section 6.2. It holds: (1) SN can guess the fact that X and Y are performing the test with probability not greater than $\frac{1}{k_A}$, (2) they detect their proximity only if they are really in proximity, (3) X discovers the proximity of Y only if Y discovers the proximity of X.*

Proof 8.4. We start with (1). Consider the handshake procedure. Due to the IBE scheme, to encrypt the messages M_Q and $\bar{M}_Q$, X and Y do not contact any PKI to obtain the public-key of the other user. These messages are exchanged through **P3**, therefore by Theorem 8.3, the anonymity of X and Y can be broken with probability not greater than $\frac{1}{k_A}$. Then, they invoke **P1** to send M_X and M_Y to SN that replies by sending H_{XY} through **P2**. Since M_X and M_Y do not contain any information linkable to the identities of X and Y, due to Theorems 8.1 and 8.2, again, SN can identify them with probability not greater than $\frac{1}{k_A}$.

Consider now the position notification procedure. X and Y send a list of digests obtained by applying a cryptographic hash function on a centroid and a salt that depend on the position of the user. SN cannot recover the salt that is physically distributed in the zone in which the users are located, thus the digest cannot be reversed and then the centroids cannot be identified. Therefore, the list of digests does not carry any information about the identity of the users. Since this list is sent through **P1** and the corresponding response through **P2**, again, by Theorems 8.1 and 8.2, the anonymity of X and Y can be broken only with probability not greater than $\frac{1}{k_A}$.

Finally, regarding the ephemeral confirmation procedure, E_X and E_Y are exchanged through **P3**. Thus, by Theorem 8.3, the anonymity of X and Y can be broken only with probability not greater than $\frac{1}{k_A}$.

The proof of (1) is concluded.

Regarding (2), proceed by contradiction. The statement of (2) does not hold if two cases occur: (2.1) either X or Y tries to invent a fake positive proximity result, or (2.2) SN tries to invent a fake positive proximity result between X and Y. We do not consider any other user since the other users involved in the communication see only encrypted data and cannot tamper them.

Considering (2.1), the proximity between X and Y is detected only if the lists of digests H_X and H_Y have at least one digest in common. To obtain the same digest, X and Y have to share the same centroid and the same salt. Even though the centroid can be obtained from any point of the world, the salt is physically distributed in a specific zone associated with the centroid, then the users have to be located in the same area. Regarding (2.2), the way in which SN can invent a contact is to provide X and Y with a tampered $H_{XY}||H_R$ when they are not in proximity. Anyway, the ephemeral confirmation procedure prevents such an attack since both H_{XY} and H_R can be checked by X and Y.

Finally, consider (3) and suppose that Y wants to discover the proximity of X without X can do it. The only way is to use a different ephemeral identifier, say $E_{Y'}$, during the position notification procedure such that X cannot recognize $h(E_X \oplus E_{Y'})$. Anyway, since X does not recognize $h(E_X \oplus E_{Y'})$, X do not perform the ephemeral confirmation procedure, so that Y does not know E_X, and then Y cannot detect to be in proximity of X.

Theorem 8.5. *Consider two users X and Y (who do not know each other) leveraging the* `UN-service` *described in Section 6.3. It holds: (1) SN can guess the fact that X and Y are performing the test with probability not greater than $\frac{1}{k_A}$, (2) they detect their proximity only if they are really in proximity, (3) X discovers that Y expresses a preference for X only if Y discovers that X expresses a preference for Y, (4) no spoofing of advertisement is allowed.*

Proof 8.5. We start by proving (1). During the advertising procedure, X and Y send M_X and M_Y to SN through Primitive **P1**. Since M_X and M_Y do not contain any information linkable to the identities of X and Y, due to Theorem 8.1, SN can identify them with probability not greater than $\frac{1}{k_A}$.

If X and Y are in proximity, they receive some information by SN through Primitive **P2**. Therefore, by Theorem 8.2, SN can identify them with probability not greater than $\frac{1}{k_A}$.

Consider now the preference-expression procedure. X and Y send H_{XY} (i.e., a non-identifier), to SN through Primitive **P1**. SN replies through Primitive **P2**. Again, by Theorems 8.1 and 8.2, SN can identify them with probability not greater than $\frac{1}{k_A}$.

