DOCTORAL SCHOOL
MEDITERRANEA UNIVERSITY OF REGGIO CALABRIA

DEPARTMENT OF INFORMATION ENGINEERING, INFRASTRUCTURES
AND SUSTAINABLE ENERGY
(DIIES)

PHD IN
INFORMATION ENGINEERING

S.S.D. ING-INF/05
XXXV CYCLE

ANONYMOUS PROTOCOLS FOR COMMUNICATION AND SERVICE DELIVERY

CANDIDATE
VINCENZO DE ANGELIS

ADVISOR
Prof. FRANCESCO BUCCAFURRI

COORDINATOR
Prof. ANTONIO IERA

REGGIO CALABRIA, JANUARY 2023

Finito di stampare nel mese di **Gennaio 2023**

Edizione 

Quaderno N.

Collana *Quaderni del Dottorato di Ricerca in Ingegneria dell'Informazione*
Curatore *Prof. Antonio Iera*

ISBN

Università degli Studi *Mediterranea* di Reggio Calabria
Via dell'Università 25. Reggio Calabria

VINCENZO DE ANGELIS

ANONYMOUS PROTOCOLS FOR COMMUNICATION AND SERVICE DELIVERY

The Teaching Staff of the PhD course in

INFORMATION ENGINEERING

consists of:

Antonio IERA (coordinator)
Pier Luigi ANTONUCCI
Giuseppe ARANITI
Francesco BUCCAFURRI
Claudia CAMPOLO
Giuseppe COPPOLA
Marintonia COTRONEI
Lorenzo CROCCO
Dominique DALLET
Claudio DE CAPUA
Francesco DELLA CORTE
Giuliana FAGGIO
Gioia FAILLA
Fabio FILIANOTI
Patrizia FRONTERA
Sofia GIUFFRÈ
Giorgio GRADITI
Voicu GROZA
Tommaso ISERNIA
Gianluca LAX
Aimè LAY EKUAKILLE
Gaetano LICITRA
Antonella MOLINARO
Francesco Carlo MORABITO
Andrea Francesco MORABITO
Giacomo MORABITO
Rosario MORELLO
Fortunato PEZZIMENTI
Filippo Gianmaria PRATICÒ
Domenico ROSACI
Giuseppe RUGGERI
Mariateresa RUSSO
Antonino VITETTA

Contents

List of Figures

List of Listings

List of Tables

1

Introduction

This first chapter of this thesis is devoted to introducing the topic of anonymity and of how it will be treated throughout the rest of the thesis. We start by discussing the motivation leading to this work, by highlighting the benefits (and the drawbacks) of anonymity features in real-life situations. Due to the vastness of the topic, we focus our attention on two macro-areas of anonymity: anonymous communication and anonymous service delivery.

1.1 Motivation

Despite the relevant effort devoted in the literature [219, 275, 199, 168] in the last 30 years, a universally accepted definition of "Anonymity" does not exist. It depends on the context we refer to. For example, if we want to hide the sender of a given message, we need an anonymous communication network [245]. On the other hand, if we want to make records stored on a database unlinkable to the real identity of a user, we have to apply data anonymization techniques [195, 105].

To answer the increasing demand for anonymity services, a lot of tools, frameworks, and techniques have been designed. One of the most relevant and currently adopted is the Tor protocol [258] allowing a user to connect anonymously to a web server. Furthermore, through the hidden services [207] mechanism, the web servers can hide their IP addresses from a curious provider or the client itself. We dedicated two entire chapters (3 and 4) of this thesis to how to extend Tor to achieve stronger anonymity features.

We have to point out that this opens several ethical issues [114]. On the one hand, the Tor network and, in general, the provision of anonymous services offers an opportunity for the proliferation of the black market and cyberterrorism [5, 183]. On the other hand, anonymity services may have a positive impact in terms of censorship resistance. Furthermore, a lot of privacy-preserving applications require anonymity features such as electronic auctions [119, 173], anonymous surveys [128], or e-voting

[52, 118]. Another example is provided by anonymous proximity-based services (PBS) [181] which we mention in Chapter 6.

We strongly believe in this second vision of anonymity and propose new solutions enabling stronger anonymity features than traditional solutions. In addition, we think that investigating new proposals represents an effective way to understand how to fight the malicious use of anonymity services.

In this last direction, in this thesis, we investigate the trade-off between accountability and anonymity [92]. Indeed, accountability refers to the possibility of identifying and attributing responsibility to an entity for a given action. Specifically, we propose a solution in which the linkage between the real identity of a user leveraging an anonymous service and their pseudonymous remains hidden even though two of the three parties involved in the protocol collaborate. However, in the case of need, e.g., if required an agent authorized by the law, the above linkage can be disclosed if all three parties collaborate. This can be useful, for example, in an anonymous social network [276] to fight cyberbullying. The solution we propose relies on the blockchain technology [78] that is a reference technology in the provision of anonymous services.

Particular attention is devoted in this thesis to the protection of the users against a global adversary able to monitor the entire traffic exchanged in the network. This is an ambitious goal since it represents a very strict threat model (see Section 1.4). We want to highlight that this type of adversary is not an abstraction and it is present in real-life systems. The most emblematic example is represented by a social network providing anonymity services (see Chapter 6). In this case, if all the communications are delivered within the social network, the social network provider itself, if malicious, represents a concrete case of a global adversary. Another example of a global adversary is represented by the collaboration of some internet service providers that can track the messages from the originator to the destination. In this thesis, the solutions presented in Chapters 2, 3, 4, 5, 6, and 8 deal with this adversary.

Due to the vastness of the topic, in this thesis, we focus on two particular aspects of anonymity i.e., anonymous communication and anonymous service delivery which will be discussed in the next two sections.

1.2 Anonymous Communication

The aim of an anonymous communication network is to protect the identity of the users sending and/or receiving data over the Internet [245, 109, 88]. Since these data often contain personal information linkable to the user's identity, they have to be encrypted before sending, thus guaranteeing *data confidentiality*. However, data

confidentiality is a necessary but not sufficient condition to achieve anonymity. Indeed, even though an attacker cannot access the content of data, the knowledge of the IP address of the sender and/or the recipient reveals information about the geographic location or even the real identity of a user [210]. Then, offering anonymity to a user means hiding their IP address.

To do this, several anonymous communication protocols have been proposed in the literature. They differ in terms of security goals and the efficiency they achieve. Regarding security, we mean both security properties (i.e., sender anonymity, recipient anonymity, or relationship anonymity) and adversary capabilities that form a threat model (see Section 1.4 for a more complete description). On the other hand, for efficiency, three main metrics can be considered: latency, cover traffic, and anonymity (which is also a security feature). Latency represents the delay needed to transfer a message from the sender to the recipient. Cover Traffic is dummy traffic introduced to hide real traffic and protect the sender and/or the recipient from traffic analysis attack [17, 193, 190]. Since cover traffic requires a waste of bandwidth (and then energy consumption) to transfer dummy packets, it should be reduced as possible. Observe that, to offer protection against a global observer able to monitor the entire traffic exchanged in the network, the inclusion of cover traffic is necessary [71]. Otherwise, the adversary can simply observe the traffic originating from a node and identify it as the sender. Similarly, by following the flow of traffic until the destination, a global adversary can identify the recipient. As highlighted in Section 1.1, in this thesis, we reserve a lot of attention against this type of adversary, therefore almost all the solutions we propose include cover traffic. An interesting question is how the three above metrics (latency, cover traffic, and anonymity) are related between them. We investigate in detail this aspect in Chapter 5. In particular, our study is based on the well-known trilemma, called the *anonymity trilemma* [74], which states the existence of a trade-off between the three metrics.

In the first two parts of this thesis, we propose new protocols for anonymous communication that improve or outperform the state-of-the-art approaches by providing stronger anonymity features. In Part I, we propose new anonymous communication protocols built over the transport layer. In this class of approaches, in principle, it is possible to select any application layer leveraging the anonymous communication features implemented by the protocols. However, as with traditional solutions, our approaches perform well when some conditions are met at the application level. For example, the solutions presented in Chapters 3 and 4 represent extensions of the Tor protocol to achieve sender and recipient anonymity, respectively. In this case, as the standard Tor protocol, the solutions are applied for web-browsing ap-

plications. Instead, the solution discussed in Chapter 5 performs well with uplink-intensive applications.

In Part II, we propose two new protocols built over an existing application layer. Clearly, the disadvantage is that they work only for the specific application layer in which they are implemented. On the other hand, they do not require the set-up of an external ad-hoc anonymous communication network to work. This allows us to leverage the communication primitives offered by the specific application layer without requiring heavy infrastructural network changes. In this thesis, we present two protocols belonging to this class. The first is discussed in Chapter 6, in which the application layer is represented by a social network. Therein, we implement a protocol for anonymous short communication and show its application to the proximity services domain [181]. As observed in Section 1.1, the social network provider represents a real case of a global adversary. The second protocol we propose in Part II is based on MQTT as an application layer. We leverage the bridging mechanism offered by MQTT to deliver anonymous publish/subscribe messages. All the messages are exchanged through the MQTT protocol and no infrastructural change is required. In this solution, we do not offer protection against a global adversary. However, we allow anonymous publishing/subscribing of topics against colluding MQTT bridge brokers and against the public broker hosting the topics.

The rest of this section is devoted to providing a brief overview of the main anonymous communication approaches and protocols present in the literature. We refer the reader, for a more complete discussion, to the excellent survey presented in [245].

The first class of approaches to achieve anonymous communication is represented by *mixnet*. This concept was first introduced by Chaum [59]. The idea is to use *mix nodes* that collect messages coming from different sources, shuffle them, and send them to another mix node or to the destination. The messages are encrypted in several layers by using the public keys of the mix nodes. The innermost message is encrypted with the public key of the recipient. Each mix node removes a layer of encryption and discovers the next node to forward the message. The correlation between a message entering a mix node and a message exiting from the same node is made harder for an adversary since the mix node does not forward immediately the message to the next node. Instead, it waits for a batch of messages and after shuffling them sends the entire batch of messages each to the proper mix node (or destination). This delay introduces a price in terms of latency.

Starting from this original proposal, a lot of approaches [158, 273, 223, 269], also recent, were developed to improve the efficiency (especially in terms of latency) of the mixnets.

Compared to other approaches discussed in the following, mixnets offer a good (low) latency with a relevant price in terms of cover traffic. From the security point of view, they offer the best characteristics by resisting a global active adversary and traffic analysis attacks even though they are vulnerable to the collusion of mix-nodes [243].

The second class of approaches we consider is based on *Onion-Routing*. The most representative protocol of this class is certainly Tor [258]. The Tor network is open source and includes a large set of collaborating routers (about 7000 in September 2022) [226]. This makes Tor the most used solution in real life for anonymous communication (about 3000000 users in September 2022) [226].

In Tor, each client runs locally an *Onion proxy* (OP) which establishes a virtual circuit of three *Onion routers* (OR) to communicate anonymously with the destination. Each Onion router only knows the previous and the successive node of the path. During a set-up phase, some symmetric keys are exchanged between the OP and each OR. These keys are used to encrypt the messages in a layered fashion as explained for the mixnets. However, since once the path is established, it remains the same for the entire duration of the connection, then the next node of the path does not need to be included in the encryption. This makes this layered encryption more efficient. All the messages are exchanged in fixed cells of 512 bytes, to make harder traffic analysis attacks. A more detailed description of how the Tor protocol works is provided in Section 4.2 of Chapter 4.

Compared with other solutions, Tor offers the best features in terms of latency. Also cover traffic is limited. However, from the point of view of security, it does not offer good guarantees [261, 150, 239]. Indeed, it suffers from timing attacks [167, 107], traffic confirmation attacks [231],watermarking attacks [136], and self-promotion attacks [247]. Moreover, no protection against a global adversary is provided in Tor. This latter point motivates the two solutions provided in Chapters 3 and 4.

Other solutions belonging to the Onion Routing-based protocols are, basically, extensions or improvements of Tor [4, 248, 212].

Another class of anonymous communication protocols we investigate in this thesis is composed of P2P approaches. Several protocols we propose are P2P since they require the collaboration of other users that act as relay nodes and deliver a message to the destination. Besides new proposals designed from scratch (see Chapters 2, 3, 6), we provide, in Chapter 5, an improvement of the Tarzan protocol [98] and in Chapter 7, we apply the Crowds [230] to the MQTT domain.

Often, P2P approaches are DHT-based since they require a Distributed Hash Table (DHT) to discover other peers in the network.

The main advantage of P2P solutions is scalability since each user adhering to the network provides additional capability to the network itself. The question of why the nodes should collaborate remains an open problem. However, as matter of fact, concrete implementations of anonymous P2P solutions exist [127]. A discussion about this point is provided in Chapter 6.

Regarding latency, cover traffic, and security, we can say nothing a priori but they depend on the specific considered P2P protocol.

The fourth class is represented by *DC-Nets* [58]. They offer anonymous communication through multi-party computation. From the point of view of security, they offer information-theoretically secure anonymity. However, they suffer from scalability problems. Moreover, they may require both a huge overhead in terms of cover traffic and high latency. DCnet protocols are used for group communication/messaging.

Finally, the last approach we present is based on *buses* [124, 21, 296]. In this solution, a predetermined route is used by the sender to anonymously communicate with the destination. This approach introduces lower cover traffic. However, it does not scale since the messages flow through all the nodes of the network. This leads to a prohibitive cost in terms of latency.

1.3 Anonymous Service Delivery

In Part III of this thesis, the concept of anonymity is investigated from a different point of view with respect to the first two parts. We consider a scenario in which a service provider offers a given service to a user. In order to provide this service, data, potentially linkable to the real identity of the user, have to be sent to the service provider. In this case, the objective of our study is not to protect the communication between the user and the service provider. Instead, we aim to provide solutions in which the user discloses the minimum possible information about themselves so that a given degree of anonymity is achieved against the service provider itself. In other words, in our adversary model, the service provider represents the attacker. Clearly, if the service provider, or in general an attacker, has particular capabilities (e.g., background information about the users) also the communication has to be protected. This is the case of the solution proposed in Chapter 8, in which we include a mechanism to offer anonymity also against a global adversary. However, this mechanism is partially orthogonal to the way in which the anonymous service is delivered.

An interesting point is that, in the protocols we propose, multiple parties are involved to deliver a service. Then, it is relevant to investigate the level of trust we have to give to each party and the degree of anonymity the user can obtain.

A very important class of services in which privacy (and in particular anonymity) features are required is represented by location-based services (LBS) [255, 174]. They are characterized by the fact that the user has to provide their position to obtain the service. Clearly, the position is a very sensitive piece of information acting as a quasi-identifier [240] allowing an easy de-anonymization of the user. In this case, a common approach followed in the literature (and also in Chapter 8 of this thesis) is the construction of a cloaking area including k users that potentially may require the same service. This way, the provider is unable to distinguish the identity of the user requiring a service from the identities of other $k-1$ users. Unfortunately, the construction of the cloaking area relies on the presence of a trusted party, called LTS, that knows the position of the users. A benefit of the solution introduced in Chapter 8 is that it is hierarchical and the view of each LTS is reduced with respect to a centralized solution.

Until now, in this thesis, we presented anonymity as the main and only objective to reach. However, it is interesting to observe that if we relax the anonymity requirements, we can enable other useful features. This is exactly the goal we pursue in Chapters 10 and 9. Specifically, in Chapter 10, we considered a smart-city scenario in which a user interacts with multiple subsystems (service providers) and produces data. In this case, each subsystem knows the identity of the user and the data they produce. Then, the objective is not to provide anonymity for the user against the subsystem. However, each subsystem should know only the data it generates and should not be able to link its data with those generated by other subsystems. In addition, it may be useful that this linkage can be performed by other authorized parties to extract useful statistical information. Still, the linkage of these data should be performed in an anonymous form, i.e., the authorized parties link the data of the users but do not know their identity. The solution we propose enables all the above features.

Another benefit we obtain by relaxing the degree of anonymity is presented in Chapter 9. Therein, we have a service provider that knows the user only through a pseudonymous username. This is the typical case of an anonymous social network. A common problem with these types of services is that the users, protected by anonymity, may behave illegally, fueling phenomena such as cyberbullying. In this case, accountability properties, allowing the disclosure of the real identity of the user in case of need, are desirable. Some solutions [47, 44, 236] reach this goal by including a third party that by collaborating with the service provider can re-identify

the anonymous user. Our solution improves the trade-off between anonymity and accountability by requiring that the linkage of the pseudonymous username with the real identity can be performed if three (instead of two) parties collaborate.

1.4 Threat Models

Since this thesis deals with anonymity issues, all the proposed solutions include a security analysis aiming to show how the claimed security goals are achieved. An exception is represented by the solution described in Chapter 5, in which we do not include a security analysis since it inherits all the security benefits from [98]. However, the experimental validation investigates the size of the anonymity set as a metric to measure the degree of anonymity.

The security analyses are performed in terms of the capabilities of the adversary and security properties we want to achieve, that define the threat model for a specific solution.

Among the different solutions, we pursue different goals (security properties). In addition, some types of adversaries, which differ in terms of ability, are only applicable to some solutions but not to others. Then, this results in threat models that vary according to the considered solution. In particular, we observe that the security properties considered in the threat models presented in Parts I and II are the same. Indeed, the solutions therein proposed are about anonymous communication in which a standard terminology for the concept of anonymity is present in the literature [219].

On the other hand, the above notion of anonymity cannot be applied (directly) to the solutions presented in Part III. Then, therein, we consider different security properties.

In this section, we present an overview of the security properties considered in this thesis and of the possible adversaries against which these properties should be guaranteed. Clearly, not all the solutions guarantee all the security properties (but possibly a subset of them). Moreover, as already mentioned, not all types of adversaries can be considered for all the solutions. The specific threat model for each solution is then discussed within each chapter of this thesis.

We start by considering three security properties that apply to the domain of anonymous communication (Parts I and II).

An Anonymous Communication Network may offer [219]:

1. *sender anonymity* if the adversary cannot sufficiently identify the sender in a set of potential senders, called *sender anonymity set*.

2. *recipient anonymity* if the adversary cannot sufficiently identify the recipient in a set of potential recipients, called *recipient anonymity set*.
3. *relationship anonymity* if 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.

Observe that the definitions given in [219], with the use of the term sufficiently, means "both that there is a possibility to quantify anonymity and that for some applications, there might be a need to define a threshold where anonymity begins".

In the solutions presented in Parts I and II, we quantify anonymity in terms of the size of the anonymity set. Then, given an anonymity set of k possible senders/recipients, the adversary cannot identify the actual sender/recipient among these k. This corresponds to the notion of communication k-anonymity [274].

It is easy to realize that: sender anonymity implies relationship anonymity and recipient anonymity implies relationship anonymity.

For the solutions presented in Part III, we consider different security properties.

In Chapter 8, we refer to a slightly different notion of anonymity, i.e., location k-anonymity [102, 145, 113]. According to this notion, a request coming from a user in a given position (known to the provider) cannot be distinguished from the requests coming from other $k-1$ users whose positions are known to the provider. As explained in Chapter 8, to obtain location k-anonymity, the *reciprocity* [149, 106, 63] property has to be guaranteed.

In Chapter 9, we express the security properties in the "opposite" form by means of *compromises*, in the sense that we show as a given compromise does not occur in our solution. Therein, we consider two properties. The first regards the impossibility for some parties to link a username with the real identity of a user (*pseudonymity*). The second property is the *accountability*. Specifically, the possibility for other parties (different from the previous ones) to discover the above linkage username-real identity in case of need.

Finally, in Chapter 10, we refer to the property of *unlinkability* of data in the sense that it is not possible to link data coming from the same user except some authorized parties called *analysts*. However, the analysts link the data without knowing the real identity of the user.

This concludes the discussion about the security properties considered in this thesis.

Now, we introduce the possible adversaries with their capabilities.

Similar to the security properties, the adversaries considered in the field of anonymous communication have different capabilities with respect to the adversaries considered for anonymous service delivery.

Indeed, the solutions presented in Parts I and II include the deployment of an overlay network for anonymous communication. Therefore, the different considered adversaries reflect their capabilities to compromise the network. On the other hand, for the protocols in Part III, we do not focus on the anonymous communication of the actors but on the possibility to provide a service without disclosing users' identities.

Regarding anonymous communication, the first distinction we consider is between:

1. *A local adversary:* that is able to monitor the traffic originating from /incoming to some nodes of the network.
2. *A global adversary:* that is able to monitor all the messages exchanged in the network.

Another possible distinction is between:

1. *A passive adversary:* that monitors the exchanged messages but is not able to block them or forge new messages (even dummy).
2. *An active adversary:* that monitors the exchanged messages and is able to block them and/or forge new messages.

Clearly, the adversaries we consider in our threat models are a combination of these two classes. Mainly, the solutions we propose aim to offer protection against a global passive adversary (this is an ambitious goal). However, in some solutions such as that of Chapter 4, we also analyze a global passive adversary with the power of compromising some nodes by accessing the content of the received messages. Furthermore, in the solution proposed in Chapter 7, we are vulnerable to the global adversary but active attacks performed by malicious nodes are ineffective (under some conditions).