Finally, regarding the preference-confirmation procedure, X and Y communicate between them through Primitive **P3** and then, by Theorem 8.3, SN can identify them with probability not greater than $\frac{1}{k_A}$. This concludes the proof of (1).

Regarding (2) (proceeding by contradiction), it does not hold if two cases occur: (2.1) either X or Y tries to invent a fake proximity between them, or (2.2) SN tries to invent a fake proximity between X and Y.

With the same reasoning as the proof of Theorem 8.4, (2.1) does not occur since X and Y have to obtain the salt. Regarding (2.2), it does not occur since the preference-confirmation procedure ensures that X and Y have at least a digest in common.

Concerning (3), during the preference-expression procedure, SN replies to X and Y only if $H_{XY} = H_{YX}$. Thus, they learn reciprocally their preference.

Finally, concerning (4), suppose Y tries to spoof an advertisement of another user Z located in the same area and uses this advertisement to enter into contact with X. Since Y does not know the public key PK_Z of Z, X and Z are unable to generate the DH secret during the preference-confirmation procedure. The proof is then concluded.

Theorem 8.6. *Consider a user X performing a proximity test with a target T leveraging the* `TN-service` *described in Section 6.4. It holds: (1) SN can guess the fact that X is performing the test with probability not greater than* $\frac{1}{k_A}$*, (2) X detects the proximity with T only if they are really in proximity, (3) T does not detect the proximity with X.*

Proof 8.6. Regarding (1), X sends only a list of obscured positions H_X through **P1** and receives an advertisement through **P2**. Similarly, T sends non-identifiers (through **P1**) to SN. Therefore, by Theorems 8.1 and 8.2, SN can identify X with probability not greater than $\frac{1}{k_A}$.

Concerning (2), to decrypt the advertisement of T, X has to share the same salt. Therefore, they have to be in proximity.

Finally, regarding (3), X, during the entire proximity test, does not provide any identifying information. Therefore, there is no way for T (even collaborating with SN) to detect that X is performing the test.

This concludes the security analysis.

9. Conclusion

Social networks are nowadays an integral part of people's lives. One of the main concerns related to their pervasiveness is that they expose users to serious privacy threats. Among others, a big problem is the concentration of a huge amount of personal information of billions of users in the hands of the social network provider. This may seriously threaten users' privacy if we consider the concrete risk that the social network provider behaves as an honest-but-curious adversary or that a data breach can occur. Starting from this awareness, this paper proposes a solution enabling the implementation of services requiring high protection of users' privacy, also against the social network platform playing as a global passive adversary. The solution is obtained by combining encryption mechanisms with an overlay routing protocol built on top of social network users, which, thanks to the presence of cover traffic, provides sender and recipient anonymity. Then, we applied the above solution to the case of the provision of proximity-based social networking, by designing different privacy-preserving proximity-based services. After implementing a prototype, we conducted a careful experimental campaign, to validate the feasibility of the provided services. The results are satisfactory, because, at a considerable privacy level (i.e., anonymity degree), the most complex proximity-based service, i.e., the privacy-preserving extension of the Facebook *Nearby Friends* service, is performed with times of order of magnitude of minute, thus suitable for the type of the service. The security analysis shows that the aimed privacy goals are fully reached, regarding both anonymity of communication and privacy of proximity-based services.

Two relevant points to discuss are why users should collaborate to implement the proposed collaborative approach, and, as it happens for other collaborative protocols, whether this exposes users to some vulnerabilities.

Concerning the first issue, a general statement could be that this is a question applicable to any collaborative protocol (such as P2P systems, for example). The general answer to this enigma is that the intrinsic incentive of any collaborative approach is that the users contribute to implement a service of which they can be clients. Therefore, the value that a single user transfers to the community is returned when the user exploits the service implemented by collaboration. In the domain of privacy in social networks, we argue that the above argumentation is rather convincing. Indeed, the need for privacy is growing. Moreover, recent facts have increased the perception of people that social network providers can misuse sensitive data. Combined with the fact that proximity and location data can be very sensitive, we expect that a real-life application of approaches like that presented in this paper is possible. On the other hand, in centralized social networks, only the collaboration of users can be used to obtain actual anonymity.

Concerning the second point, that is whether our collaborative protocol exposes users to some vulnerabilities, we can say that this is not the case. Unlike P2P protocols, in which users are forced to open some communication ports on their device, our P2P communication is only simulated. Indeed, the messages exchanged between users to implement rings are actually sent via communication tools offered by the platform. Therefore, the message exchange is managed as usual. No greater network exposure of the user's device is required. Therefore, the attack surface is not affected by our approach.

In conclusion, we argue that in a scenario in which P2P social networks [59, 60] are little realistic, our approach is a plausible example of how to achieve privacy goals by keeping the current centralized social network architectures.

References

[1] F. Shirazi, M. Simeonovski, M. R. Asghar, M. Backes, C. Diaz, A survey on routing in anonymous communication protocols, ACM Computing Surveys (CSUR) 51 (3) (2018) 1–39.

[2] M. R. Nosouhi, S. Yu, K. Sood, M. Grobler, Hsdc–net: Secure anonymous messaging in online social networks, in: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), IEEE, 2019, pp. 350–357.

[3] M. Kacimi, S. Ortolani, B. Crispo, Anonymous opinion exchange over untrusted social networks, in: Proceedings of the second ACM EuroSys workshop on social network systems, 2009, pp. 26–32.

[4] S. Mascetti, C. Bettini, D. Freni, X. S. Wang, S. Jajodia, Privacy-aware proximity based services, in: 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, IEEE, 2009, pp. 31–40.

[5] A. M. Ramtohul, K. K. Khedo, Proximity based social networking in urban environments: Applications, architectures and frameworks, in: Precision Positioning with Commercial Smartphones in Urban Environments, Springer, 2021, pp. 71–107.

[6] H. P. Li, H. Hu, J. Xu, Nearby friend alert: Location anonymity in mobile geosocial networks, IEEE Pervasive Computing 12 (4) (2012) 62–70.

[7] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, D. Boneh, et al., Location privacy via private proximity testing., in: NDSS, Vol. 11, 2011.

[8] I. Oleynikov, E. Pagnin, A. Sabelfeld, Where are you bob? privacy-preserving proximity testing with a napping party, in: European Symposium on Research in Computer Security, Springer, 2020, pp. 677–697.

[9] K. Järvinen, Á. Kiss, T. Schneider, O. Tkachenko, Z. Yang, Faster privacy-preserving location proximity schemes, in: International Conference on Cryptology and Network Security, Springer, 2018, pp. 3–22.

[10] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, B. Y. Zhao, Whispers in the dark: analysis of an anonymous social network, in: Proceedings of the 2014 conference on internet measurement conference, 2014, pp. 137–150.

[11] G. Danezis, C. Diaz, A survey of anonymous communication channels, Tech. rep., Technical Report MSR-TR-2008-35, Microsoft Research (2008).

[12] M. J. Freedman, R. Morris, Tarzan: A peer-to-peer anonymizing network layer, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002, pp. 193–206.

[13] P. Kotzanikolaou, G. Chatzisofroniou, M. Burmester, Broadcast anonymous routing (bar): scalable real-time anonymous communication, International Journal of Information Security 16 (3) (2017) 313–326.

[14] K. Bennett, C. Grothoff, Gap–practical anonymous networking, in: International Workshop on Privacy Enhancing Technologies, Springer, 2003, pp. 141–160.

[15] A. Hirt, M. Jacobson, C. Williamson, Taxis: scalable strong anonymous communication, in: 2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, IEEE, 2008, pp. 1–10.

[16] A. Beimel, S. Dolev, Buses for anonymous message delivery., Journal of Cryptology 16 (1) (2003).

[17] A. L. Young, M. Yung, The drunk motorcyclist protocol for anonymous communication, in: 2014 IEEE Conf. on Communications and Network Security, IEEE, 2014, pp. 157–165.

[18] M. G. Reed, P. F. Syverson, D. M. Goldschlag, Anonymous connections and onion routing, IEEE Journal on Selected areas in Communications 16 (4) (1998) 482–494.

[19] M. K. Reiter, A. D. Rubin, Crowds: Anonymity for web transactions, ACM transactions on information and system security (TISSEC) 1 (1) (1998) 66–92.

[20] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, D. S. Wallach, Ap3: Cooperative, decentralized anonymous communication, in: Proceedings of the 11th workshop on ACM SIGOPS European workshop, 2004, pp. 30–es.