Another interesting point is to understand what happens when the adversary is the recipient of the communication. In this case, providing sender anonymity against the recipient of the communication is not a trivial task, especially when the recipient has to reply to the sender. However, almost all the solutions provided in this thesis achieve this feature.

The last consideration about the adversaries in anonymous communication protocols regards the case in which the protocol needs a third party, such as a *directory server* providing "yellow pages" service. In this case, this party can be considered:

- *Fully trusted*: it performs the steps of the protocol legally and does not attempt to break the security properties.
- *Honest but curious*: it performs the steps of the protocol legally but attempts to break the security properties.

- *Malicious*: it attempts to break the security properties possibly by deviating from the steps of the protocols.

Concerning anonymous service delivery, we have different parties (including a service provider) that communicate among themselves to provide an (anonymous) service. To define our threat model, we consider each involved party as an adversary belonging to one of the three above-mentioned categories: Fully trusted, Honest but curious, or Malicious. Furthermore, we also consider the possible collusion among different parties and investigate if this collusion compromises the security properties.

We recall that, in anonymous service delivery, the main aim is not to protect the communication among the parties, that is the objective of anonymous communication. However, in the solution presented in Chapter 8, we address also the problem of anonymous communication against a passive global adversary coinciding with the service provider in collusion with a telephone service provider.

1.5 Outline of the thesis

This thesis is organized into four parts.

Part I describes four solutions implemented over the transport layer supporting anonymous communication. In particular, in Chapter 2 we describe a preliminary approach to offer anonymity guarantees against a global passive adversary able to observe the entire traffic exchanged in the network. The results of this approach are published in a research paper [36]. Even though the approach is not complete and presents some limitations, it is at the basis of the solutions discussed in Chapters 3 and 6.

Chapter 3 presents a proposal to extend the Tor protocol to achieve sender anonymity against a global adversary. Basically, the solution exploits the approach presented in Chapter 2 to build a P2P anonymous network of senders before the entry point of the Tor circuit. The results of this approach are published in a research paper [43].

The dual problem is faced in Chapter 4. Therein, we present a solution to achieve recipient anonymity in Tor against the global adversary. In this case, we do not set a P2P network but leverage the hidden services mechanism present in Tor. Some preliminary results of this approach are published in a research paper [40].

Finally, Chapter 5 concludes the first part of this thesis by proposing an extension of the well-known Tarzan protocol [98] to obtain better performance in uplink-intensive applications. The idea of the approach is to remove the bidirectional links (needed also to obtain a response) present in Tarzan by rearranging the topology of

the mimics in order to form cycles that enable the response. This allows an important reduction of the cover traffic.

In Part II, two anonymous communication protocols are presented. They are implemented over an existing application layer. In Chapter 6, the application layer is represented by a social network. The solution we propose resists a global passive adversary (i.e., the social network provider) and offers both sender and recipient anonymity. It is suitable for short communication and can be adopted to implement privacy-preserving proximity-based services. The results of this approach are included in two research papers [37, 42].

Instead, the application layer considered in the solution of Chapter 7 is MQTT. We leverage the bridging mechanism natively offered by MQTT to deploy a network enabling anonymous publishing/subscribing to topics. The approach we follow is based on the Crowds protocol [230]. An important point to observe is that all the messages are exchanged through the standard MQTT primitives This allows us not to require changes in the standard MQTT infrastructure.

Part III includes three solutions for anonymous service delivery. The first solution is presented in Chapter 8, in which we propose a hierarchical LTS system offering protection against a global adversary. The services we consider in this solution are location-based services and the aim is to provide the users with guarantees about the fact that the provider is not able to identify their positions. Some preliminary results of this approach are published in a research paper [39].

In Chapter 9, we present a solution to the trade-off between accountability and anonymity. In particular, we consider a scenario in which a user is known to the service provider just by means of a pseudonymous username. To enable the possibility to re-identify the user in case of malicious or illegal behavior, we require the collaboration of three parties. This is the main advantage with respect to other solutions in which just the collaboration of two parties is enough. The results of this solution are published in a research paper [45].

The solution presented in Chapter 10 enables the anonymous linkage of open data only by some authorized parties. Specifically, we are in a smart city scenario in which a user interacts with several subsystems by producing data published in the form of open data. Even though these data are anonymized for privacy reasons, our solution enables their linkage only to authorized parties which, however, are unable to discover the real identity of the user to whose data refers. This concludes the third part of this thesis.

Finally, in Part IV, we draw the conclusions of the thesis.

Part I

Anonymous communication over the transport layer

The design of anonymous communication protocols resisting a global adversary able to observe the entire traffic exchanged in the network is not a trivial task. On one hand, as stated in [71], any solution pursuing this goal has to include cover traffic, i.e., dummy traffic among which the real traffic is hidden. On the other hand, cover traffic represents energy consumption and bandwidth waste, so it should be reduced as much as possible. For example, a trivial solution to achieving protection against the global adversary consists in sending periodically dummy messages to all the users of the network and replacing one of them with the actual message when needed. Obviously, this solution is not applicable.

Reducing the cover traffic can also have an impact on the latency, since, in general, reduces the opportunity for a user to send a message. This leads to the so-called anonymity trilemma [74], which states the existence of a trade-off between anonymity, cover traffic, and latency.

In this part of the Thesis, we propose solutions to this trade-off that are implemented over the transport layer. This makes them, in principle, independent of the application layer to which the messages are generated. However, even if they work with any application layer, they perform better when some applications are considered.

This part includes four proposals.

In Chapter 2, we describe a preliminary approach to offer anonymity guarantees against a global passive adversary. Even though the approach is not complete and presents some limitations, it introduces the concept of *ring* that is at the basis of the solutions discussed in Chapters 3 and 6.

Chapter 3 presents a proposal to extend the Tor protocol to achieve sender anonymity against the global adversary. Basically, the solution exploits the approach presented in Chapter 2 to build a P2P anonymous network of senders before the entry point of the Tor circuit.

The dual problem is faced in Chapter 4. Therein, we present a solution to achieve recipient anonymity in Tor against the global adversary. In this case, we do not set

a P2P network but leverage the hidden services mechanism present in Tor. A relevant contribution of this solution is represented by the formal security analysis we conducted.

Finally, Chapter 5 concludes the first part of this thesis by proposing an extension of the famous Tarzan protocol [98] to obtain better performance in uplink-intensive applications. The idea of the approach is to remove the bidirectional links (needed also to obtain a response) present in Tarzan by rearranging the topology of the mimics in order to form cycles that enable the response. This allows an important reduction of the cover traffic. In this Chapter, the anonymity trilemma is also investigated through experimental validation.

2

A preliminary approach leveraging onion routing to achieve anonymity against a global adversary

Through this chapter, we provide the first protocol of this thesis achieving anonymity against a global adversary. The proposed approach is preliminary and some aspects are missing or deserve a better investigation, in particular regarding implementation and experimentation. However, we present this solution, since it introduces, in a simple form, the idea of ring. This concept is deepened and exploited in the next chapters to build more robust and performing protocols also on top of existing application layers. Furthermore, in this chapter, also the concept of Onion routing, well-known in the literature, is presented. It is the basis of Tor, which is the most popular anonymous communication protocol used for low-latency network applications. However, Tor does not protect in the strict threat model of a global adversary. Therefore, in the next two chapters, we will see how to extend the standard Tor Protocol to achieve sender and recipient anonymity, respectively. In this chapter, the concept of Onion routing is used in its original definition that introduces a communication overhead as observed in Section 2.5. The results of the proposed approach are published in a research paper [36].

2.1 Introduction

Onion routing, originally proposed in [108], aims to reach anonymity by forwarding the message over multiple proxies (relay nodes) which, thanks to a public-key encryption wrapping, are only aware of the next hop of the route.

A lot of practical implementations (Tor [193] is the most famous) and extensions of the original idea have followed mainly through overlay protocols, but also implemented at the network layer [60, 61]. Despite its age, Onion is still a state-of-the-art approach, currently subject of attention in the research community [162]. However, Onion suffers from a serious drawback regarding anonymity. Indeed, in the global passive adversary model, in which all the traffic can be observed by the adversary, both sender and recipient anonymity are not achieved. Indeed, it suffices for the adversary to place itself at the first relay node of the Onion circuit to identify the

sender, and to place itself at the last relay node, to identify the recipient. This fact may have a relevant impact because a combined timing attack can allow the attacker to pair the sender and recipient, thus breaking also relationship anonymity [93].

To solve the above drawback, we propose a routing protocol that extends Onion by introducing the concept of *ring* to strengthen sender anonymity and an *inertia route* to strengthen recipient anonymity. Specifically, in the global passive adversary model, sender and recipient anonymity are achieved in an anonymity set of configurable size K. As a consequence, the weaknesses of Onion mentioned above are solved in our protocol and the global passive adversary cannot break relationship anonymity.

2.2 Overview of the Protocol

In this section, we present an overview of the proposed routing protocol.

Our protocol is cooperative, as all nodes play also routing functions, but we assume the presence of a (hierarchical and distributed) directory system DS providing the resolution of symbolic identities of nodes into network addresses. With SID we refer to a symbolic identity and with NID to a network address. Obviously, we assume that the nodes of this directory system are reliable in the sense that they collaborate with the protocol, but they can under the observation of the adversary.

Our protocol combines the classical protocol Onion with the concepts of *ring* and *inertia route* to fulfill the anonymity requirements.

A *ring* is a circular route of nodes of a given configurable size K. It represents the anonymity set for the sender. Therefore, any potential sender Alice is included in a number of rings. It is always possible for Alice to know, for each ring, the nodes (and the public keys) belonging to it. Each ring has an owner responsible for generating *containers* that injects in the ring. Such containers turn continuously in the ring and represent *cover traffic*, since they form *dummy* traffic that can or not contain data. Indeed, every message transmission is done by filling one of the empty containers, in such a way that, thanks to probabilistic encryption, empty and filled containers are indistinguishable from an external observer.

The *inertia route* is the mechanism at the basis of recipient anonymity in a proper anonymity set of nodes (of configurable size K too). In words, the inertia route is a random walk (selected by the sender) external to the ring and includes the recipient in a sufficiently undetectable position. As for the ring, the inertia route includes cover traffic. Indeed, once a message included in a container reaches the intended recipient, it continues empty (i.e., as a dummy message) in the inertia route until the

terminal node is reached. Moreover, a backward burst of cover traffic is activated in the inertia route to hide the sender of the response.

Suppose that Alice wants to send a message to Bob (which represents a SID known to Alice). Her first task is to send a request to DS in an anonymous way by using one of the rings she belongs to. The DS response provides Alice with the NID of Bob plus the information needed for Alice to build the route external to the ring playing the role of recipient anonymity set. Obviously, Bob is in this set. Observe that the ring-based mechanism allows us to have recipient anonymity for the DS response. At this point, Alice selects an exit node of the ring and establish a transmission path composed of the portion of the ring from Alice to the exit node concatenated with the external route including Bob. The message is sent by Alice through this path by using Onion-based public-key encryption wrapping [72, 59]

2.3 The onion-based routing protocol

Through this section, we describe, in detail, the protocol introduced in the previous section.

We assume the presence of a *Directory System* DS, for which the following holds. DS has a public key utilized by network users to obtain the information needed to arrange anonymous communication with other users. DS manages the Public Key Infrastructure (PKI) of the whole network so that it generates and distributes the public keys to all the nodes of the network. Public keys are utilized by the nodes to set Onion-wrapping encryption. DS, as already mentioned, provides the resolution of symbolic identifiers of nodes (SIDs) into network address identifiers (NIDs) needed to build the route to any destination. Specifically, DS stores a table called *Resolution Table (RT)*. Every entry of the RT consists of a tuple of the form: $\langle SID, NID, P \rangle$, where P is the public key of the node NID.

DS is also responsible for the set-up of rings and for the storage of information about them. As clearly stated in Section 2.4, DS is assumed trusted in the sense that it executes correctly the protocol, but the adversary may know the content of all the DS incoming and outcoming messages.

A *ring* is a circular sequence of NIDs of size K. Each ring has an owner whose NID identifies it. Each node belongs to at least l rings. Observe that the smaller l, the higher network latency and the less cover traffic is. DS stores also a *Membership Ring Table* (MRT), which, for each NID, associates the rings it belongs to. A tuple of MRT is of the form: $\langle NID, NID_1, \ldots, NID_{\bar{l}} \rangle$, where $NID_1, \ldots, NID_{\bar{l}}$ ($\bar{l} \geq l$) are the rings which NID belongs to. DS stores another table called *composition ring table* (CRT), which, for each ring, includes the K nodes that compose it and their associated public keys.

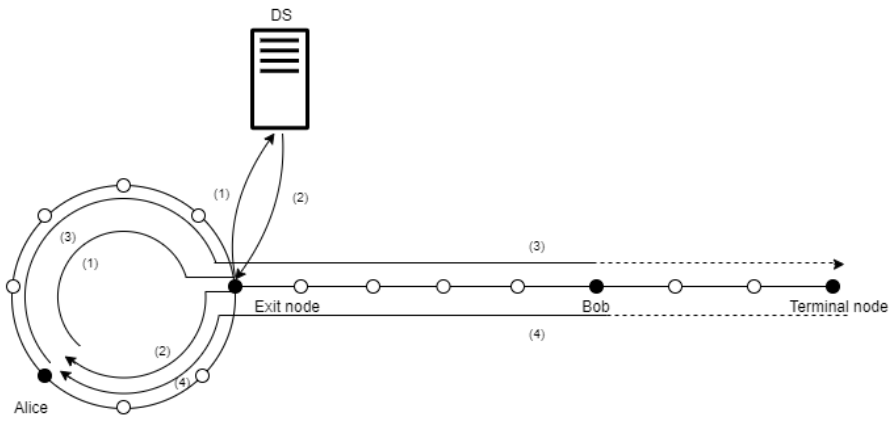


Fig. 2.1: Graphical representation of the onion-based protocol.

.

A tuple of CRT is of the form: $\langle NID, (NID_1, P_1), \ldots, (NID_K, P_K)\rangle$, where $NID{=}NID_1$, and the sequence (NID_2, P_2), $\ldots, (NID_K, P_K)$ represents the successive nodes in the ring (in order). All the information regarding rings is periodically cached by nodes, in such a way that any node stores locally the rings to which it belongs. The owner of a ring is responsible for generating the containers it injects into the ring. They are random messages of fixed length.

Now, we describe how anonymous communication works. Suppose that Alice wants to send a message to Bob. Recall that, the anonymization of the communication leverages the ring structure. In it, a number of containers turn continuously, each reporting the NID identifying the ring. In general, each message is sent by any user by filling an empty container without changing its size. Empty and filled containers are indistinguishable. This is obtained by probabilistically encrypting the container hop-by-hop with the public key of the next node in the ring. Thus, each node that receives a container (empty or filled), decrypts it, possibly fills it (provided that it is empty), encrypts with the public key of the next node in the ring, and then sends it to this node, and so on. We denote the encryption of a message M with a key X by $E_X(M)$. We use a probabilistic encryption function so that when the same message is encrypted several times, we obtain different ciphertexts.

First Alice performs a *DS request* to obtain the information needed to build the communication route. The DS request proceeds as follows. Alice, in order to send the request to DS, takes the DS public key, denoted by K_{DS}, needed to protect the sensitive request information, in this way: $E_{K_{DS}}(K_A, SIDs)$ where K_A is a key generated on the fly by Alice to allow the encryption of the *DS Response*, and $SIDs$ is a list of length K of SIDs in which $K-1$ are dummy SIDs and one is the SID of Bob.

Moreover, Alice, for each ring she belongs to, thanks to tables MRT and CRT, periodically chooses (randomly) an exit node, which we denote as at distance j from

her in the ring. Observe that, within the request, a random R is included, which is an identifier of the request, useful for the exit node in the DS response, as we will see later. Thus, it is important that R is not encrypted with the public key of DS (obviously, it is hop-by-hop probabilistically encrypted as all messages to prevent distinguishability). Alice, to send the request, has to wait for an empty container. She takes the first container that arrives by a whatever ring which she belongs to, and fills it with the message $(NID_j, R, E_{K_{DS}}(K_A, SIDs))$, where NID_j is the NID of the node j.

At this point, the DS request turns in the ring until it arrives at the exit node j, which recognizes to be the exit node by checking the NID specified in the request. Before j sends the request directly to DS, it stores the request identifier R and the NID of the ring from which the request came (the latter information is available in the container). This is done to allow that when DS will reply to the exit-node j at this request, thanks to R, the exit node will identify the request and will inject the response into the suitable ring. Subsequently, it generates a new empty container, re-injects it in the ring in order to keep constant the arriving frequency of containers in the ring.

After DS has received Alice's request, DS generates the *DS response* as follows. The DS processes the request by decrypting it and recovering the list of K SIDs requested by Alice, the on-the-fly key of the Alice K_A, and the random R. So, DS searches in the *Resolution Table* the list of the K SIDs and recovers the matching NIDs. Hence, DS builds the DS response as follows: $(R, E_{K_A}(NIDs - PKs))$ by including R and the encrypted list of the K NIDs (with the associated public keys), with K_A. DS sends the DS response to the exit node j. The latter thanks to R and the recorded information, identifies the ring in which to inject the response. To do this, j waits for the arrival of the first empty container of the appropriate ring and fills it with the DS response. When the filled container arrives at Alice, she takes R and decrypts it, to recover the K NIDs with which she builds the communication route.

Alice, similarly to the DS request phase, chooses $j \leq K$, by identifying the number of hops until the exit node. This way, Alice builds the first portion of the path including ring nodes (denoted as r_xs) and the remaining K nodes (denoted as w_ys) outside the ring. Bob is placed somewhere in the outside portion of the path. In sum, the overall path is: $\pi = \langle r_1, r_2, \dots r_j, w_{j+1}, w_2, \dots w_{j+K} \rangle$. Observe that r_1 is the NID of Alice, r_j is the exit node and there exists $1 \leq i \leq K$ such that w_{i+j} is the NID of Bob. The portion of the route going from Bob until the endpoint is called *inertia route*.

The communication towards Bob is Onion-based, so the public keys of the nodes of the path chosen by Alice are utilized to set Onion-wrapping encryption. Alice, according to the Onion protocol, builds the message to Bob, waits

for the first container empty, and fills it with the message. This way, the message encrypted by Alice is the following: $\bar{M} = E_j(\cdots E_{i+j}(\cdots E_{j+K-1}(E_{j+K}(D,0), w_{j+K})$ $\cdots, M, r_j, dest, w_{i+j+1})\cdots, exit, w_{j+1})$. Observe that $\bar{M}$ is sent in the ring through hop-by-hop probabilistic encryption. To simplify the notation, the symbol E_i denotes the encryption with the public key of the i-th node of the path from r_1 to w_{j+K} . The message that Alice intends to send to Bob will be included within the Onion wrapping encryption, in the specific position in which only Bob can decrypt. A parameter *dest* is attached to the message to identify the destination step of decryption as well as the NID of the exit node r_j. Similarly, the message received by the exit node contains a parameter *exit* in such a way that that node re-injects in the ring a new empty container. Finally, the last node (that we call *terminal*) of the path decrypts the message $(D, 0)$, that is a dummy message D plus the information 0 that terminates the proxying process. For the response, the terminal node generates constant-rate containers (similarly to rings) that cross the entire route external to the ring (allowing Bob to encapsulate the response messages) until the exit node. Then, such containers are injected into the ring reaching the source (Alice). The cover-traffic burst ends when a *stop message* is sent by Bob in a container, it reaches the exit node, and this node directly contacts the terminal node to turn off the cover-traffic generation. This strategy prevents classic intersection attacks identifying the recipient as the traffic pattern for requests and corresponding responses are the same.

A graphical representation of the protocol running is reported in Figure 2.1. Therein, the sender (Alice) belongs to a ring of 8 nodes (thus, $K = 8$). The arrow marked with (1) represents the DS request sent by Alice to DS through a container reaching the exit node and then going directly to DS. DS responds with the message (2), which reaches the exit node and, then, is injected into the circle through a container that reaches Alice. Thanks to the DS response, Alice can construct the Onion path including Bob (i.e., the inertia route). Then, she sends the message (arrow (3)) through a container until the exit node and, then through the inertia route. Observe that, after reaching Bob, the communication continues in a dummy fashion. This is represented by the dashed trait of the arrow. Finally, thanks to the cover-traffic burst generated by the terminal node, the response reaches the exit node and, then, Alice (arrow (4)). Note that, the communication is dummy until Bob (the dashed trait of the arrow (4)), inserts in the cover traffic the actual response.

2.4 Security Analysis

In this section, we sketch a security analysis of our solution. We start by defining the threat model.

Assumptions A1.