[21] Q. Wang, N. Borisov, Octopus: A secure and anonymous dht lookup, in: 2012 IEEE 32nd Int. Conf. on Distributed Computing Systems, 2012, pp. 325–334.

[22] W. Wang, M. Motani, V. Srinivasan, Dependent link padding algorithms for low latency anonymity systems, in: Proceedings of the 15th ACM conference on Computer and communications security, 2008, pp. 323–332.

[23] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, P. Francis, Towards efficient traffic-analysis resistant anonymity networks, ACM SIGCOMM Computer Communication Review 43 (4) (2013) 303–314.

[24] D. L. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, Communications of the ACM 24 (2) (1981) 84–90.

[25] M. Burnside, A. D. Keromytis, Low latency anonymity with mix rings, in: Int. Conference on Information Security, Springer, 2006, pp. 32–45.

[26] L. Von Ahn, A. Bortz, N. J. Hopper, K-anonymous message transmission, in: Proceedings of the 10th ACM conference on Computer and Communications Security, 2003, pp. 122–130.

[27] P. Mittal, M. Wright, N. Borisov, Pisces: Anonymous communication using social networks, arXiv preprint arXiv:1208.6326 (2012).

[28] F. Buccafurri, V. De Angelis, M. F. Idone, C. Labrini, Anonymous short communications over social networks, in: International Conference on Security and Privacy in Communication Systems, Springer, 2021, pp. 43–63.

[29] E. Union, Regulation EU No 910/2014 of the European Parliament and of the Council, `http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32014R0910&from=EN` (23 July 2014).

[30] H. Jiang, J. Li, P. Zhao, F. Zeng, Z. Xiao, A. Iyengar, Location privacy-preserving mechanisms in location-based services: A comprehensive survey, ACM Computing Surveys (CSUR) 54 (1) (2021) 1–36.

[31] L. Šikšnys, J. R. Thomsen, S. Šaltenis, M. L. Yiu, O. Andersen, A location privacy aware friend locator, in: International Symposium on Spatial and Temporal Databases, Springer, 2009, pp. 405–410.

[32] L. Šikšnys, J. R. Thomsen, S. Šaltenis, M. L. Yiu, Private and flexible proximity detection in mobile social networks, in: 2010 Eleventh International Conference on Mobile Data Management, IEEE, 2010, pp. 75–84.

[33] Y. Zheng, M. Li, W. Lou, Y. T. Hou, Location based handshake and private proximity test with location tags, IEEE Transactions on Dependable and Secure Computing 14 (4) (2015) 406–419.

[34] M. D. Firoozjaei, J. Yu, H. Choi, H. Kim, Privacy-preserving nearest neighbor queries using geographical features of cellular networks, Computer Communications 98 (2017) 11–19.

[35] P. C. Ng, J. She, R. Ran, A compressive sensing approach to detect the proximity between smartphones and ble beacons, IEEE Internet of Things Journal 6 (4) (2019) 7162–7174.

[36] C. Huang, R. Lu, H. Zhu, J. Shao, A. Alamer, X. Lin, Eppd: efficient and privacy-preserving proximity testing with differential privacy techniques, in: 2016 IEEE International Conference on Communications (ICC), IEEE, 2016, pp. 1–6.

[37] J. Son, D. Kim, M. Z. A. Bhuiyan, R. Tashakkori, J. Seo, D. H. Lee, Privacy enhanced location sharing for mobile online social networks, IEEE Transactions on Sustainable Computing 5 (2) (2018) 279–290.

[38] A. Ye, Q. Chen, L. Xu, W. Wu, The flexible and privacy-preserving proximity detection in mobile social network, Future Generation Computer Systems 79 (2018) 271–283.

[39] P. Kotzanikolaou, C. Patsakis, E. Magkos, M. Korakakis, Lightweight private proximity testing for geospatial social networks, Computer Communications 73 (2016) 263–270.

[40] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, S. Jajodia, Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies, The VLDB journal 20 (4) (2011) 541–566.

[41] F. Buccafurri, V. De Angelis, M. Francesca Idone, C. Labrini, A privacy-preserving protocol for proximity-based services in social networks, in: 2021 IEEE Global Communications Conference (GLOBECOM), 2021, pp. 1–6. `doi:10.1109/GLOBECOM46510.2021.9685284`.