Rings are built in a way that the background knowledge does not allow the adversary to have more information than the sender's uniform distribution. **A1** refers to the realistic case of nodes associated with end users. A similar assumption cannot be done for recipients in the case of web traffic [246], whereas it is valid for P2P applications.

Adversary Model.

We consider a global passive adversary able to monitor all the traffic of the network and to passively control the DS system. This means that each of the nodes can be compromised to the extent that the adversary can observe all the incoming and outcoming traffic from the node as well as the content of DS requests and DS responses.

Security Properties.

We study the security properties *sender anonymity* and *recipient anonymity*. Observe that if one of them is satisfied, then also *relationship anonymity* holds. According to [219], the anonymity of a given item of interest (sender, recipient, relationship, etc.) is guaranteed if the adversary cannot *sufficiently* identify the item in the anonymity set. We observe that we reach this objective when the success probability for the adversary is $\frac{1}{K}$, where K is the size of the rings because K can be realistically set to a sufficiently large value such that the above probability refers to an unlikely event.

Regarding Onion, it is well-known [93] that neither sender anonymity nor recipient anonymity is guaranteed in the global passive adversary model. Indeed, the attacker can observe the traffic from the sender (identifying it) by compromising the first relay node of the Onion circuit. Similarly, the recipient can be identified by compromising the last node of the circuit. Moreover, due to the timing attacks, also the relationship anonymity is not satisfied in Onion.

Now, we analyze sender anonymity for our protocol. We recall that each node can send a message only after receiving a container from the previous node in the ring. When a container is forwarded between two nodes, it is encrypted with probabilistic encryption both if it is filled (i) and if it is empty (ii). The only way for the attacker to distinguish (i) from (ii) is to observe the size of the container. However, this is prevented by padding properly the empty container in such a way that it has the same size as the filled container. Observe that, the probabilistic encryption scheme ensures that, when a sender encrypts more times the same empty container, each encryption results in a different ciphertext. This way, even if the adversary stores any of these ciphertexts, it is unable to understand if a new ciphertext, sent from such a sender, is a filled container or just another empty container. Therefore, the

sender cannot be detected when it fills the container. Consider now the DS request, which is preliminary to the communication. The way in which such a request is performed ensures that it is not possible to link the initiator of the request even if the adversary controls the DS system. Indeed, the request does not contain any information about the initiator and it is sent through the ring until the exit node and from the exit node to DS. This latter replies to the exit node, and the initiator receives the reply through the ring. By Assumption **A1**, since the senders follow a uniform distribution, we obtain that the adversary can break sender anonymity with probability $\frac{1}{K}$, where K is the number of nodes in a ring. Thus, the adversary can break relationship anonymity with probability bounded by $\frac{1}{K}$.

Finally, consider recipient anonymity. The adversary knows that the recipient is one of the nodes in the path from the exit node to the last node, but it does not know where the node is. If we assume uniform destination distribution (occurring when P2P applications are considered), the adversary can break recipient anonymity with probability $\frac{1}{K}$. In the case of skewed destination distribution (occurring for example in Web-browsing), the probability that the adversary identifies the recipient is higher than $\frac{1}{K}$, since some recipients are more likely to be chosen than other recipients. Therefore, the more skewed the destination distribution, the less recipient anonymity is. However, independently of recipient anonymity, relationship anonymity is guaranteed with uncertainty at least $\frac{1}{K}$ thanks to sender anonymity.

2.5 Discussion and Limitations

The protocol presented in this chapter has the objective to introduce the reader to the concepts of ring and Onion routing, which will be exploited in the next chapters. Indeed, even though the idea presented at a high level solves the problem of anonymity against a global adversary, several aspects should be better investigated. First, the security analysis is just sketched and a more formal analysis should be conducted. No implementation is provided and no experimental validation is performed. For example, it is not clear how to set the size of the containers. Indeed, the basic Onion routing approach used requires a size that increases linearly with the length of the path. This may result in an intolerable size. Furthermore, when DS replies to the sender with the public keys of other K users, the size of this message may be very high. Another aspect not addressed in this protocol is fault tolerance taking into account what happens when some nodes fall. The next proposals overcome all the above drawbacks.

3

Providing Tor with sender anonymity against a global adversary

Tor is the de facto standard used for anonymous communication over the Internet. Despite its wide usage, Tor does not guarantee sender anonymity, even in a threat model in which the attacker passively observes the traffic at the first Tor router. In a more severe threat model, in which the adversary can perform traffic analysis on the first and last Tor routers, relationship anonymity is also broken. In this chapter, we propose a new protocol extending Tor to achieve sender anonymity (and then relationship anonymity) in the most severe threat model, allowing a global passive adversary to monitor all of the traffic in the network. We compare our proposal with Tor through the lens of security in an incremental threat model. The results of this approach are published in a research paper [43].

3.1 Introduction

The Tor overlay network [258] is the most popular anonymous communication protocol used for low-latency network applications. It is the state-of-the-art implementation of the Onion protocol [108]. Tor is based on two concepts: relay nodes (also called *Tor routers*) and layered encryption. Relay nodes act as proxies in an Onion route. Each relay node receives its message from the preceding one and forwards it to the next, until the destination is reached. Differently from random walk [230], the route is deterministic and chosen by the sender. Moreover, the message is wrapped through layered encryption, which the sender can apply by knowing the cryptographic keys of all the relay nodes of the route. This way, each node is able to drop an encryption layer, and can see the address of the next relay node to which the still encrypted message should be forwarded. Eventually, the message with only one layer of encryption reaches the destination. According to this scheme, each node in the route only knows the address of the preceding node and the address of the next node. Therefore, by design, the first relay node knows the address of the sender. *Sender anonymity* is then not supported if we allow the adversary to control the first relay node. The practical impact of this weakness is that sole collaboration with an

Internet service provider allows the adversary to detect that a user is utilizing the Tor system. Sender anonymity is obviously broken in a severe threat model with a global passive adversary, able to monitor all the traffic in the network. Anyway, breaking sender anonymity is not enough to nullify the final goal of the protocol, which is *relationship anonymity*. Indeed, the aim of Tor, as in general happens for an anonymous communication network, is to prevent the adversary from detecting that a given sender is communicating with a given recipient. Consider that, despite the fact that anonymity services are often used for criminal purposes, there are a lot of ethical applications of anonymous routing, including censorship resistance. However, relationship anonymity can be broken in Tor in a global passive adversary model. As a matter of fact, Tor is vulnerable to many passive attacks [209, 150], allowing traffic de-anonymization. It can be easily recognized that if the adversary can monitor the traffic at the bounds of the Tor circuit (i.e., the first and the last router), traffic analysis attacks break relationship anonymity [219, 211], thereby fully de-anonymizing the communication.

The solution proposed in this chapter aims to overcome the above drawbacks of Tor, by achieving sender anonymity (in the sense of *communication k-anonymity* [274]) in the most severe threat model, in which a global passive adversary is allowed, which monitors all the traffic in the network. Recall that sender anonymity is enough to guarantee relationship anonymity, as stated in [219]. Therefore, we obtain effective protection for users' privacy.

The approach we use to obtain sender anonymity in Tor exploits the concept of *ring*, introduced in Chapter 2, to hide the sender within an anonymity set of potential senders arranged circularly. To prevent the adversary from detecting the initiator of the communication, we equip the ring with cover traffic that the senders can opportunistically use to send their messages, by filling one or more circulating *tokens*. Thanks to probabilistic encryption, empty and filled tokens are indistinguishable from the adversary. The route Tor is then built from a proxy node of the ring to the destination. The adversary can see that a node of the ring is working as a proxy node, but it is not able to understand which node the sender is among the nodes of the ring. Traffic analysis attacks are not possible due to the cover-traffic mechanism.

3.2 Overview of the Tor Network and Notation

In this section, we introduce the notation used to describe our proposal. Moreover, we provide an overview of the Tor protocol by introducing only the aspects relevant to the solution proposed in this chapter. Further details can be found in Section 4.2 of Chapter 4.

We start with the notation. For both symmetric and public-key encryption, we denote by $E_k(M)$ the encryption of a message M with key k. Similarly, we denote by $D_k(C)$ the decryption of the ciphertext C with (symmetric or public) key k. Even though we do not explicitly highlight this aspect, the encryption we consider is only probabilistic, in such a way that, for an eavesdropper, two different encryptions of the same message are unlinkable.

The Tor network is an overlay network, based on TCP/TLS connections, consisting of multiple relay routers called *Onion routers* (OR). Each client runs locally an *Onion proxy* (OP) which establishes a virtual circuit of ORs to communicate anonymously with the destination. To build a circuit, the OP periodically contacts a trusted server called *Directory Server* (DS) that keeps the information about the state of the network and provides the OP with *router descriptors* of the ORs. These router descriptors contain the IP addresses and the public keys of the ORs, along with their network information, such as the bandwidth. Then, the OP selects, according to some strategies, a number n of OR relays that form the virtual circuit. By default, $n = 3$. The first OR is called the *entry router*, the second the *middle router*, and the last the *exit router*. Once the three ORs have been selected, the OP starts a set-up phase to build the virtual circuit. This phase is performed in such a way that each OR only knows the previous and the next node of the path. Moreover, in this phase, the OP exchanges some messages with the ORs, which include some Diffie–Hellman (DH) parameters, to share a secret key. These messages are encapsulated into *control cells* of a fixed size of 512 bytes. Since the OP has to be sure about the authenticity of the ORs, the DH parameters are encrypted by using the public keys of the ORs. At the end of this set-up phase, the OP shares a secret key with each OR. These keys are used by the OP to encrypt (symmetrically) in Onion fashion the messages intended for the destination. Once the circuit is established, the OP sends the messages to the destination encapsulated into *relay cells* of size 512 bytes. These relay cells include a header of 3 bytes in plaintext plus 11 bytes encrypted for the exit router. Therefore, the effective payload is 498 bytes.

3.3 The Proposed Protocol

In this section, we describe our protocol, which achieves sender anonymity even in the most severe threat model including a global passive adversary. We denote by *(client) nodes* the nodes that collaborate in the protocol without playing the role of Tor routers. Senders are among the client nodes. Moreover, we have in the network n_d destination hosts, which are distinct from client nodes and Tor routers.

The description of the protocol is given in three main steps. The first step is describing the ring manager and the token-based mechanism. Some management functions are illustrated, along with the basic mechanism for implementing anonymity for the sender. The second step is describing the *set-up phase*. This is the phase in which keys are exchanged, the setting of further parameters is executed and cover traffic is established. This is a preliminary step to make possible anonymous communication, which is explained in the last step of the description, denoted as *communication phase*.

3.3.1 Ring Manager and Token-Based Mechanism

In this section, we describe the basic mechanism of our approach that allows us to provide the sender with anonymity against a global passive adversary.

We assume the presence of a *ring manager* (RM) that partitions the nodes of the network in several *rings*.

The ring manager selects the nodes forming a ring in such a way that the background knowledge does not allow a possible adversary to have more information than the uniform distribution of senders. In other words, given a ring, any node of the ring is potentially a sender (with no probability bias). This is achieved by selecting, for a given ring, hosts belonging to the same, even large, geographical region.

A ring is a sequence of k nodes such that each node has exactly a *preceding* (*prec*, for short) and a *next* node. In our setting, each node only knows its prec and its next node. Several messages, called *tokens*, move through the ring. There are two kinds of tokens. The first type is used in the set-up phase. The second type is used in the succeeding communication phase. The detail will be discussed next. Tokens are filled by senders to deliver their messages to a proxy node, which, once a Tor circuit is established, sends them to the destination host. To obtain that any eavesdropper is unable to distinguish an empty token from a filled token, each node encrypts the token with a symmetric key shared with its next node.

RM maintains, along with the next node, the public keys and the network addresses of each node of the network. For each ring, each belonging node receives from RM the set of the public keys of the other nodes of the ring, and among these keys, the information about which is the public key associated with the next node in the ring. We assume that RM is a centralized entity.

3.3.2 Set-Up Phase

The first purpose of this phase is to exchange a set of symmetric keys between the nodes of a ring. These keys will be used to encrypt the messages without requiring the complexity of public-key encryption.

We first introduce some notation. Given a ring, we denote by $r_1, \dots r_k$ the k nodes forming the ring, in order. Given a node r_i, we denote by $next(r_i)$ the next element in the ring, that is, $r_{(i\%k+1)}$, where % is the operator *mod*. We denote by PK_{r_i} the public key associated with the node r_i and by $addr(r_i)$ its network address.

Now, we can describe how key exchange is executed. This is done in detail next. We have two kinds of key exchange. The first is aimed at providing each node with a symmetric key shared with the next node. These keys are used to implement hop-by-hop encryption when messages turn in the ring. This key exchange is called *forward key exchange*, and it is described in detail next, in Section 3.3.2.

The second kind of key exchange is aimed at obtaining key sharing between the sender and the proxy node. However, since both roles of sender and proxy can be played by all the nodes in the ring, the key exchange mechanism involves every pair of nodes. Synthetically, each node of the ring exchanges a symmetric key with the other $k-1$ nodes. Observe that, even though a key is exchanged between two nodes A and B, a different key will be exchanged between B and A. Indeed, the two keys will be used for different purposes depending on whether the node plays the role of *sender* or *proxy*. Therefore, the two keys are called the *sender key* and *proxy key*, respectively. A requirement of this phase is that, if A exchanges a key with B, B learns nothing about the network address of A. The detail of this mechanism, called *sender and proxy key generation*, is provided next, in Section 3.3.2. Since the above keys will be included in special tokens, before describing the key generation mechanism, we describe, in Section 3.3.2, how such tokens are arranged.

Forward Key Exchange

Each node r_i receives from RM the set Q of the public keys of the nodes of the ring it belongs to, $addr(next(r_i))$, and among Q, the information about which public key is associated with $next(r_i)$ (the associations of the other keys with the proper network address remain unknown to r_i). The address of the next node will be used to forward tokens.

Initially, each node r_i exchanges a symmetric key called *forward key* with its next node. This key is used only to encrypt the token hop-by-hop. In detail, each node r_i generates a public DH parameter y_i and encrypts it with the public key $PK_{next(r_i)}$, obtaining $C = E_{PK_{next(r_i)}}(y_i)$. C is sent to $next(r_i)$ (we recall that r_i knows $add(next(r_i))$. The latter decrypts y_i, generates the *forward key* k_{r_i} and replies to r_i with its public DH parameter $\bar{y}_{next(r_i)}$ along with the hashed value $H(k_{r_i})$ (in plaintext). In summary, each node r_i shares a forward key k_{r_i} with its next node, and the tokens can be properly encrypted hop-by-hop.

Token Generation

After exchanging the forward keys, at a given time t_0, each node r_i generates $k-1$ empty tokens and sends them to its next. In turn, $next(r_i)$ forwards the tokens to its next, and so on. Each token is encrypted by r_i with k_{r_i}; then it is sent to $next(r_i)$, which decrypts it with k_{r_i}, processes the token, re-encrypts it with $k_{next(r_i)}$ and forwards it to $next(next(r_i))$.

The structure of these tokens is the following: $\langle F, PDH, R, H\rangle$ where F is a flag denoting whether the token is empty ($F = 0$) or filled ($F = 1$), PDH is a field containing a public Diffie–Hellman parameter (possibly encrypted), R is a random playing the role of identifier and H is a hashed value (the exact meaning of these fields will be clear in the following). Observe that PDH, R and H are meaningful only if $F = 1$. The tokens are born with $F = 0$. Therefore, at the beginning, there are $k(k-1)$ empty tokens turning in the ring.

Starting from a time $t_1 > t_0$, each node r_i waits a random time δ_i, and then fills the first available empty token, as explained in the following.

Sender and Proxy Key Generation

First, F is set to 1. Then, r_i selects a random public key PK_{r_j} from $Q \setminus \{PK_{r_i}\}$. r_i selects its public DH parameter y_{ij} and encrypts it with PK_{r_j}, thus obtaining $C_{ij} = E_{PK_{r_j}}(y_{ij})$. Then, PDH is set to C_{ij}. R is set to a random value used by r_j to reply with its public DH parameter, which is needed by r_i. This DH parameter is used in the construction of the key that r_i will use to send a message by using r_j as a proxy. This key k_{ij} is called the *sender key* for r_i (with respect to r_j), and the *proxy key* for r_j (with respect to r_i). Finally, H is filled with random bits.

The token T is encrypted by r_i with k_i, by obtaining $C_T = E_{k_i}(T)$. Then, C_T is sent to $next(r_i)$.

When C_T reaches $next(r_i)$, it decrypts C_T, by obtaining T, and since $F = 1$, it tries to read the field $PDH = C_{ij}$ of T. If $next(r_i) \neq r_j$, $next(r_i)$ is not able to decrypt such a field, and then it re-encrypts the token with the forward key $k_{next(r_i)}$ shared with $next(next(r_i))$ and forwards the token. The token moves through the ring until it reaches r_j. At this point, r_j decrypts C_{ij} and obtains y_{ij}, with which it generates the key k_{ij} which is shared with r_i. The token is filled as follows. F remains set to 1. PDH is set to $\bar{y}_{ji}$. $\bar{y}_{ji}$ represents the public DH parameter of r_j that will be used by r_i to generates the key k_{ij}. R remains unaltered, and finally, H is set to the hashed value $H(k_{ij})$. This new token moves through the ring until r_i. Observe that all the nodes between r_j and r_i, after decrypting the token with their forward keys, understand

that the token is used to reply to a node, but are unaware of the sender and the recipient of this token.

When r_i receives the token, it identifies the token as a reply of r_j thanks to the random R. Then, r_i can generate the key k_{ij} as r_j. This token is then dropped by r_i. Finally, r_i drops from the set Q the node r_j. Note that any external observer only knows that a key was exchanged by a given node r_i, but does not know with which node.

The entire process (which started at time t_1) is repeated $k-2$ times, until all k_{iy} are exchanged.

When all the $k(k-1)$ tokens are disposed of, each node r_i owns (in addition to the forward key) two symmetric keys k_{ij} and k_{ji} shared with each other node r_j of the ring. The key k_{ij} represents a sender key for r_i, since it is used by r_i when has to send a message by selecting r_j as a proxy node (see next section). On the other hand, k_{ij} represents a proxy key for r_j, since it is used by r_j when playing the role of a proxy node.

In Figure 3.1, the sequence diagram of the set-up phase is depicted.

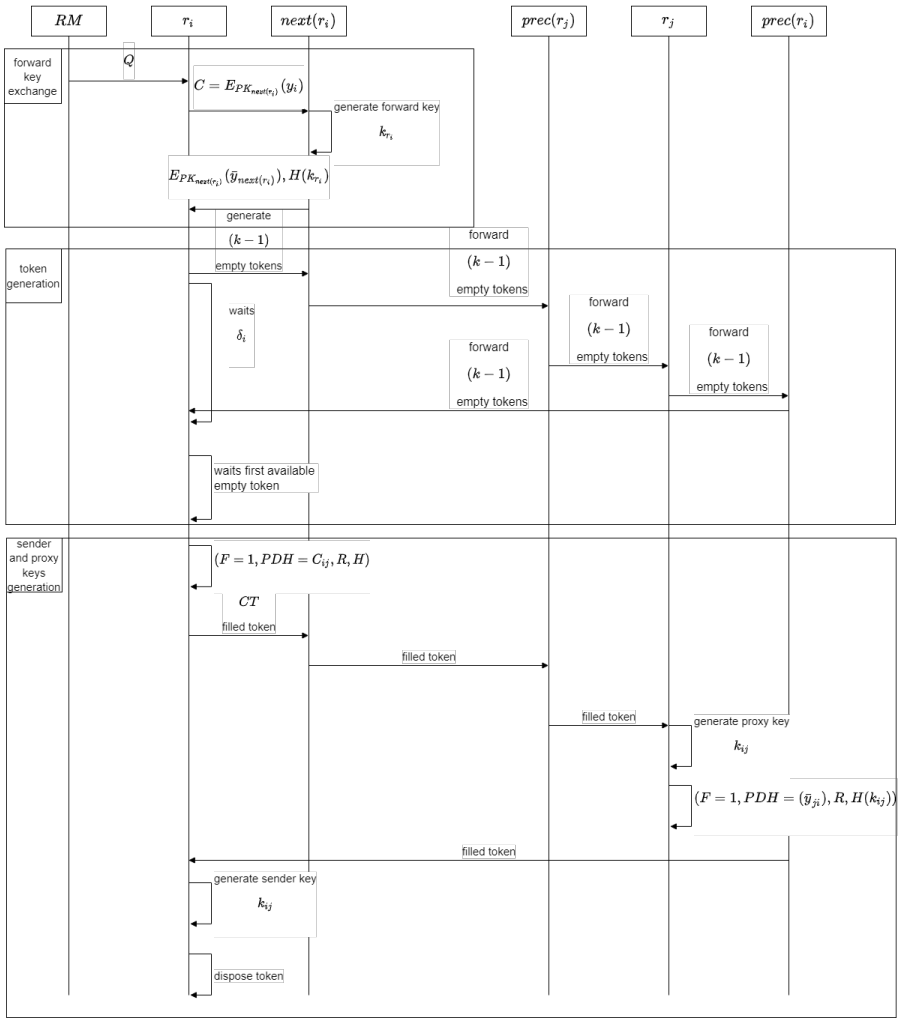


Fig. 3.1: Sequence diagram of the Set-up phase.

3.3.3 Communication Phase

In this section, we describe the core of our protocol, which is the communication between a sender and recipient. We remark that the communication is bi-directional, in the sense that we address both the request and the response. We split the description of the communication phase into three parts. The first part is the structure of tokens in which messages are encapsulated. Observe that these tokens are different from those used in the set-up phase, which we described in Section 3.3.3. After describing the structure of the tokens, we show how tokens are generated (see Section 3.3.3. Finally, in Sections 3.3.3 and 3.3.3, we describe how anonymous communication is established between a sender and a recipient.

Structure of the Token

As in the set-up phase, in the communication phase, a token-based mechanism is enabled. We assume that a given number of tokens move through the ring encrypted, hop-by-hop, from one node to the next, with the forward key exchanged in the set-up phase.

The structure of a communication-phase token is the following: $\langle F, HID, CI, DA, P \rangle$. In Figure 3.2, an expanded description of this structure is reported.

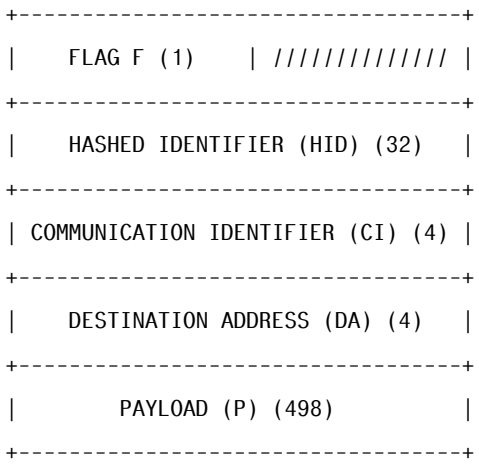


Fig. 3.2: Structure of the token.

As the communication phase is the core of our protocol, we describe in detail how the token is organized. Its size is 539 bytes, of which 41 are reserved for the header, and 498 for the payload. The size of the payload is set to the same value as the size of the payload of the relay Tor cells.

First, we describe the meaning of the field F. It is composed of two bits (even though we reserve 1 byte for this field), with the following possible meanings: 00 means *empty* token; 01 means token *reserved* for a given communication identifier; and 10 means that it is *used* for a message. A token in the state 01 (reserved) or 10 (used) is said to be *filled*.

During the description of the protocol, which we provide next, the meanings of the remaining fields are clarified.

Token Generation

Consider now the process of token generation. When a token is generated by a node r_g, the fields are set as follows. F is set to used (i.e., 10). r_g picks randomly from the set Q (where Q is the set of all the public keys of the ring) a public key, say PK_{r_p}, associated with the node r_p. The field HID is set to $H(PK_{r_p})$. It is used as

an identifier to allow r_p to recognize that this token is intended for it. Finally, the field DA includes the encryption $\bar{S}$ with the sender key (of r_g) k_{gp} of a fixed string S different from any other network address. This string allows r_p to identify the fact that this token, even if used, does not contain any message to forward outside the ring (see below), but it has to be emptied by r_p. The reason why the token is not directly generated empty derives from security aspects. The security analysis is provided in Section 3.6. The other fields (CI,P) are filled with random bits.

The entire token is then encrypted with the forward key k_{r_g} and sent to $next(r_g)$. This node decrypts the token, and with the state of the token being filled, through the field HID, it checks whether this token is intended for it. In this case (i.e., $r_p = next(r_g)$), it processes the token. Otherwise, the token is encrypted, as usual, by $next(r_g)$ with the forward key $k_{next(r_g)}$ and sent to $next(next(r_g))$. The token moves through the ring until it reaches r_p.

At this point, r_p verifies that it has been selected as the recipient of the token, even though it does not know that the token was generated by r_g. Therefore, r_p tries to decrypt the fields CI, DA, P with all its $k-1$ proxy keys until it finds the correct key k_{gp}. Since $D_{k_{gp}}(DA) = S$, r_p knows that it has to empty the token. Thus, r_p sets F to 00 and $HID = H(PK_{next(r_p)})$. In this case, we say that $next(r_p)$ will play the role of *proxy node* (with respect to a potential sender for a communication identifier not established yet). The other fields are set to random bits.

r_p encrypts the token with the forward key k_{r_p} (shared with its next) and forwards it to $next(r_p)$. The empty token crosses the ring encrypted hop-by-hop, as usual.

The process of generation of the tokens is represented in the sequence diagram in Figure 3.3.

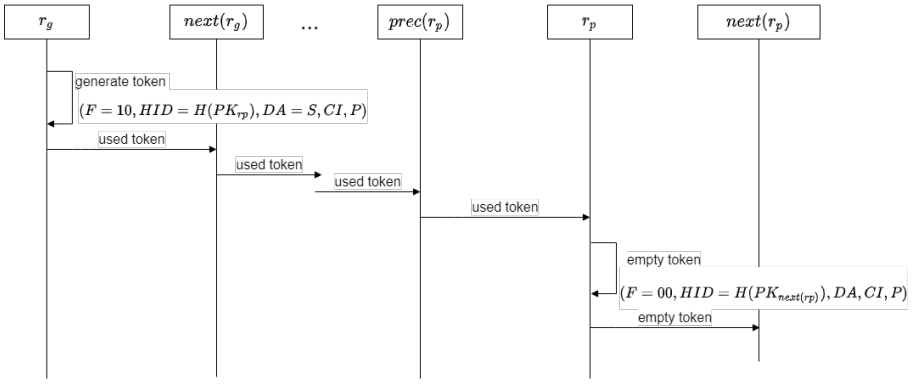


Fig. 3.3: Process of generation of the tokens.

Transmission of a Message

Consider a node r_i that wants to send a message M to a destination D (outside the ring). Suppose M is already encrypted for D. First, r_i splits M into blocks $M_1, \ldots, M_q$ ($q \geq 1$) with size 498 bytes (i.e., the size of the payload P of a token). r_i waits for the first empty token (with $F = 00$). Let be $HID = H(PK_{r_j})$ (this means that r_j will play the role of proxy node for a communication session started by r_i, as we will see next). Through HID, r_i identifies the public key PK_{r_j} and the corresponding sender key k_{ij}.

The token is filled as follows. F is set to 10 (used). $HID = H(PK_{r_j})$ is unaltered. CI is set to $E_{k_{ij}}(R)$ where R is a random value identifying the current *communication session* associated with the sender key k_{ij} (note that for a given communication session, a Tor circuit will be established outside the ring). The field DA includes the encryption with key k_{ij} of the network address of the destination D. Observe that the size of this field is 4 bytes, and thus is compliant only with IPv4. Obviously, for IPv6, the size should be increased. Moreover, the TCP port is not included in this field for privacy reasons. It will be included in the payload encrypted at the application layer. Finally, P is set to $E_{k_{ij}}(M_1)$ (possibly padded, if $q = 1$). The token moves through the ring (encrypted hop-by-hop) until it reaches r_j.

Regarding the other messages M_t (with $2 \leq t \leq q$), r_i waits for either (1) an empty token with $HID = H(PK_{r_j})$ or (2) a reserved token ($F = 01$) with $HID = H(R)$, meaning that the token is reserved for the communication session started by r_i identified by R.

In both cases, the token is filled as follows. F is set to 10, HID is set to $H(R)$ in case (2) (indeed, in case (1) it is already set with this value), $CI = E_{k_{ij}}(R)$, DA includes the encryption with key k_{ij} of the network address of the destination D and $P = E_{k_{ij}}(M_t)$. Additionally, these tokens move through the ring until they reach r_j. Eventually, all the blocks of the message M reach the same proxy node r_j, which will use the same Tor circuit.

We now see how such a Tor circuit is established by r_j. When r_j receives the (used) token containing M_1, r_j identifies this token through $HID = H(PK_{r_j})$. Anyway, it does not know the sender r_i. Therefore, r_j tries to decrypt the fields CI, DA, P with all its $k-1$ proxy keys until it finds the correct key k_{ij}. Since $D_{k_{ij}}(DA) \neq S$ (we recall that S is a fixed string denoting that the token does not contain a message), r_j has to send the message outside the ring to the destination D through the Tor system.

Before doing this, r_j sets the flag $F = 01$ (reserved) and the field $HID = H(R)$ where $R = D_{k_{ij}}(CI)$. This means that this token is associated with the communication session identified by R. R is also stored by r_j and associated with k_{ij} in such a way that further tokens can be associated with this communication session. The random

R is also used by r_i to detect further reserved tokens for this communication session. The other fields are filled with random bits and the token is then forwarded into the ring.

At this point, r_j can send the message $M_1 = D_{k_{ij}}(P)$ to the destination D. To do this, it builds a Tor circuit with destination $D_{k_{ij}}(DA)$ and sends the message M_1 to D through this circuit. The construction of the Tor circuit is performed in the standard way, by contacting the Directory Server (DS) and by selecting the entry, middle, and exit nodes as illustrated in Section 3.2.

When r_j receives a (used) token containing a message M_t with $2 \leq t \leq q$, r_j identifies such token through HID and forwards M_t to D through the Tor circuit. The token is set to reserved ($F = 01$) and HID remains unaltered to the value $H(R)$. The other fields are set to random bits, and the token is then forwarded into the ring.

The transmission of the message M is represented in the sequence diagram in Figure 3.4.

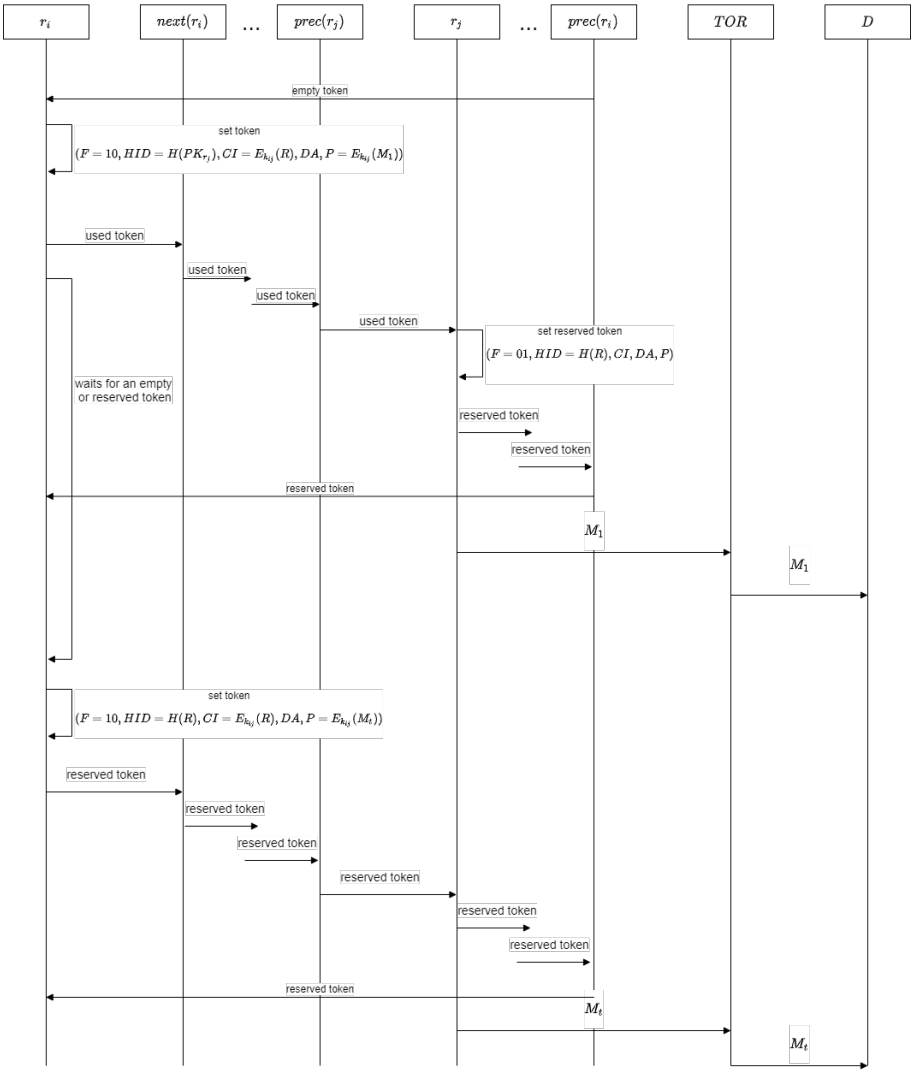


Fig. 3.4: Transmission of the message M.

Transmission of the Response

When r_j receives the response M' (already encrypted by D) through the Tor circuit, r_j injects the response into the ring. Specifically, let $P'_1, \dots P'_l$ be the Tor cells including the response M', and let denote by P_k the payload of the cell P'_k ($1 \leq k \leq l$). For each P_k, r_j waits for either (1) an empty token or (2) a reserved token with $HID = H(R)$. The token is filled as follows. F is set to 10. In the case of an empty token the field HID is set to $H(R)$. The communication identifier CI is derivable by the random R associated with the current communication session; and then, with this Tor circuit stored by r_j when the Tor circuit has been established. Specifically, $CI = E_{k_{ij}}(R)$. The field DA is filled with random bits. Finally, P is set to $E_{k_{ij}}(P_k)$.

At this point, the token moves through the ring and is identified by r_i through HID. When r_i receives all the tokens containing the block P_k, it retrieves the entire

response M'. For each of these tokens, r_i changes the state from used to reserved and forwards the token. Specifically, F is set to 01, $HID = H(R)$ is unaltered and the other fields are filled with random bits. These reserved tokens (along with other possible empty tokens) are used by r_i and r_j to exchange the other requests/responses associated with the communication session identified by R.

The transmission of the response M' is represented in the sequence diagram in Figure 3.5.

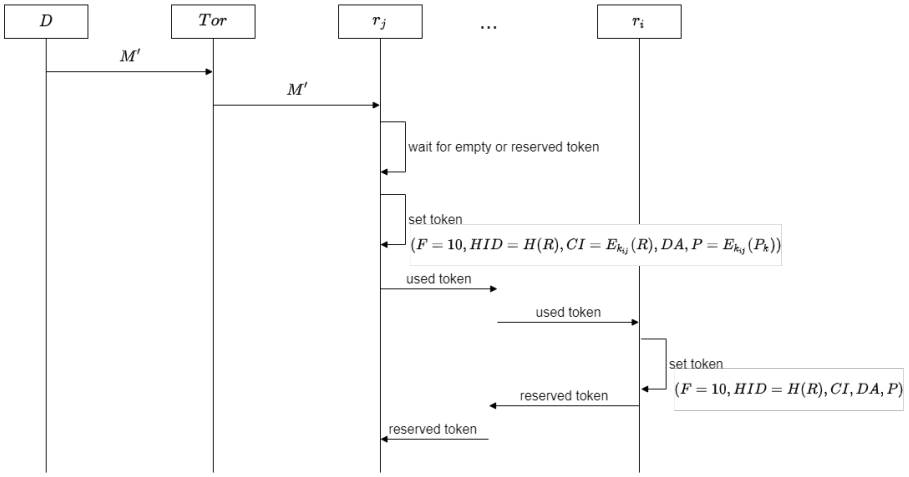


Fig. 3.5: Transmission of the response M'.

When the communication session ends, r_i and r_j perform some actions aimed at emptying the tokens reserved for this session and destroying the Tor circuit. Specifically, for each reserved token with $HID = H(R)$, r_i fills the token in such a way that r_j recognizes that they have to be emptied. To do this, F is set to 10, $HID = H(R)$ remains unaltered, CI is set to $E_{k_{ij}}(R)$, DA is set to the encryption with key k_{ij} of S and P is filled with random bits.

When r_j receives such a token, it retrieves the string S and recognizes that the session deactivation actions have to be performed. If this token is the first including S, r_j destroys the Tor circuit. For this token and the successive ones, including S, r_j empties them and forwards them into the ring. Specifically, F is set to 00 and $HID = H(PK_{next(r_j)})$. The other fields are filled with random bits.

This process of emptying the tokens and destroying the Tor circuit is represented in the sequence diagram in Figure 3.6.

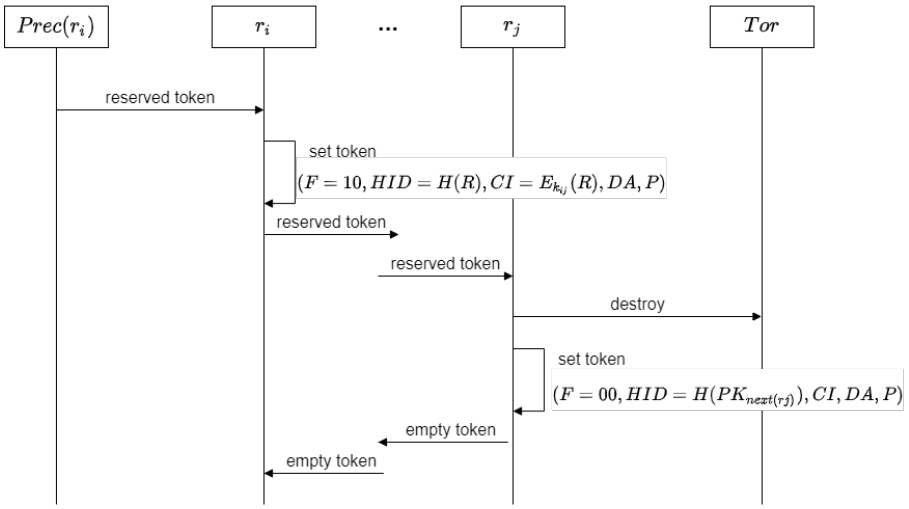


Fig. 3.6: Process of emptying the tokens and destroying the Tor circuit.

To conclude this section, we provide a brief summary, by omitting the technical details of the communication phase. In Figure 3.7, we sketched a high-level graphical representation of this phase.

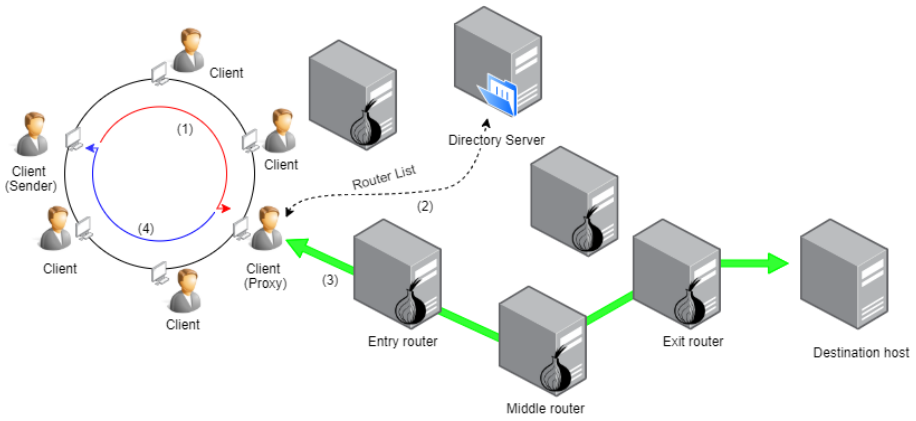


Fig. 3.7: Communication phase (overview).

The sender waits for an empty token, selects a proxy node, and fills the token with a message. This token will be injected into the ring, in which it will move until the proxy node is reached. The path of the ring from the sender to the proxy node is represented with a red arrow. Once the proxy node receives the message (possibly, encrypted), it contacts the Directory Server (dashed arrow) to select the entry, middle, and exit routers and builds a Tor circuit through them. At this point, the proxy node forwards the message through this Tor circuit until the destination. The latter will provide the response (possibly encrypted) through the same Tor circuit until the proxy node. Both the ongoing path and the return path are represented by

the green arrow in the figure. Finally, when the proxy node receives the response, it waits for a number of empty or reserved tokens and fills them with the response. These tokens are injected into the ring until they reach the originator of the request. The path of the ring from the proxy node to the originator, traversed by the response, is represented with a blue arrow.

3.4 Fault Tolerance

Even though fault tolerance is one of the aspects that is typically missed in anonymous communication networks, we sketch in this section how a certain degree of fault tolerance can be easily introduced in a system based on our protocol. To confirm the above claim, consider the current Tor itself has no fault tolerance at all. Indeed, if a Tor router stops working during a communication, the communication is lost, and there is no protocol to recover the communication on the fly (indeed, setting a backup Tor circuit is not enough to obtain this goal). As we focus on the part of the proposal that plays the role of *add-on*, with respect to the existing Tor system, we do not consider in this section the Tor communication occurring outside the system, between the proxy node and the destination. Apart from the fact that the fault tolerance of Tor can be considered an orthogonal problem, it is also true that Tor routers can be considered more stable than standard client nodes involved in the rings.