[42] R. Zhang, Y. Zhang, J. Sun, G. Yan, Fine-grained private matching for proximity-based mobile social networking, in: 2012 Proceedings IEEE INFOCOM, IEEE, 2012, pp. 1969–1977.

[43] B. Niu, T. Zhang, X. Zhu, H. Li, Z. Lu, Priority-aware private matching schemes for proximity-based mobile social networks, arXiv preprint arXiv:1401.8064 (2014).

[44] X. Yi, E. Bertino, F.-Y. Rao, K.-Y. Lam, S. Nepal, A. Bouguettaya, Privacy-preserving user profile matching in social networks, IEEE Transactions on Knowledge and Data Engineering 32 (8) (2020) 1572–1585. `doi:10.1109/TKDE.2019.2912748`.

[45] D. Recordon, D. Reed, Openid 2.0: a platform for user-centric identity management, in: Proceedings of the second ACM workshop on Digital identity management, 2006, pp. 11–16.

[46] A. De Caro, V. Iovino, G. Persiano, Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts, in: M. Joye, A. Miyaji, A. Otsuka (Eds.), Pairing-Based Cryptography - Pairing 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 347–366.

[47] Z. Brakerski, A. Lombardi, G. Segev, V. Vaikuntanathan, Anonymous ibe, leakage resilience and circular security from new assumptions, in: J. B. Nielsen, V. Rijmen (Eds.), Advances in Cryptology – EUROCRYPT 2018, Springer International Publishing, Cham, 2018, pp. 535–564.

[48] A. Pfitzmann, M. Hansen, A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management (2010).

[49] E. Union, European Blockchain Services Infrastructure (EBSI), `https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/EBSI/`, [Online; accessed 2022] (2022).

[50] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, l-diversity: Privacy beyond k-anonymity, ACM Transactions on Knowledge Discovery from Data (TKDD) 1 (1) (2007) 3–es.

[51] N. Li, T. Li, S. Venkatasubramanian, t-closeness: Privacy beyond k-anonymity and l-diversity, in: 2007 IEEE 23rd International Conference on Data Engineering, IEEE, 2007, pp. 106–115.

[52] F. Buccafurri, F. Furfaro, G. Mazzeo, D. Saccà, A quad-tree based multiresolution approach for two-dimensional summary data, Information Systems 36 (7) (2011) 1082–1103, special Issue: Advanced Information Systems Engineering (CAiSE'10). `doi:https://doi.org/10.1016/j.is.2011.03.007`.
URL `https://www.sciencedirect.com/science/article/pii/S030643791100041X`

[53] I. S. .-. W. Group, et al., Ieee standard specifications for public key cryptography, The Institute of Electrical and Electronics Engineers, Inc (2000) 1–228`doi:10.1109/IEEESTD.2000.92292`.

[54] T. Bucher, Facebook, Vol. 104, John Wiley & Sons, 2021.

[55] Q. Ji, D. Li, Z. Jin, Divisive algorithm based on node clustering coefficient for community detection, IEEE Access 8 (2020) 142337–142347.

[56] C. He, X. Fei, Q. Cheng, H. Li, Z. Hu, Y. Tang, A survey of community detection in complex networks using nonnegative matrix factorization, IEEE Transactions on Computational Social Systems (2021).

[57] D. Jin, Z. Yu, P. Jiao, S. Pan, D. He, J. Wu, P. Yu, W. Zhang, A survey of community detection approaches: From statistical modeling to deep learning, IEEE Transactions on Knowledge & Data Engineering (01) (5555) 1–1. doi:10.1109/TKDE.2021.3104155.

[58] I. Fette, A. Melnikov, The websocket protocol, RFC 6455, RFC Editor, http://www.rfc-editor.org/rfc/rfc6455.txt (December 2011).
URL http://www.rfc-editor.org/rfc/rfc6455.txt

[59] N. Masinde, K. Graffi, Peer-to-peer-based social networks: A comprehensive survey, SN Computer Science 1 (5) (2020) 1–51.

[60] P. Lin, P.-C. Chung, Y. Fang, P2p-isn: A peer-to-peer architecture for heterogeneous social networks, IEEE Network 28 (1) (2014) 56–64.