The basic change we have to introduce to obtain fault tolerance is the notion of a *ring layer*. The ring manager, instead of building simply rings of k nodes, builds rings of k layers, each composed of j nodes. We can figure out that the value of j, for good fault tolerance, should be very low (for example, 2 or 3) if we are in a network with a high level of activity. Anyway, higher values of j do not result in infeasible computation, as we will see next. The nodes of each layer know each other in the sense that they are aware of the reciprocal addresses. With the notation $r_1,\dots r_k$, used earlier for the rings, now we indicate a sequence of layers, such that $r_i = \{x_1^i,\dots x_j^i\}$ is a set of j nodes. Besides the individual public keys of the nodes, there is also a public key per layer, called the *public layer key*. This impacts both the set-up phase and the communication phase. Concerning the set-up phase, some changes occur for the key exchange task. Forward keys are exchanged for each pair $x_p^i, x_q^{i\%k+1}$, $(1 \le p,q \le j)$. Thus, we have j^2 forward key exchanges per pair of consecutive layers. Instead, by leveraging public layer keys, the pair of keys used as sender key and proxy key k_{st} and k_{ts} will be established between layers instead of individual nodes. To do this, the ring manager selects one *representative* node alive per layer and informs each selected node about the other selected nodes (and then about their public keys). Then,

the Diffie–Hellman process described in Section 3.3 happens among these representative nodes. At the end of this process, any representative node has a pair composed of a sender key and a proxy key between its layer and any other layer. These keys are exchanged with all the other nodes in the layer. Indeed, in the pre-set-up phase, the nodes of the same layer exchange a symmetric key per pair, by enacting the $j(j-1)$ Diffie–Hellman processes.

Concerning both the circulation of tokens and the communication task, the only change is that the function *next*, associating to each node of the ring the next node to forward a message, becomes non-deterministic. Specifically, a node in layer s that has to forward a message just has to choose one alive node in the layer $next(s)$ and forward the message to it. For the proxy node, essentially no change is required because the encryption is done for the layer so that any node in the layer is able to decrypt the message and then initiate the Tor circuit. Similar considerations can be made for the response.

To conclude this section, we evaluate our fault-tolerance mechanism from a probabilistic perspective, to allow the correct setting of the parameter j, once a given reliability probability is fixed. We denote by p the probability that, at a given instant, a node is alive. We assume p is the same for each node. Therefore, the probability that, given a layer of j nodes, at least one node of the layer is alive is $p' = 1-(1-p)^j$. To guarantee reliability (i.e., the communication is not lost), at least one node per level (for the k levels) has to be alive. Therefore, the probability that the communication succeeds is $p'' = (1-(1-p)^j)^k$. Clearly, it decreases as k increases and increases as j increases. Suppose now we set the reliability threshold to a given value τ. Then, j must set in such a way that $j > \frac{log(1-e^{\frac{log(\tau)}{k}})}{log(1-p)}$.

In Figure 3.8, we set $\tau = 0.999$ and show how the ratio $\frac{j}{k}$ varies for different values of p and k.

Observe that the exemplified value chosen for τ refers to a very reliable system. Indeed, according to the standard IEC 61508, this value falls into the range of probability of failure on demand (PFD), classifying the system as reliability class SIL 3, which is the second most-reliable class.

As expected, for high values of p, the number of nodes j (and then the ratio $\frac{j}{k}$) required to obtain $\tau = 0.999$ decreases. Regarding k, as k increases the absolute value of j increases but slower than k. Therefore, the ratio $\frac{j}{k}$ increases with k.

To give a practical example, with $k = 100$ and $p = 0.9$, we obtain a ratio $\frac{j}{k} = 0.05$, which means that each layer of the ring has to contain only five nodes.

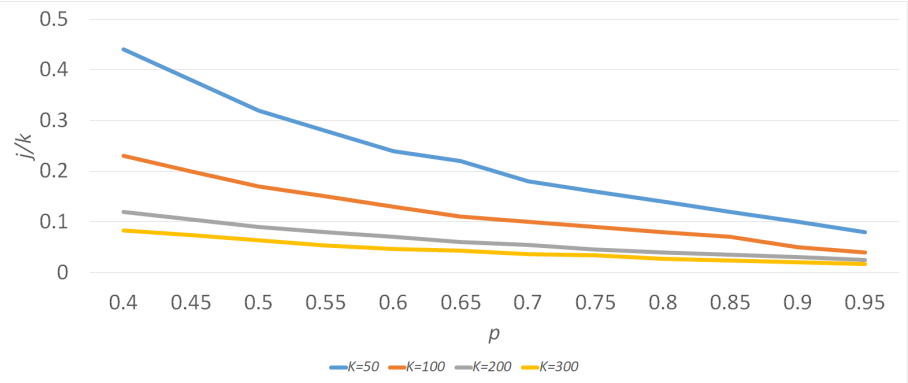


Fig. 3.8: Ratio $\frac{j}{k}$ as k and p vary.

3.5 Computational Complexity

In this section, we discuss the computational complexity of our protocol. We focus on the part of the protocol regarding the ring. Indeed, for the rest of the protocol, involving just a Tor circuit, the reader may refer to the results available in the literature [258].

The communication phase requires, besides the hop-by-hop encryption of the messages (which is standard in any protocol supporting secure communication), the attempted decryptions that the intended proxy node has to perform before sending the message outside the ring. On average, there are $\frac{k-1}{2}$ decryptions applied only to the first token of a given communication (recall that the size of a token is about 500 bytes). In the worst case, there are $k-1$ decryptions. This overhead does not appear relevant, as it regards only the proxy node, and for good privacy levels (e.g., $k = 100$), the extra time required is small. Observe that the magnitude of an AES encryption/decryption is 10^2 Mbytes per second on standard personal computers.

Now, we consider the set-up phase.

First, consider the protocol without fault tolerance (see Section 3.3). Similarly to the Tor set-up phase, we require k key exchanges for the forward keys and $k(k-1)$ key exchanges for sender/proxy keys. For values of k guaranteeing a good anonymity level, the cost of this phase is not prohibitive. When fault tolerance is included, we pay a price in terms of the complexity of the set-up phase. Indeed, we require $j(j-1)$ key exchanges per layer in the pre-set-up phase, and then j^2 forward key exchanges per pair of adjacent layers (executed in parallel) plus $k(k-1)$ exchanges for sender/proxy key exchanges. In summary, we increase the previous cost by $j(j-1)+j^2$. Due to the fact that we expect that j is very small, this computational overhead does not appear as an actual issue for the protocol. Recall that the set-up

phase, differently from Tor, is not done for each communication, but it is done to set up the network, so it can be considered an operation with a long-term lifetime.

3.6 Security Analysis

In this section, we analyze the security of our solution. We start by defining the threat model we consider. We introduce the following assumption:

Assumption 1 (A1). *Rings are formed in such a way that the background knowledge does not allow the adversary to have more information than the sender's uniform distribution.*

Observe that Assumption A1 is easily satisfied if rings are built among hosts belonging to the same, even large, geographical region.

Adversary Model (AM). We consider four types of adversaries.

- **External (E).** In this case, the adversary monitors the incoming and outgoing traffic of the DS. In addition, in our proposal, the adversary monitors traffic coming in and going out from the RM.
- **Weak (W).** In this case, the adversary monitors the traffic between a client node and the entry Tor router. In Tor, the client node corresponds to the OP. To be fair, in our proposal, we allow the weak adversary to monitor all the traffic between the client nodes and the traffic between the client node playing as a proxy and the entry Tor router.
- **Strong (S).** In this case, the adversary monitors the traffic between a client node and the entry Tor router and the traffic between the exit Tor router and the destination host. In our proposal, in addition, the adversary can monitor all the traffic between the client nodes.
- **Global (G).** In this case, the adversary monitors all the traffic of the network.

Furthermore, for all four adversaries, regarding our proposal, we enable another capability: the adversary knows the entire composition of the rings.

Observe that the capabilities of Global, Strong, and Weak adversaries are in order (i.e., Global is stronger than Strong and Strong is stronger than Weak). Furthermore, Global is stronger than External.

Both the External adversary and the Weak adversary model refer to a very feasible case in which an entity is able to control just an autonomous system. The feasibility of the External adversary can be contrasted by distributing the DS and the RM. The Strong adversary is a weak form of the Global adversary, because the autonomous systems of the entry router and exit router can be very far from each other and even be in different continents [17]. The Global adversary is the standard global passive adversary.

Security properties. We analyze two security properties (see Section 3.2): (1) Sender anonymity (SA); (2) Relationship anonymity (RA).

In the following analysis, we discuss how Tor and our proposal behave with respect to the security properties in the four adversary models. The results of the analysis are summarized in Table 3.1. First, we give a preliminary basic result in the following lemma.

Lemma 3.1. *In our proposal, a ring of size k is a sender anonymity set of size k against the Global adversary.*

Proof. Due to the hop-by-hop probabilistic encryption mechanism that is used to move tokens inside the ring, the only point of the ring from which the adversary can draw some information more than a random guess to identify a sender is the proxy node. Indeed, this is the only point of the ring in which the possible state transitions of a token could be in principle related to the observable incoming or outgoing traffic in/out of the proxy. Transitions occurring in other points are not identifiable with probability higher than $\frac{1}{k}$. Since reserved and used tokens cannot be filled by other client nodes different from the sender (associated with the reserved tokens), the only possibility for the adversary to identify a sender anonymity set of size less than k is to detect an empty token outgoing from a node and track it until it reaches a proxy node, which sends a message outside the ring before doing less than k steps. The only event in which the adversary can guess that a token is emptied is when a proxy node, say r_x, dismisses a Tor circuit. Indeed, according to the protocol, there is no other case in which tokens are emptied. However, r_x sets the field HID to $H(PK_{next(r_x)})$, and this means that such a token moves around the entire ring (in which it is, possibly, filled) before reaching $next(r_x)$, which possibly builds a Tor circuit outside the ring. Therefore, we can argue that the sender anonymity set has at least size k, even for the Global passive adversary. The proof is then concluded.

The above lemma is the basis for the fulfillment of the security properties offered by our proposal.

This is proven through the following theorems. The first theorem states that Tor does not guarantee **SA** against any adversary. This corresponds to the first four fields of the first row of Table 3.1.

Theorem 3.2. *In Tor, any adversary breaks **SA** with probability 1.*

Proof. Consider the External adversary. Since it observes the traffic intended for the DS, it receives the request of the sender and then the sender is identified. Since the Global adversary has the same capabilities as the External adversary, **SA** does not hold against it. Now, we consider the Weak adversary able to observe the traffic

Table 3.1: Comparison between Tor (**T**) Our proposal). Shown are the probabilities of the adversaries breaking the properties **SA** and **RA**.

	SA				**RA**			
AM	**E**	**W**	**S**	**G**	**E**	**W**	**S**	**G**
T	1	1	1	1	$\frac{1}{n_d}$	$\frac{1}{n_d}$	1	1
Our proposal	$\frac{1}{k}$	$\frac{1}{k}$	$\frac{1}{k}$	$\frac{1}{k}$	$\frac{1}{n_d \cdot k}$	$\frac{1}{n_d \cdot k}$	$\frac{1}{k}$	$\frac{1}{k}$

between the sender and the entry Tor router. Clearly, **W** identifies the sender. The Strong adversary has the same capabilities as the Weak adversary. The proof is then concluded.

Now, we prove that our proposal guarantees that a sender can be identified (by any adversary) with probability $\frac{1}{k}$. This corresponds to the first four fields of the second row of Table 3.1.

Theorem 3.3. *In our proposal, any adversary breaks **SA** with probability* $\frac{1}{k}$.

Proof. Consider the Global adversary **G**. By Lemma 3.1, it can identify the sender with a probability not higher than $\frac{1}{k}$. Since **G** is stronger than all the other adversaries (i.e., **S**, **W**, and **E**), we conclude that for those three adversaries also, **SA** is broken with a probability not higher than $\frac{1}{k}$.

Now, we have to consider the remaining fields of Table 3.1 regarding relationship anonymity. These are covered by the following two theorems.

Theorem 3.4. *Let n_d be the size of the recipient anonymity set. In Tor, the External and Weak adversary break RA with probability* $\frac{1}{n_d}$. *Furthermore, the Strong and Global adversary break RA with probability* 1.

Proof. Consider the External adversary. By Theorem 3.2, it identifies the sender SN of a communication with probability 1. Anyway, **E** has no information about the recipient R of such a communication. Therefore, **E** (without further knowledge) identifies that SN communicates with R only with the smallest probability, i.e., $\frac{1}{n_d}$.

Similarly, the Weak adversary identifies the sender with probability 1, but has no information about the recipient. Therefore, **RA** is broken with probability $\frac{1}{n_d}$.

Consider the Strong adversary **S**. Since it monitors the outgoing traffic from the exit Tor router, it can identify the recipient R of a communication with probability 1. Since **S** also monitors the traffic between the sender SN and the entry Tor router, it can perform traffic analysis attacks [17] and identifies that SN communicates with R with probability 1. The Global adversary has the same power as the Strong adversary. The proof is then concluded.

Theorem 3.5. *Let n_d be the size of the recipient anonymity set. In our proposal, the External and Weak adversary break RA with probability $\frac{1}{n_d \cdot k}$. Furthermore, the Strong and Global adversary break RA with probability $\frac{1}{k}$.*

Proof. Since **SA** implies **RA** [219], by Theorem 3.3, it follows that **RA** can be broken with a probability not higher than $\frac{1}{k}$ by any adversary. Consider now the adversaries E and W. Even though they can identify the sender with a probability not higher than $\frac{1}{k}$, they do not have any information about the recipient. Therefore, they can only guess the recipient among all the possible recipients of the network n_d. Therefore, for E and W, RA is broken with a probability not higher than $\frac{1}{n_d \cdot k}$. For the other adversaries (i.e., S and G), the above upper bound of the success probability cannot be decreased, because both S and G are able to identify the recipient so that the probability of breaking **RA** is the same as the probability of breaking **SA**. The proof is then concluded.

This ends the security analysis. As is evident by Table 3.1, the benefit in terms of security of our proposal can be measured as a multiplicative factor k, increasing the degree of anonymity provided by Tor both for **SA** and **RA**.

3.7 Related Work

The issues most relevant to our proposal are the vulnerabilities Tor suffers from.

As stated by the creators themselves [258], the Tor overlay network, based on Onion routing [108], does not provide anonymous guarantees in the severe threat model of a global passive adversary [209], which is able to observe the entire traffic of the network.

Anyway, even if we relax the powers of the adversary, many attacks are still effective [150, 239, 91]. The most famous class of attacks is represented by the traffic analysis attacks [17, 88, 193] in which the adversary analyzes the traffic to find correlations. Among the traffic analysis attacks there are the timing attacks [167, 259, 107], in which the adversary observes the timing of the messages arriving at and leaving from the nodes to find correlations. Other interesting subclasses of traffic analysis attacks are traffic confirmation attacks [231], in which the adversary controls and observes two possible end-relays of a Tor circuit to conclude that they really belong to the same circuit, and watermarking attacks [136], in which the adversary manipulates the traffic stream by introducing an identifiable pattern. Another category of attacks targets the router selection used to build the Tor circuit. Indeed, the standard selection is based on network and CPU performance reported by the nodes themselves. This enables self-promotion attacks [247]. A countermeasure can be found in [146].

The performance of Tor was investigated in [19, 213, 157]. Performance analysis in relation to de-anonymization attacks was performed in [51].

In our approach, we extend Tor achieving sender anonymity (and then relationship anonymity) [219] in the sense of *communication k-anonymity* [274], against a global passive adversary. This goal can be reached only with the introduction of cover traffic [71] (as required by our approach).

A solution including cover traffic achieving the dual goal (i.e., recipient anonymity) of our proposal can be found in [40].

Among the approaches supporting cover traffic, the most significant are *mixnets*, originally proposed in [59], and *buses* [124, 21, 296].

In the literature, several proposals include cover traffic in mixnets [279, 98, 165, 158]. The introduction of cover traffic makes traffic analysis more difficult. For example, a possible approach is to introduce cover traffic to maintain a constant transmission rate. A very recent mixnet-based approach designed for the network layer was presented in [61]. However, it does not provide sender anonymity against a global adversary. Another relevant approach in this category, even if dated (but still very solid), is Tarzan [98]. As discussed in [37] and [42], mixnets, in general, require a suitable amount of cover traffic.

More related to our work are buses, as we also consider a pre-determined route that is used by the sender. However, buses are unrealistic in a large network (such as the Internet), since the fixed route is an Eulerian path passing through all the nodes, including thus all the possible pairs of senders–receivers.

Similar considerations can be made for DC-Nets [245], based on a secure multiparty cryptographic protocol, in which it is required that all participants are involved in every run of the protocol and initially share a pairwise key.

4

Providing Tor with recipient anonymity against a global adversary

Tor is a well-known routing protocol implementing the Onion multi-layered encryption to achieve communication anonymity. Among other possible attacks, Tor is vulnerable to passive attacks based on the compromise of multiple nodes, allowing the adversary to observe the traffic flow and then identify the relationship between sender and recipient (even if the latter is a hidden server). Relationship anonymity, in every threat model, can be reached by achieving at least one between sender and recipient anonymity. In this chapter, we pursue the dual goal faced in Chapter 3. Specifically, the idea here proposed is to work on the recipient-side, by considering that, due to the hidden-service mechanism, the recipient of the communication is actually a router playing as a rendezvous point with the hidden server. This allows us to obtain recipient anonymity also against traffic eavesdropping by enabling cover traffic and router collaboration. Some preliminary results of this approach are published in a research paper [40].

4.1 Introduction

As observed in Chapter 3, Tor offers neither sender anonymity nor recipient anonymity in a threat model in which the adversary can at least observe the traffic flow in the network. Indeed, the adversary can always detect the sender, by monitoring the traffic at the first relay node, and always observe the recipient, by monitoring the traffic outgoing from the last relay node. Relationship anonymity can be broken if traffic-analysis attacks are performed, allowing the adversary to relate sender and recipient communication activities. The introduction in Tor of the *rendezvous points* [114], where the communication between sender and *hidden server* (i.e., recipient) meets, just avoids the publicity of the identity of the services (that are then called *hidden*) but does not solve the above problem.

It is worth noting that real-life adversaries like those described above exist, because it just suffices to control a few Autonomous Systems.

The purpose of this chapter is to study how to extend Tor to achieve relationship anonymity also against global eavesdroppers. Dual to Chapter 3, in which we operated server-side, the idea we follow here is to operate recipient-side. This approach is based on the consideration that, due to the hidden-service mechanism, the recipient of the communication is actually a router playing as a rendezvous point with the hidden server. Therefore, we could enable both the inclusion of cover traffic and the collaboration among k Tor relays (natively prone to collaboration) with the purpose of hiding (also against global eavesdroppers) the actual recipient within an anonymity set of relays (thus obtaining k-anonymity [274]).

The chapter explores the above approach by proposing two protocols, called L-Tor and B-Tor. The former extends Tor by preserving the linear topology of the communication circuit. The latter enables a tree-like circuit.

The two protocols reflect the inherent trade-off underlying the approach between achieved anonymity degree and communication latency. We carefully study this trade-off both analytically and experimentally, reaching the conclusion that B-Tor actually solves the drawbacks of Tor while preserving low-latency applications.

4.2 Background and Notations

In this section, we recall some background notions and define some notations used throughout the chapter.

We report in the table of Figure 4.1 the list of symbols (and their description) used throughout the rest of the chapter.

4.2.1 Notations

We denote by $\langle SK, M \rangle$ the symmetric encryption with the key SK of the message M. We assume to use any secure probabilistic encryption scheme in such a way that, for an eavesdropper, two different encryptions of the same message are unlinkable to each other. We denote by $E(PK, M)$, the public-key encryption with key PK of a message M. $H(M)$ represents the application of a cryptographic hash function (e.g., SHA256) to a message M.

4.2.2 The Tor Network

A high-level discussion of the Tor protocol is provided in Section 3.2 of Chapter 3. However, to implement L-Tor and B-Tor, we require some changes in the steps applied in the standard Tor protocol. Moreover, we exploit the different types of cells present in Tor to implement some functionalities in our proposals. Finally, also

Symbol	Description
$\langle SK, M \rangle$	Symmetric encryption of M with the key SK
$E(PK, M)$	Public-key encryption of M with the key PK
$H(M)$	Application of a cryptographic hash function to a message M.
OR	Onion Router
OP	Onion Proxy
DS	Directory Server
DH	Diffie-Hellman
HS	Hidden Server/Hidden Service
RP	Rendezvous point
n	Total number of ORs for building circuits
L_T	Average lifetime of a circuit
p_b	Probability to break an OR
Adv	Generic adversary. It can be **W**, **S**, or **A**
W	Weak global Adversary
S	Static global Adversary
A	Adaptive global Adversary
δ_{Adv}	Time **Adv** spends to launch an attack to a node
Δ_{Adv}	Overall time available to **Adv** to compromise the network
n_{Adv}	number of attempts performed by **Adv** to compromise nodes
$S_{n_{Adv}}$	random variable counting the number of successes on n_{Adv} attempts.
AHS	Actual HS
ARP	Actual RP (It is the RP associated with HS)
k	Number of potential recipients selected in L-Tor and B-Tor to hide the AHS
$x_1, \ldots, x_k$	k RPs selected in L-Tor
x_i	ARP in L-Tor
BRP	Branch rendezvous points
CRP	Chain rendezvous points
r	Number of BRPs selected to build a circuit in B-Tor
l	Number of CRPs for each BRP selected to build a circuit in B-Tor
TCRP	Terminal CRP
$b_1, \ldots, b_r$	r BRPs selected in B-Tor
$x_{t,1}, \ldots, x_{t,l}$	l CRPs associted with the BRP b_t in B-Tor

Table 4.1: Notation.

the concept of *hidden services* is required. Therefore, in this section, we need a more detailed discussion of Tor with respect to that provided in Section 3.2 of Chapter 3.

The Tor network is an overlay network, based on TCP/TLS connections, consisting of multiple relay routers called *Onion routers* (OR). Each client runs locally an

Onion proxy (OP) which establishes a *virtual circuit (or path)* of ORs to communicate anonymously with the destination. To build a circuit, the OP periodically contacts a trusted server called *Directory Server* (DS) that keeps the information about the state of the network and provides the OP with *router descriptors* of the ORs. These router descriptors contain the IP addresses and the public keys of the ORs along with their network information, such as the bandwidth. Then, the OP selects, according to a *path selection algorithm*[248], a number t of OR relays forming the virtual circuit. By default, $t = 3$ and the first OR is called *entry router*, the second *middle router*, and the last *exit router*.

From now on, we interchangeably use the terms ORs, relays, and nodes. Furthermore, we use the terms sender, OP, and client to denote the same entity.

In Tor, the messages are exchanged in fixed-length cells of 512 bytes to make harder traffic analysis attacks [258]. There are two types of cells: *control* cells and *relay* cells that, in turn, are divided into further subtypes. We describe just those useful for our purpose.

We distinguish three phases in which the communication in the Tor network happens.

Set-up phase

Suppose the OP has selected three ORs. Then, it starts a set-up phase to build the virtual circuit. This phase is performed in such a way that each OR of the path only knows the previous and the next node of the path. Moreover, in this phase, the OP exchanges some messages with the ORs which include some Diffie-Hellman (DH) parameters to share a secret key. Since the OP has to be sure about the authenticity of the ORs, the DH parameters are encrypted by using the public keys of the ORs.

The messages exchanged during this phase are encapsulated into the control cells `CREATE` and `CREATED`, and in the relay cells `RELAY EXTEND` and `RELAY EXTENDED`.

Initially, the OP sends $E(PK_e, A_1)$ to the entry router, where PK_e is the public key of the latter and A_1 is the public DH parameter of the OP. This message is encapsulated into a cell `CREATE`. The entry router, through A_1, computes the DH shared key K_1 and replies to the OP by sending $(B_1, H(K_1))$, where B_1 is the public DH parameter of the entry router. This message is encapsulated into a cell `CREATED`. The OP computes K_1 and checks that the digest $H(K_1)$ corresponds to the value received from the entry router.

At this point, a secret key K_2 has to be exchanged between the OP and the middle router without the latter knowing the IP address of the former. To do this, the OP generates a new DH parameter A_2 and sends $\langle K_1, E(PK_m, A_2)\rangle$ to the entry router where PK_m is the public key of the middle router. This message is encapsulated

into a RELAY EXTEND cell so that the entry router understands to extend the circuit of one hop. Then, the entry router decrypts the message by using K_1 and forwards $E(PK_m, A_2)$ to the middle router encapsulated into a cell CREATE. The middle router, through A_2, computes the shared key K_2 and replies to the entry router with $B_2, H(K_2)$, where B_2 is the public DH parameter of the middle router. This message is encapsulated into a cell CREATED. Finally, the entry router forwards to the OP the message $\langle K_1, (B_2, H(K_2))\rangle$ encapsulated into a RELAY EXTENDED cell. The OP is now able to compute K_2 and check its integrity.

A similar procedure is performed to extend the path of one another hop until the exit router which, eventually, will share a key K_3 with the OP.

The details of the entire set-up phase are reported in the sequence diagram of Figure 4.1.

Forward phase

At the end of the set-up phase, the OP shares a secret key with each OR. These keys are used by the OP to encrypt (symmetrically) in *onion fashion* the messages intended for the destination. All these messages are encapsulated into the RELAY DATA cells. Specifically, to send a message M to the destination, the OP builds $\langle K_1, \langle K_2, \langle K_3, M\rangle\rangle\rangle$ and sends it to the entry router. The entry router removes a layer of encryption through K_1, obtaining $\langle K_2, \langle K_3, M\rangle\rangle$ that forwards to the middle router. The middle router removes the upper-most layer of encryption with K_2 obtaining $\langle K_3, M\rangle$, which forwards to the exit router. Finally, the exit router retrieves M through K_3 and forwards it to the destination.

Response phase

The response phase is performed in the reverse way of the forward phase so that each OR that receives a message adds a layer of encryption by using the symmetric key exchanged with the OP. Again, all the messages are encapsulated into the RELAY DATA cells. In detail, the destination sends the response R to the exit router. The exit router encrypts it with K_3 obtaining $\langle K_3, R\rangle$, which forwards to the middle router. The middle router adds a new layer of encryption, through K_2, obtaining $\langle K_2, \langle K_3, R\rangle\rangle$, which forwards to the entry router. Finally, the entry router forwards $\langle K_1, \langle K_2, \langle K_3, R\rangle\rangle\rangle$ to the OP, which removes all the layers of encryption and obtains R.

The above three described phases are reported in the sequence diagram of Figure 4.1.

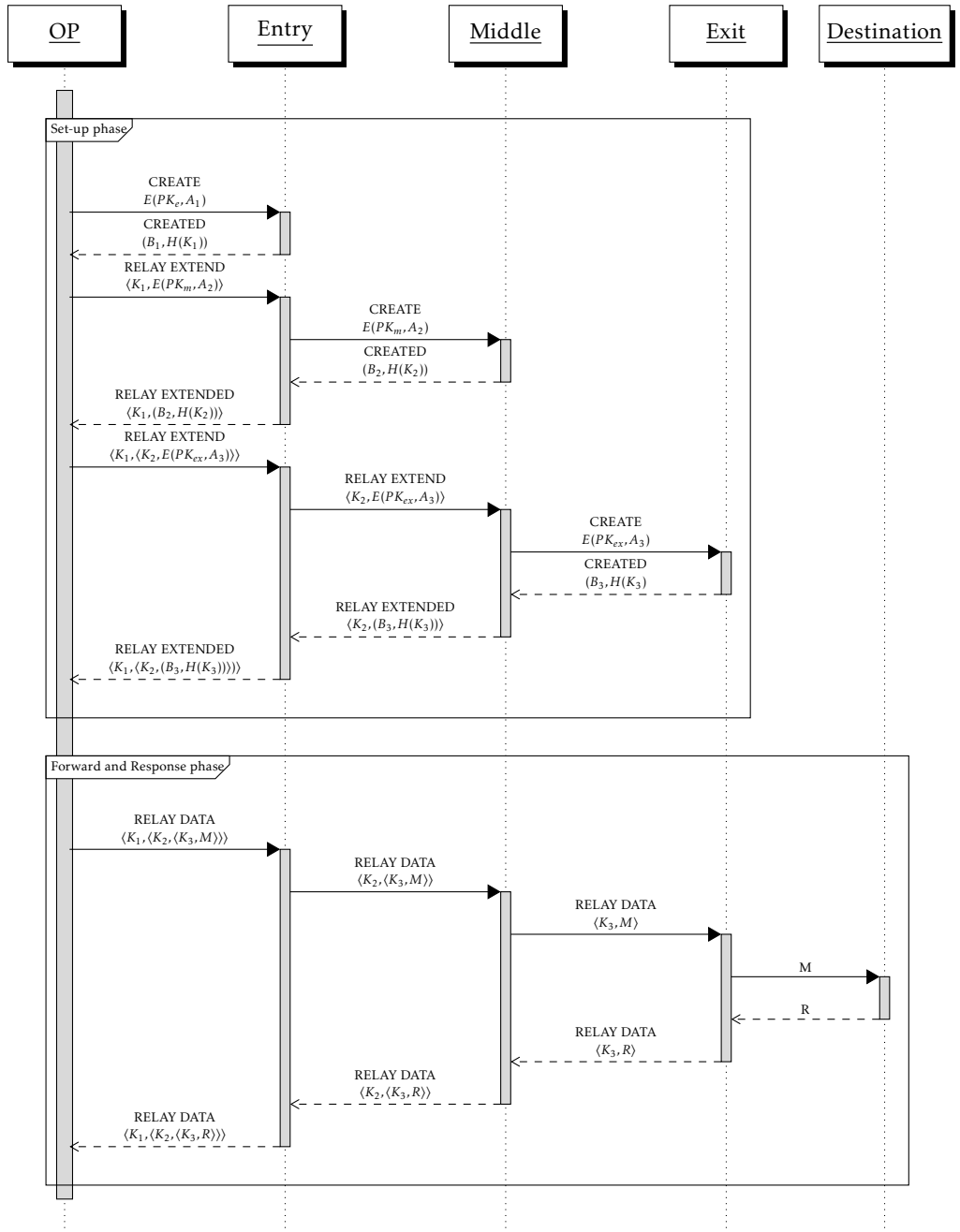


Fig. 4.1: Sequence diagram of the three phases of Tor.

4.2.3 Hidden services

The protocol so far discussed offers a certain level of anonymity to the sender. However, the IP address of the recipient is known to both the OP and the exit router. There are several scenarios in which the destination offers a particular service and does not want to be disclosed. Moreover, if the mapping between these services and the IP addresses of the servers hosting them is disclosed, they might be obscured by

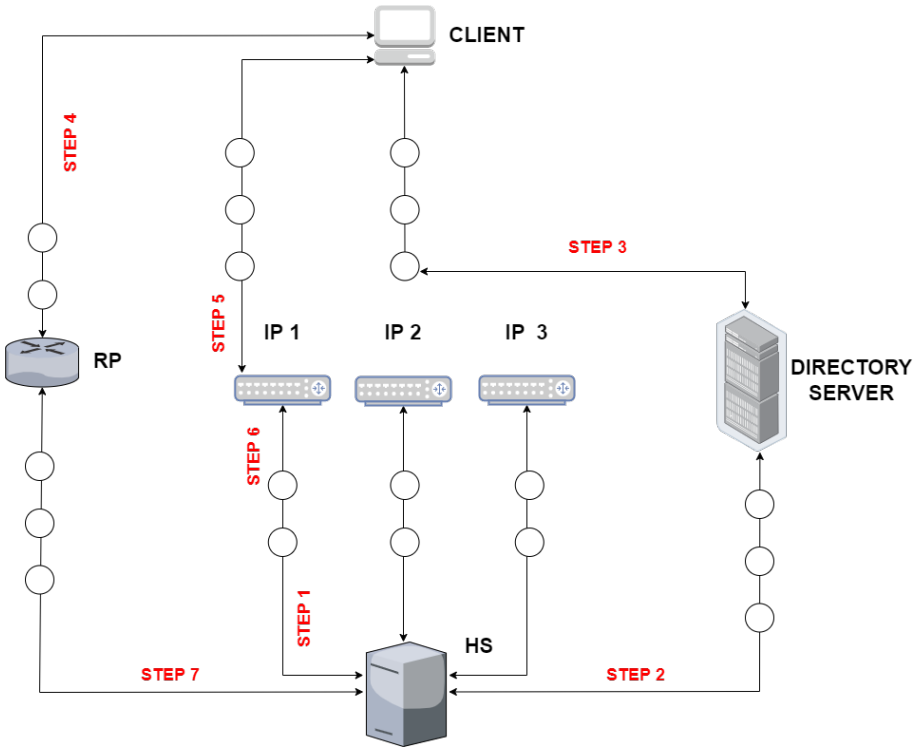


Fig. 4.2: Connection of the client towards a hidden server

an authority. Tor provides a mechanism to contrast the above problem based on the so-called *hidden servers* (and the corresponding *hidden services*).

We denote by HS the *hidden server*, and, for simplicity, we use the same symbol to denote the intended hidden service. We need a mechanism so that a client can reach the HS without knowing its IP address. In the description of this mechanism, we refer to the architecture and steps of Figure 4.2.

First, the HS builds three Tor circuits with three ORs called *introduction points* (Step 1). Each Tor circuit includes two ORs and the introduction point (three nodes in total). Once choosing the introduction points, the HS sends (through a standard Tor circuit composed of three ORs) a *service descriptor* including the addresses of the three introduction points and its public key to the directory server DS (Step 2). This descriptor is signed with the private key corresponding to the included public key. The *onion address* identifying HS is the encoding of a hash derived from its public key so that whoever receives the latter can verify its correctness.

At this point, a client that wants to connect with the HS on the basis of the knowledge of its onion address, contacts the DS (through a standard Tor circuit composed of three ORs) to obtain the service descriptor (Step 3). Then, the client selects an introduction point to connect with the HS. Before doing it, the client selects a Tor node called *rendezvous point* (RP) and builds a Tor circuit of two nodes (the RP is in

the third position). Then, the client provides the RP with a one-time secret string of 20 bytes, called *rendezvous cookie* (Step 4).

Moreover, the client builds a standard Tor circuit (including three ORs) with the selected introduction point and provides the address of the RP and the rendezvous cookie (Step 5). This information is forwarded from this introduction point to the HS (Step 6).

Finally, the HS builds a standard Tor circuit with the RP and provides it with the rendezvous cookie (Step 7). The RP matches the two rendezvous cookies provided by the client and the HS and establishes a circuit between them consisting of 6 ORs (including the RP). From now on, all the messages are exchanged through this circuit.

4.3 Threat Model

The goal of our proposals is to enhance the Tor protocol by providing recipient anonymity (and then relationship anonymity) against an external observer able to monitor the entire traffic exchanged in the network. We refer to this adversary as the *global adversary*.

Even though the hidden service mechanism included in Tor allows us to achieve a certain degree of recipient anonymity against some malicious nodes, Tor does not offer any protection for the recipient against the global adversary [209]. Indeed, it can simply identify the recipient of the communication by following the flow of messages originating from the sender and traveling through the six-hop circuit established through the rendezvous point. The final point of this circuit is the destination.

However, besides a passive eavesdropper, we are also interested in the security guarantees in terms of anonymity that our solution offers when some nodes (including the RPs) are *compromised*. We say that a node is compromised when the adversary obtains all the secret keys owned by the node and can access the content of the packets it receives.

We now introduce some notations. Our protocols extend Tor by proposing a different way to build circuits. We denote by n the total number of candidate ORs (currently, they are around 7,000). We denote by L_T the average lifetime of a circuit. Finally, we denote by p_b the probability that a node is broken when an attack is launched against it. For simplicity, we assume that p_b is the same for all the n nodes of the network.

Given an adversary **Adv**, we denote by δ_{Adv} the time the adversary **Adv** spends to launch an attack on a node (both in case of success and failure). Moreover, we

denote by Δ_{Adv} the overall time the adversary **Adv** has available to compromise the network.

Then, $n_{Adv} = \frac{\Delta_{Adv}}{\delta_{Adv}}$ represents the number of overall attempts done by the adversary. Clearly, among the n_{Adv} attempts, each attempt succeeds with probability p_b and fails with probability $1 - p_b$. Therefore, the probability to have exactly x_{Adv} successful attempts (i.e., the probability to compromise exactly x_{Adv} nodes) can be expressed according to the binomial distribution: $P(S_{n_{Adv}} = x_{Adv}) = \binom{n_{Adv}}{x_{Adv}} p_b^{x_{Adv}} (1 - p_b)^{n_{Adv} - x_{Adv}}$, with $x_{Adv} \leq n_{Adv}$ where $S_{n_{Adv}}$ is the random variable counting the number of successes on n_{Adv} attempts.

Our threat model is based on the threat model presented in [98]. It includes three types of adversaries with different capabilities.

W (*weak* global adversary): It monitors passively the entire traffic exchanged in the network and does not compromise any node. It is able to identify the nodes forming the circuit.

S (*static* global adversary): It is a weak global adversary which, in addition, is able to compromise nodes with the following features: $\Delta_S \gg L_T$ and $\delta_S \geq L_T$.

A (*adaptive* global adversary): It is a weak global adversary which, in addition, is able to compromise nodes with the following features: $\Delta_A \simeq L_T$ and a $\delta_A \leq L_T$

Observe that, according to the previous notations, n_S and n_A represent the overall attacks performed by **S** and **A**, respectively. Similar observations can be applied to S_{n_S}, x_S, S_{n_A}, and x_A.

W represents the standard *global passive adversary*. In real life, **W** may refer to the collaboration of several *honest-but-curious* ISPs (Internet Service Providers) that do not compromise the nodes but observe the traffic exchanged between them.

S refers to an adversary that compromises a priori a given number of nodes (that may be successively randomly selected by the sender to build a circuit) since it can freely operate (i.e., it is not constrained by anything but the overall time Δ_S). However, it is not very reactive with respect to the construction of the communication circuit, in the sense that each attack requires a time greater than the lifetime of the circuit itself (i.e., $\delta_S \geq L_T$). Therefore, the nodes to attack are randomly selected before the construction of the circuit. Concerning Δ_S, as stated in [98], it is reasonable to assume that it is bounded by the time period between two subsequent security assessments executed on each node. Indeed, the adversary can only control a compromised node for some period of time, until the compromise is detected and subsequently stopped.

A refers to an adversary with technical capabilities much higher than **S**, allowing it to perform on-the-fly attacks (i.e., $\delta_A \leq L_T$). However, it can operate only on

intended targets because of external constraints. This adversary represents the real-life case of a government agency with strong computational power (not available to any other attacker) authorized by the law or by the court to break the anonymity of a given communication. Therefore, **A** only attacks the nodes of the circuit. This means that Δ_A is basically the lifetime of the circuit.

4.4 Motivations and Introduction to our Approach

As discussed in the previous section, the standard Tor protocol does not offer recipient anonymity against the global adversary. Anyway, in several application scenarios (e.g, censorship resistance), this property should be guaranteed. Therefore, we search for an extension of Tor to achieve this goal.

Since, in the considered threat model, all the adversaries are at least global eavesdroppers, in Tor, following the flow of traffic is enough to identify the sender and the recipient of a communication.

It is well known in the literature [71] that any solution achieving such a goal requires the introduction of *cover traffic*, i.e., dummy traffic exchanged in the network with the aim to hide the real traffic.

However, for the Tor protocol, the introduction of cover traffic (like in the obfs4 protocol [291]) is not sufficient, since the topology itself (i.e., the way in which the nodes are organized) reveals some information to the global adversary. Specifically, the sender is always the first node and the recipient is always the last node, even when the hidden services mechanism is enabled.

Therefore, a solution protecting the recipient should include cover traffic, but should also introduce a degree of uncertainty provided by the topology itself, in such a way that the recipient should be hidden among other possible recipients. A similar concept is present in mixnets [98], from the side of sender anonymity, in which the actual sender is hidden within a given anonymity set of senders.

The approach we plan to follow in this chapter is then to obtain recipient anonymity by arranging a Tor circuit identifying not just a single recipient but a set of potential HSs (we assume that a recipient is always an HS) among which only one is the actual recipient.

There are several ways to do this. Suppose that a given client wants to anonymously communicate with a certain HS, which we call *actual* HS (AHS). A trivial approach is that the sender chooses k HSs including the AHS and builds a standard (i.e., six-hop) circuit with each HS. If dummy traffic is enabled in each circuit (represented by fake requests/responses forwarded to/from the HSs different from the AHS), we obtain anonymity of the recipient with uncertainty $\frac{1}{k}$. Anyway, this option

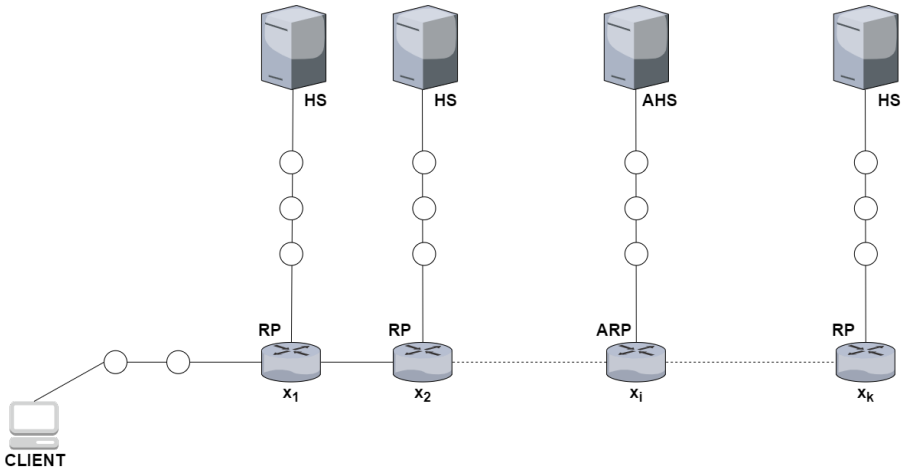


Fig. 4.3: Circuit of L-Tor

is not realistic because it requires huge bandwidth waste client-side, by considering that the client has to generate, for each actual request, k requests and receives k responses.

A more sophisticated idea is to keep just one Tor circuit but longer than the standard one, in such a way that the rendezvous point associated with the AHS (called ARP) is hidden within the route, as sketched in Figure 4.3.

Therein, we have a long Tor circuit of RPs and each HS is connected to an RP. We design this solution in this chapter (in Section 4.5), by calling it L-Tor. As intuitive, L-Tor introduces a price. Indeed, as analyzed in Section 4.8, the set-up latency increases quadratically with the number k of HSs, while communication latency grows linearly with k (both in the forward and response phases). Then, to achieve a good level of anonymity, we require excessive latency times. This aspect is also analyzed experimentally in Section 4.9. Therefore, L-Tor can be acceptable only for applications that do not require low latency. On the other hand, as we will see in Section 4.7, L-Tor achieves a high level of security.

After this, the problem addressed in the chapter is how to arrange a Tor-like circuit to achieve recipient anonymity without paying the high price in terms of latency required by L-Tor. To do this, we can also tolerate reducing the achieved security goal, still maintaining a good advantage with respect to standard Tor. This leads to the definition of B-Tor (see Section 4.6), in which the anonymity set of recipients is obtained by adopting a tree-like topology of the circuit. Sections 4.7 and 4.9 will show that B-Tor offers a good solution to the above trade-off.

4.5 L-Tor: A first extension of the Tor Protocol

Through this section, we describe the L-Tor protocol introduced in the previous section.

4.5.1 Overview of L-Tor

To illustrate L-Tor, we refer to the steps of Section 4.2.3 and highlight the changes introduced by L-Tor.

Steps 1 and 2 are essentially the same, i.e., all the HSs of the network select three introduction points and advertise their service descriptor to the DS.

In Step 3, the client (through a Tor circuit) asks for k HSs (in place of a single HS). The actual HS is one of them.

Step 4 is the core of L-Tor, in which the client selects k RPs (one for each HS) and builds a circuit with them to exchange the rendezvous cookies. As explained above, to avoid bandwidth overhead, we want to avoid the client sets k independent circuits with each RP. We will return later on this step, anyway, eventually, each RP is connected with the client and obtains a rendezvous cookie to match with that provided by the corresponding HS.

Steps 5–7 proceed as in Section 4.2.3, i.e., the client (through an introduction point) provides each HS with the address of the corresponding RP. Each HS connects with an RP and the circuit with the client is established.

Regarding Step 4, the idea is to build a Tor circuit of k RPs in which the client sends a single request. Each RP forwards this request to the corresponding HS, which replies with a response. Among these responses, only one is the actual response. However, the circuit of RPs is crossed just by the actual response hidden within the minimal required cover traffic, so that the client does not have to handle the traffic multiplication of the trivial solution (simply requiring that the client establishes k standard Tor circuits).

In the next sections, we describe the above protocol in detail.

4.5.2 Set-up phase

In the set-up phase, a circuit between the client and k collaborating HSs (including the AHS) is built.

As in Steps 1 and 2 of Section 4.2.3, each HS (including the AHS) of the system selects the introduction points and advertises its service descriptor to the DS. This is done, as usual, through standard Tor circuits.

At this point, the client selects k HSs among which the AHS is included and (through a Tor circuit) asks the DS for the service descriptors of the k HSs. This

corresponds to Step 3 of Section 4.2.3. Clearly, these k HSs have to be selected in such a way that they appear equally likely to be selected from the client. To avoid intersection attacks [73], the construction of the anonymity set of recipients happens as follows. When the client wants to contact a given HS (i.e., the AHS) for the first time, it selects randomly other $k-1$ HSs and performs the protocol as described in the following. From now on, a one-to-one mapping between this anonymity set and each HS occurring in it is established. This means that, for successive communications, if the client wants to contact any HS belonging to this set, it uses the same anonymity set. Obviously, for each fresh HS, a new anonymity set is built.

In Step 4, the client selects k RPs (one for each HS) and will exchange the rendezvous cookies with them. Furthermore, these k RPs will be arranged in a Tor circuit. Therefore, they can be selected according to the path selection algorithm of Tor (possibly with some modifications taking into account the fact that k could be greater than the standard 3 relays selected in Tor).

We denote by $x_1, x_2, \ldots, x_k$ the RPs in the order they appear in the Tor circuit.

The client randomly associates each HS with each RP. We denote by x_i $(1 \leq i \leq k)$ the RP associated with the AHS and say that x_i is the *actual* RP (ARP).

At this point, the client builds a standard Tor circuit including the RPs from x_1 to x_i. The only difference with a standard construction is that, after extending the circuit of one hop by including a new RP, the client provides the latter with the rendezvous cookie. Moreover, along with the cookie, the client communicates to x_i a flag denoting that it is associated with the AHS (i.e., that it is the ARP). Clearly, x_i does not know the address of the AHS as in the standard approach. Thus, unless x_i is compromised by a global adversary, the recipient anonymity is preserved. In the end, the client shares a secret key with all the RPs from x_1 to x_i and provides them with the proper rendezvous cookies.

Now, if $i = k$, then Step 4 ends, otherwise the circuit has to be further extended until x_k. This operation is performed by x_i playing the role of client. However, it should be done in such a way that neither x_{i+1} (and the successive RPs) nor the global adversary can identify x_i as the ARP. To achieve this, the pattern followed (also in terms of inter-step times) is the same as before. Specifically, the client sends a `RELAY EXTEND` cell to x_i, including the address of x_{i+1}. However, the encrypted DH parameter of the sender for x_{i+1} is replaced with dummy bytes. Indeed, to extend the circuit, x_i sends the standard `CREATE` cell to x_{i+1}, in which the encrypted public DH parameter is set by x_i itself in place of the client. When x_i receives the `CREATED` cell from x_{i+1}, x_i can generate a secret key shared with x_{i+1}. Moreover, x_i forwards the `RELAY EXTENDED` cell to the client as the standard Tor set-up phase. This way, this extension is, from the point of view of both x_{i+1} and the global adversary, the same

as the previous extensions until x_i. Now, the client sends the rendezvous cookie to x_i that forwards it to x_{i+1}. This procedure is iterated until the RP x_k.

In the end, we have that the client shares a symmetric key with each RP x_j where $1 \leq j \leq i$ and x_i shares a symmetric key with each RP x_t where $i+1 \leq t \leq k$. Moreover, each RP from x_1 to x_k knows the proper rendezvous cookie.

For completeness, to protect the client against a malicious RP, the connection between the client and x_1 can be done through a Tor circuit of two hops, as in the standard Step 4 of Section 4.2.3.

To conclude the set-up phase, Steps 5–7 of Section 4.2.3 are replicated in L-Tor. Then, for each HS, the client contacts (in parallel) an introduction point and provides the proper address of the RP and the corresponding rendezvous cookie (Step 5). The introduction point forwards this information to the HS (Step 6). Finally, each HS connects with the proper RP by providing the rendezvous cookie so that the circuit with the client can be established (Step 7).

A representation of the final circuit is depicted in Figure 4.3.

4.5.3 Forward phase

After setting the circuit, the client can send a request to the AHS through the RELAY DATA cells. This request is encrypted in Onion fashion as described in Section 4.2.2 until x_i.

Each RP x_j with $1 \leq j \leq i-1$, when receiving a RELAY DATA cell, removes its layer of encryption with the secret key shared with the client and forwards the obtained message to x_{j+1}. Moreover, x_j also forwards the message to the associated HS (through the Tor circuit). Observe that the HS is not able to decrypt the received message and detects that this message represents a request not really intended for it.

Similarly, x_i removes its layer of encryption and obtains the actual request to forward to the AHS (clearly, it is encrypted with the public key of the AHS so that x_i cannot access its content). After forwarding it to the AHS, x_i builds a dummy message of the same size as the request and forwards it, layered encrypted until x_k. Observe that, x_i can encrypt this message since it has exchanged a secret key with each RP x_t, where $i+1 \leq t \leq k$.

Each x_t, when receiving a RELAY DATA cell, performs as the RPs between x_1 and x_i, i.e., removes its layer of encryption, forwards the message to the associated HS, and forwards the message to the next RP of the path (x_k just forwards the message to the associated HS since it is the last RP).

As a final observation, during this phase, each HS receives the same quantity of data and each RP behaves in the same way so that the global adversary cannot identify the position of the ARP in the path, and then the recipient of the request.

4.5.4 Response phase

When an HS receives a request, even a dummy, it replies to the associated RP by generating a burst of cells including cover traffic. Among these dummy cells, the AHS will inject the actual response. This way, the global adversary, monitoring the activity of all the HSs, cannot identify the AHS.

To avoid all the responses reaching the client, resulting in bandwidth waste, each RP does not forward to the client all the cells received. Indeed, each RP knows if it is in charge to forward the request/response to/from the AHS, i.e., each RP knows if it is the ARP. Then, all the RPs, except for the ARP, can discard the traffic received from the associated HS. However, a certain degree of cover traffic has to be exchanged to hide the ARP from the global adversary. Then, we enable a cover-traffic burst mechanism.

Specifically, x_k (i.e., the last RP) generates a burst of `RELAY DATA` cells containing cover traffic. These dummy cells, encrypted in Onion fashion as in the response phase of Section 4.2.2, travel the path of RPs until the client and are exploited by x_i to inject the actual response. In particular, each RP in the backward path from x_{k-1} to x_{i+1} receives a `RELAY DATA` cell and adds the proper layer of encryption (since it does not know if ARP is back or ahead). When x_i receives a dummy cell, since it knows to be the ARP, it replaces such a cell with a cell containing (part of) the actual response. Once all the cells containing the response are sent, the other cells are forwarded with dummy traffic.

Similarly, each RP from x_{i-1} to x_1 receives a `RELAY DATA` cell and adds a layer of encryption. Finally, the client can retrieve the response by removing the layers of encryption with the secret keys shared with the RPs $x_1, \ldots, x_i$.

This concludes the description of L-Tor.

In Figure 4.4, we represent the paths followed by the messages in the L-Tor circuit. Specifically, the red arrows represent the path of the actual messages generated by the client during the forward phase. The blue arrows represent the path of the actual response generated by AHS during the response phase. The black and grey arrows represent dummy traffic originated during the forward and response phases, respectively.

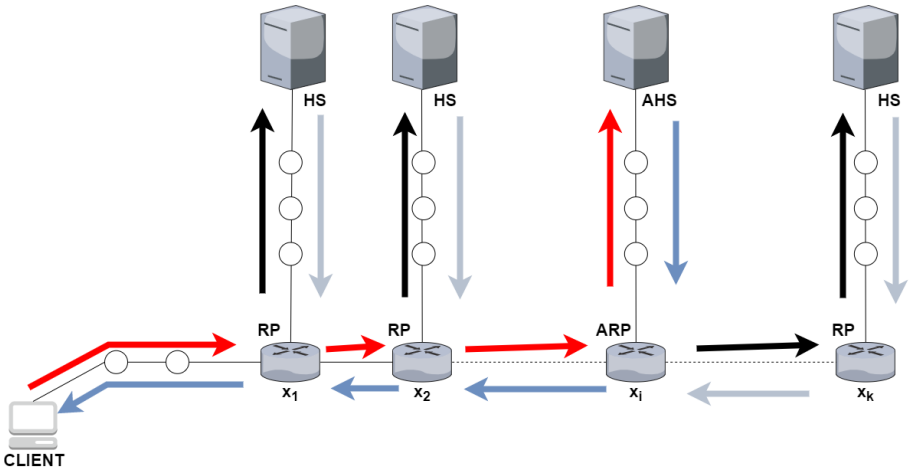


Fig. 4.4: Paths followed by the messages in the L-Tor circuit.

.

4.6 The Branched-Tor Protocol (B-Tor)

Even though L-Tor can achieve the security goal we pursue (as well shown in Section 4.7, it is not scalable and its performance could not be acceptable in several real-life applications (like web browsing). Indeed, to obtain a good anonymity level (e.g., $k = 50$), the latency required to set a circuit of k relays and to exchange messages through them could result prohibitive (see Sections 4.8 and 4.9). Therefore, we provide a new solution, called B-Tor, which, at a minimum price in terms of anonymity with respect to L-Tor, offers much better performance, making our approach applicable to low-latency applications.

4.6.1 Overview of B-Tor

As in L-Tor, also in B-Tor we consider a network where the AHS is hidden among a set of k possible HSs, each connected to an RP.

The main difference between L-Tor and B-Tor is the way in which the RPs are arranged. Indeed, we move from the linear topology of Figure 4.3 to a tree-like topology as depicted in Figure 4.5.

Specifically, in Step 4, the client builds a Tor path of RPs with just r *branch* rendezvous points (BRPs) among the k RPs.

Each of these r BRPs establishes (in parallel) a Tor circuit with l *chain* rendezvous points (CRPs) (chosen by the sender), where $l = \frac{k-r}{r}$, with the assumption that $(k-r)$ **mod** $r = 0$. The last CRP of each chain is called *terminal* CRP (TCRP).

Thus, we obtain a tree-like topology as represented in Figure 4.5, which involves all the k RPs. The ARP (i.e., the RP connected with the AHS) is any of them (i.e., it is either a BRP or a CRP).

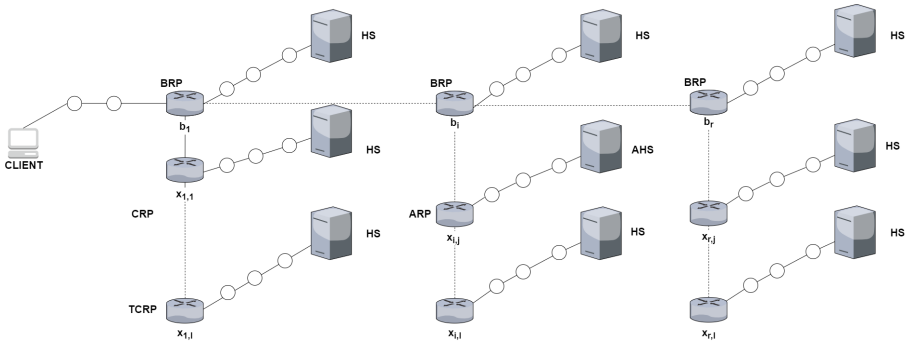


Fig. 4.5: Circuit of B-Tor

Anonymity is obtained by applying a broadcast-based technique for the forward phase, and cover traffic (generated by the TCRPs) for the response phase.

In the next section, we describe in detail B-Tor.

4.6.2 Set-up phase

As for L-Tor, we refer to the steps of Section 4.2.3.

Steps 1–3 are exactly the same as L-Tor, i.e., each HS selects the three introduction points and advertises its service descriptor to the DS. The client asks for k HSs including the AHS.

In Step 4, the client selects k RPs to associate with the HSs and arrange them in a circuit. However, this time, the circuit is different from that of L-Tor. Indeed, among the k RPs, the client selects r BRPs and builds a Tor circuit with them. We denote by $b_1,\ldots,b_r$ the BRPs in the order they appear in the circuit. Moreover, for each BRP b_t, the client selects a list $x_{t,1},\ldots,x_{t,l}$ of CRPs with which b_t will establish a Tor circuit (in the order they appear in the list). We recall that $l = \frac{k-r}{r}$. The CRPs $x_{t,l}$, for each $1 \leq t \leq r$, are called TCRPs.

B-Tor performs differently according to the position in the circuit of the ARP. We have to consider two cases. The first case is that the ARP is a BRP (Case 1). The second case is that the ARP is a CRP (Case 2).

Case 1. Suppose the ARP is the BRP b_i with $1 \leq i \leq r$. In this case, as in L-Tor, the client builds a Tor circuit of BRPs from b_1 to b_i. Moreover, again as in L-Tor, each time the circuit is extended by one hop, the client provides the new BRP with the rendezvous cookie to match with that provided by the HS associated with the new BRP. However, in B-Tor, along with the rendezvous cookie, the client sends each b_j (with $1 \leq j \leq i$) the list of the addresses of the CRPs $x_{j,1},\ldots,x_{j,l}$ and the associated list of rendezvous cookies for each of them. This information will be used by b_j to build a circuit including the CRPs $x_{j,1},\ldots,x_{j,l}$. A flag for b_i denoting that it is the ARP is also provided. Observe that, since the rendezvous cookies have a size of 20 bytes and

the addresses of 4/16 bytes (IPv4/IPv6) until $l = 20$, cookies and addresses can be carried out in a single Tor cell.

At this point, the circuit of BRPs has to be extended as in L-Tor by b_i (playing the role of client) until b_r. Specifically, the client simulates the extension of the circuit by sending the RELAY EXTEND cell to b_i, which includes the address of b_{i+1}. However, the CREATE cell generated by b_i for b_{i+1} includes the DH parameter of b_i itself. After receiving the CREATED cell from b_{i+1}, b_i forwards the RELAY EXTENDED cell to the client, so that neither b_{i+1} nor the global adversary can identify b_i as the ARP. At this point, the client provides b_i with the rendezvous cookie for b_{i+1}, the list of the addresses of the CRPs associated with b_{i+1}, and the list of the associated rendezvous cookies. This process is repeated until b_r is included in the Tor circuit. Each b_j (with $1 \leq j \leq r$), after receiving the list $x_{j,1},\ldots,x_{j,l}$, builds a Tor circuit until $x_{j,l}$ by providing each $x_{j,t}$ (with $1 \leq t \leq l$) with the proper rendezvous cookie. This is the main advantage of the set-up phase of B-Tor with respect to L-Tor. Indeed, the circuits started by the BRPs can be done in parallel between them. This reduces the latency required to set the circuit.

To summarize, in the end, the client shares a secret key with each BRPs $b_1,\ldots,b_i$ and b_i shares a secret key with each BRPs $b_{i+1},\ldots,b_r$. Moreover, each BRP b_j (with $1 \leq j \leq r$) shares a secret key with each CRPs $x_{j,1},\ldots,x_{j,l}$. Finally, each RP (both the BRPs and the CRPs) owns the proper rendezvous cookie generated by the client.

Case 2. Suppose the ARP is the CRP $x_{i,j}$ with $1 \leq i \leq r$ and $1 \leq j \leq l$. As in Case 1, the client builds a circuit of BRPs until b_i and provides each BRP b_t ($1 \leq t \leq i$) with the proper rendezvous cookie, the list of CRPs $x_{t,1},\ldots,x_{t,l}$, and the list of the rendezvous cookies (one for each CRP). However, differently from Case 1, additional information is provided to b_i that is the fact that $x_{i,j}$ is the ARP.

At this point, the circuit is extended by b_i until b_r exactly as in Case 1. Moreover, again as in Case 1, each b_z ($1 \leq z \leq r$ and $z \neq i$) builds a Tor circuit and exchanges the symmetric keys and the rendezvous cookies provided by the client with the CRPs $x_{z,1},\ldots,x_{z,l}$.

Regarding b_i, it builds a circuit only until $x_{i,j}$ (instead of x_{i_l}) and the circuit is extended by $x_{i,j}$ (playing the role of b_i) until $x_{i,l}$.

To summarize, in the end, the client shares a secret key with each BRPs $b_1,\ldots,b_i$, b_i shares a secret key with each BRPs $b_{i+1},\ldots,b_r$, each BRP b_t (where $1 \leq t \leq r$ and $t \neq i$) shares a secret key with each CRPs $x_{t,1},\ldots,x_{t,l}$, b_i shares a secret key with each CRPs $x_{i,1},\ldots,x_{i,j}$, and $x_{i,j}$ shares a secret key with each CRP $x_{i,j+1},\ldots,x_{i,l}$. Finally, each RP (both the BRPs and the CRPs) owns the rendezvous cookie.

To conclude the set-up phase, Steps 5–7 are performed in the same way as L-Tor (see Section 4.5.2).

As in L-Tor, the client connects to the first BRP b_1 through a two-hops circuit.

4.6.3 Forward Phase

As in the setup phase, also in the forward phase, we have to distinguish two cases according to the position of the AHS.

Case 1. Suppose the ARP is the BRP b_i with $1 \leq i \leq r$. The client sends the request in the `RELAY DATA` cells through the Tor circuit of BRPs from b_1 until b_i. This request is encrypted, as usual, in Onion fashion by the client through the keys shared with $b_1, \ldots, b_i$.

Each b_j (with $1 \leq j \leq i-1$) that receives a `RELAY DATA` cell, removes its layer of encryption and forwards the obtained message to b_{j+1} and to the associated HS (as in the forward phase of L-Tor). Moreover, b_j encrypts, in Onion fashion, a dummy message and forwards it through the Tor circuit of CRPs associated with it. This is done through the keys that b_j shares with $x_{j,1}, \ldots, x_{j,l}$.

When b_i receives the request, b_i forwards it to the AHS and a dummy message in the Tor circuit composed of $x_{j,1}, \ldots, x_{j,l}$. Moreover, through the keys shared with $b_{i+1}, \ldots b_r$, forwards a dummy message of the same size as the request (as in L-Tor) encrypted in Onion fashion until b_r.

Each b_t (with $i+1 \leq t \leq r$) performs as the other BRPs between b_1 and b_i, i.e., removes its layer of encryption, forwards the received messages: (1) to the associated HS, (2) in the associated Tor circuit of CRPs $x_{t,1} \ldots x_{t,l}$, and (3) to the next BRP of the path (b_r, being the last RP, just forwards the messages to the associated HS and in the associated Tor circuit of CRPs).

Finally, each $x_{p,q}$ (where $1 \leq p \leq r$ and $1 \leq q \leq l$) performs in a similar way. Specifically, it removes its layer of encryption, forwards the messages to both the associated HS and the next CRP $x_{p,q+1}$ of the path ($x_{p,l}$, being the last CRP, just forwards the message to the associated HS).

Case 2. Suppose the ARP is the CRP $x_{i,j}$ ($1 \leq i \leq r$ and $1 \leq j \leq l$).

The client performs as in Case (1), i.e., it builds an Onion encrypted request that crosses the path from b_1 until b_i. Each b_t (with $1 \leq t \leq i-1$) performs exactly as in Case (1).

Regarding b_i, as in Case (1), it forwards the request to the associated HS and a dummy message until b_r. Moreover, it forwards the request into the associated Tor circuit of CRPs until $x_{i,j}$ (that is the ARP). Specifically, b_i encrypts in Onion fashion the request by using the keys shared with $x_{i,1}, \ldots, x_{i,j}$. This is different from Case 1, in which a dummy message (in place of the actual request) is sent through the Tor circuit until the CRP $x_{i,l}$ (in place of $x_{i,j}$).

The BRPs from b_{i+1} until b_r perform exactly as Case 1. Similarly, each $x_{p,q}$ with $(p,q) \neq (i,j)$ performs as Case 1.

Finally, $x_{i,j}$ after receiving the request, forwards it to the AHS and builds a dummy message encrypted in Onion fashion to forward until the CRP $x_{i,l}$. This message is encrypted through the keys shared by $x_{i,j}$ with $x_{i,j+1} \dots x_{i,l}$.

4.6.4 Response Phase

To avoid the global adversary can see the node which originates the response, a cover traffic mechanism is enabled.

Exactly as in L-Tor, when an HS receives a request, even a dummy, it replies to the associated RP by generating a burst of cells including cover traffic, and all the RPs except for the ARP can discard the traffic received.

The ARP will inject the response in a cover traffic burst managed as follows.

Each TCRP $x_{i,l}$ (with $1 \leq i \leq r$), after receiving the response from the associated HS, generates a burst of RELAY DATA cells containing cover traffic. This burst will travel in the Tor circuit of CRPs from $x_{i,l}$ until b_i. Then, another burst of RELAY DATA cells flows in the Tor circuit of BRPs from b_r until the client in the backward direction. Observe that, each BRP b_j ($1 \leq i \leq r-1$) receives two flows: one from $x_{j,1}$ and one from b_{j+1} (while b_r receives just a flow from $x_{r,1}$).

We recall that b_j knows if the flow coming from $x_{j,1}$ contains or not the actual response, since it knows if the ARP is one between $x_{j,1}, \dots, x_{j,l}$ (or it is b_j itself). Therefore, b_j can ignore one of the two flows. Specifically, if the ARP is one between $x_{j,1}, \dots, x_{j,l}$, b_j can discard the flow coming from b_{j+1}. Otherwise, it discards the flow coming from $x_{j,1}$. Observe that in the latter case, b_j does not know whether the flow coming from b_{j+1} contains the actual response, since it could come from a BRP b_t with $t < j$ or from a CRP $x_{p,q}$ with $q < j$. Then, b_j simply forwards the traffic received from b_{j+1}. In the case in which b_j is the ARP, it can ignore both flows.

Since the two traffic flow coming from $x_{j,1}$ and b_{j+1} may have different data rates, say R_1 and R_2 respectively, to avoid the global adversary understand which of the two flows contains the response, b_i chooses as data rate for the traffic intended for b_{i-1}, the maximum between R_1 and R_2. This way, not all the cover traffic received by b_i has to be forwarded toward the client, thus saving bandwidth.

At this point, we see in detail as the RPs perform in the response phase and as the ARP injects the response.

We distinguish, as usual, two cases.

Case 1. Suppose the ARP is the BRP b_i with $1 \leq i \leq r$.

Each CRP $x_{p,l}$ ($1 \leq p \leq r$) generates a cover traffic burst of RELAY DATA cells including dummy data and adds its layer of encryption. Each CRP $x_{p,q}$ ($1 \leq p \leq r$ and

$2 \leq q \leq l-1$), receives a RELAY DATA cell and adds its layer of encryption as in the response phase of Tor (see Section 4.2.2). $x_{p,1}$ adds its layer and forwards the cells to b_p.

Each b_p ($1 \leq p \leq r$), since it knows that the ARP is not in its chain, simply ignores the cells received from $x_{p,1}$.

b_r forwards a flow of cells towards b_{r-1} by adding its layer of encryption. This flow is forwarded at the same data rate as the flow received from $x_{r,1}$. If $r = i$, the flow forwarded by b_r to b_{r-1} contains the actual response, otherwise it contains dummy traffic.

Each b_j (with $i+1 \leq j \leq r-1$), receives two flows, one from b_{i+1} and one from $x_{j,1}$, and forwards to b_{j-1} the cell coming from b_{i+1} at the maximum data rate between the two flows (possibly, by adding cover traffic cells). The cells from $x_{j,1}$ are ignored since b_j is aware of the fact that they do not contain the actual response. Moreover, before forwarding, it adds a layer of encryption as the standard Tor response.

b_i (if $i \neq r$) ignores both the flows received from b_{i+1} and $x_{i,1}$. Then, it forwards (after adding its layer of encryption) the actual response to b_{i-1} (or to the client if $i = 1$), at the maximum rate between the two flows. When the response is completely forwarded, dummy cells are generated to maintain the same data rate.

Finally, each BRP b_t ($1 \leq t \leq i-1$) performs as each BRP b_j ($i+1 \leq j \leq r-1$). Specifically, b_t receives two flows, ignores that provided by $x_{t,1}$, and forwards the flow coming from b_{i+1} to b_{t-1} (or to the client if $t = 1$) after adding its layer of encryption. This flow is sent at the maximum data rate between the two flows.

Case 2. Suppose the ARP is the CRP $x_{i,j}$ ($1 \leq i \leq r$ and $1 \leq j \leq r$).

All the RPs except for $x_{i,j}$ and b_i perform exactly as Case 1.

Regarding $x_{i,j}$, it simply replaces the RELAY cells (containing cover traffic) coming from $x_{i,j+1}$, with the actual response (or injects directly the actual response in the case $j = l$, i.e., the ARP is a TCRP).

Regarding b_i, it does not ignore the flow coming from $x_{i,1}$, but removes the layer of encryption added by the CRPs from $x_{i,1}$ until $x_{i,j}$ (through the secret key shared with them) and forwards this flow to b_{i-1} (or to the client if $i = 1$).

In Figure 4.6, we represent the paths followed by the messages in the B-Tor circuit. Specifically, the red arrows represent the path of the actual messages generated by the client during the forward phase. The blue arrows represent the path of the actual response generated by AHS during the response phase. The black and grey arrows represent dummy traffic originated during the forward and response phases, respectively.

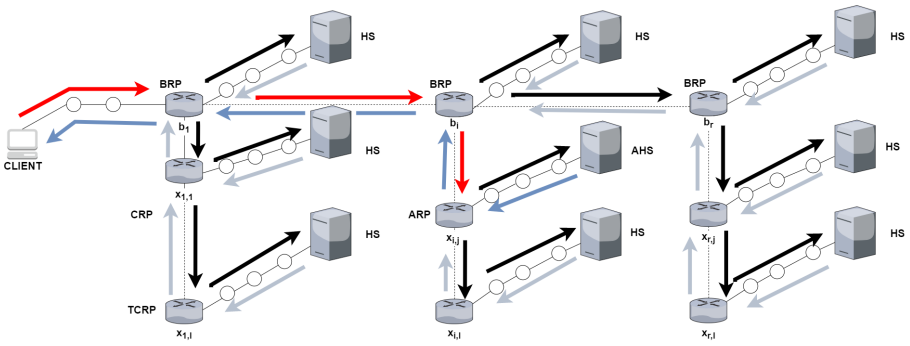


Fig. 4.6: Paths followed by the messages in the B-Tor circuit.

.

4.7 Security Analysis

In this section, we provide the security analysis of the proposed solution according to the threat model described in Section 4.3.

In particular, we compare, from the point of view of the offered security guarantees, the standard Tor protocol (which does not provide any level of anonymity in our threat model), L-Tor (which offers the maximum level of anonymity but low performance), and B-Tor (offering better performance than L-Tor with a price in terms of anonymity).

Notations. We adopt the notation of Section 4.3. Moreover, we assume k is the number of potential recipients (and then of RPs) selected by the client in L-Tor and B-Tor. Finally, we denote by $P(E)$ the probability function of an event E.

Adversaries. We consider three types of global adversaries: Weak (**W**), Static (**S**), and Adaptive (**A**) as defined in Section 4.3.

We refer to the following assumption.

Assumption.

A1: The k HSs selected in L-Tor and B-Tor are all potential recipients and they are chosen by the client in such a way that the background knowledge does not allow the adversary to have more information than recipient uniform distribution.

The satisfaction of **A1** is an aspect that is strongly application-dependent, so we cannot treat it in depth. However, this is a classical problem in communication anonymity [94, 246].

Finally, we analyze two security properties.

Security properties.

- **RPA:** Recipient anonymity.
- **RLA:** Relationship anonymity.

	RPA/RLA		
Protocol	**W**	**S**	**A**
Tor	1	1	1
L-Tor	$\frac{1}{k}$	$\leq \frac{n_S p_b}{n} + \frac{1}{k}$	$\frac{n_A p_b + 1}{k}$
B-Tor	$\frac{1}{k}$	$\leq \alpha$	$\leq \frac{1}{k} + \frac{n_A p_b}{r}$

Table 4.2: Comparison between Tor, L-Tor, and B-Tor. The table reports the probability for the adversary (**W**, **S**, and **A**) to break **RPA** and **RLA**.

Since we deal with an adversary able to observe the entire traffic exchanged in the network, our analysis is conducted in terms of anonymity set size.

Specifically, we calculate the probability that each considered adversary (among **W**, **S**, **A**) breaks the two considered security properties (**RPA, RLA**) in the three protocols (Tor, L-Tor, B-Tor).

The formal framework proving this result is provided through the following theorems and summarized in Table 4.2.

We start by considering **RPA** in Tor, L-Tor, and B-Tor against **W**, **S**, and **A**.

Given an adversary **Adv**, we introduce two events: R_{Adv} ={**Adv** breaks **RPA**} and $X_{Adv,x_{Adv}}$ = {**Adv** compromises x_{Adv} nodes}.

Then, since n_{Adv} is the maximum number of nodes that **Adv** can compromise we obtain:

$$P(R_{Adv}) = \sum_{x_{Adv}=0}^{n_{Adv}} P(R_{Adv}|X_{Adv,x_{Adv}})P(X_{Adv,x_{Adv}}).$$

Observe that $P(X_{Adv,x_{Adv}}) = P(S_{n_{Adv}} = x_{Adv})$ as defined in Section 4.3. Therefore, to find $P(R_{Adv})$, we need to compute $P(R_{Adv}|X_{Adv,x_{Adv}})$ for each x_{Adv}.

Moreover, for **W**, we have a simplification. Indeed, since **W** does not compromise any node ($n_W = 0$), then $P(R_W) = P(R_W|X_{W,0})P(X_{W,0}) = P(R_W|X_{W,0})P(S_0 = 0) = P(R_W|X_{W,0})$.

Theorem 4.1. *In* Tor, ***W***, ***S***, *and* ***A*** *break* ***RPA*** *with probability* 1.

Proof.

Adversary W

In Tor, **W** simply follows the flow of messages generated by the client through the six-hop circuit until the recipient (hidden server). Therefore, the recipient is immediately detected with a probability

$$P(R_W) = P(R_W|X_{W,0}) = 1$$

.

Adversaries S and A

Since **S** and **A** have at least the same power of **W**, regardless of the number of nodes compromised, $P(R_S|X_{S,x_S}) = P(R_A|X_{A,x_A}) = 1$. Then,

$$P(R_S) = P(R_A) = \sum_{x_S=0}^{n_S} P(X_{S,x_S}) = \sum_{x_A=0}^{n_A} P(X_{A,x_A}) = 1$$

.

Theorem 4.2. *In* L-Tor*:*

1. ***W** breaks **RPA** with probability* $P(R_W) = \frac{1}{k}$
2. ***S** breaks **RPA** with probability* $P(R_S) \leq \frac{n_S p_b}{n} + \frac{1}{k}$
3. ***A** breaks **RPA** with probability* $P(R_A) = \frac{n_A p_b + 1}{k}$

Proof.

AdversaryW

We consider **W** and analyze the three phases of L-Tor. During the set-up phase, in Steps 1 and 2, all the HSs perform the same preliminary operations (i.e., the selection of the introduction points and the advertisement of the service descriptors). Then, no identifying information about the AHS is carried out during these steps.

In Step 3, the client asks the DS for k HSs addresses. Also in the case of collaboration with the DS, by **A1**, **W** can identify the AHS among the k HSs only with probability $\frac{1}{k}$.

We focus now on Step 4. For **W** it is sufficient to identify the ARP to detect the AHS since **W** can immediately discover the mapping between the RPs and the HSs by following the flow of messages exchanged between them.

However, each of the k RPs performs in the same (indistinguishable) way from the point of **W**. Indeed, each RP from x_1 until the ARP x_i extends the circuit of one-hop according to the standard algorithm of Tor. Even though x_i builds a new circuit until x_k, this operation is performed in the same way in which the previous circuit has been extended (according to the standard algorithm of Tor), so that **W** cannot identify the ARP during this extension. Moreover, the only information transmitted that identifies the ARP is the flag provided by the client to x_i, but it is inside a RELAY cell encrypted in Onion fashion so that it changes hop-by-hop and **W** always sees a different cell. Then, by **A1**, in Step 4, **W** can identify the ARP only with probability $\frac{1}{k}$.

Finally, in Steps 5-6 the client contacts k HSs (through the introduction points), and in Step 7, each HS connects to an RP so that no identifying (of the AHS) information or operation occurs during these steps. We conclude that during the set-up phase of L-Tor, **W** breaks **RPA** only with probability $\frac{1}{k}$.

Regarding the forward phase, **W** observes a flow of RELAY cells that are encrypted in Onion fashion and change hop-by-hop. Each cell travels along the entire

circuit from x_1 to x_k by reaching all the k RPs. Moreover, the RELAY cells are forwarded by each of the k RPs to the associated HS. Then, again by **A1**, **W** can identify the ARP (or the AHS) only with probability $\frac{1}{k}$.

Finally, in the response phase, since all the HSs reply to the associated RPs, by **A1**, **W** cannot identify directly the AHS with probability greater than $\frac{1}{k}$. Then, it can try to identify the ARP.

However, similar to the forward phase, a flow of dummy RELAY cells is originated by the last RP x_k and changes hop-by-hop. When it reaches the ARP x_i, there is no way for **W** to understand whether it injects the actual response (in place of dummy cells) or simply forwards the received cells (after adding a layer of encryption).

This concludes the first part of the theorem, i.e., **W** breaks **RPA** with probability

$$P(R_W) = P(R_W|X_{W,0}) = \frac{1}{k}$$

.

Preliminary considerations on S and A

We now consider **S** and **A** that have the capability to compromise some nodes. As a preliminary consideration, it is easy to realize that the adversary (both **S** and **A**) does not obtain any advantage by compromising a node different from an RP. Indeed, all the nodes except for the RPs always receive encrypted data, and then no information about the AHS can be drawn from them.

Instead, if an RP is compromised, two cases may occur: either the RP is the ARP or the RP is different from the ARP. In the first case, the adversary immediately identifies the AHS. In the second case, the adversary does not immediately identify the AHS but reduces the size of the anonymity set by one unity, since it can exclude the HS associated with such an RP from the set of possible recipients.

Formally, we introduce the following events: B^{Adv}={The adversary **Adv** compromises the ARP}, C_i^{Adv}={The adversary **Adv** compromises exactly i RPs different from the ARP (with $0 \leq i \leq k-1$)}.

It holds:

$$P(R_{Adv}|X_{Adv,x_{Adv}}) = P(B^{Adv}|X_{Adv,x_{Adv}}) + \sum_{i=0}^{min(x_{Adv},k-1)} \frac{1}{k-i}$$

The terms $P(B^{Adv}|X_{Adv,x_{Adv}})$ and $P(C_i^{Adv}|X_{Adv,x_{Adv}})$ depends on the type of adversary.

Adversary S

Considering **S**, since it chooses randomly a priori the x_S nodes to compromise, we have

$$P(B^S|X_{S,x_S}) = \frac{x_S}{n}$$

and

$$P(C_i^S|X_{S,x_S}) = \frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n}{x_S}}$$

that derives (with a little modification to take into account the fact that the selected nodes have to be different from the ARP) from the hypergeometric distribution (under the assumption that $n-k-x>0$).

Then for **S**, we have

$$P(R_S|X_{S,x_S}) = \frac{x_S}{n} + \sum_{i=0}^{min(x_S,k-1)} \frac{1}{k-i}\frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n}{x_S}}$$

.

We can simplify the second term as follows.

$$\sum_{i=0}^{min(x_S,k-1)} \frac{1}{k-i}\frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n}{x_S}} = \frac{1}{k}\sum_{i=0}^{min(x_S,k-1)} \frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n}{x_S}} + \frac{1}{k}\sum_{i=0}^{min(x_S,k-1)} \frac{i}{k-i}\frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n}{x_S}}$$

By applying the following properties of the binomial coefficients:

$$\binom{x}{y} = \frac{x+1-y}{y}\binom{x}{y-1}$$
$$\binom{x}{y} = \frac{x}{y}\binom{x-1}{y-1}$$
$$\binom{x}{y} = \frac{x}{x-y}\binom{x-1}{y}$$

we obtain:

$$\frac{1}{k}\sum_{i=0}^{min(x_S,k-1)} \frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n}{x_S}} = \frac{1}{k}\sum_{i=0}^{min(x_S,k-1)} \frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\frac{n}{n-x_S}\binom{n-1}{x_S}} = \frac{n-x_S}{kn}\sum_{i=0}^{min(x_S,k-1)} \frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n-1}{x_S}} = \frac{n-x_S}{kn}$$

where the last step derives from the properties of the hypergeometric distribution.

Furthermore,

$$\frac{1}{k}\sum_{i=0}^{min(x_S,k-1)} \frac{i}{k-i}\frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n}{x_S}} = \frac{1}{k}\sum_{i=1}^{min(x_S,k-1)} \frac{i}{k-i}\frac{\binom{k-1}{i}\binom{n-k}{x_S-i}}{\binom{n}{x_S}} = \frac{1}{k}\sum_{i=1}^{min(x_S,k-1)} \frac{\binom{k-1}{i-1}\binom{n-k}{x_S-i}}{\binom{n}{x_S}} =$$
$$\frac{1}{k}\sum_{i=1}^{min(x_S,k-1)} \frac{\binom{k-1}{i-1}\binom{n-k}{x_S-i}}{\frac{n}{x_S}\binom{n-1}{x_S-1}} = \frac{x_S}{kn}\sum_{t=0}^{min(x_S-1,k-2)} \frac{\binom{k-1}{t}\binom{n-k}{x_S-t-1}}{\binom{n-1}{x_S-1}} \leq \frac{x_S}{kn}\sum_{t=0}^{min(x_S-1,k-1)} \frac{\binom{k-1}{t}\binom{n-k}{x_S-t-1}}{\binom{n-1}{x_S-1}} =$$
$$\frac{x_S}{kn}$$

This leads to:

$$P(R_S|X_{S,x_S}) \leq \frac{x_S}{n} + \frac{x_S}{kn} + \frac{n-x_S}{kn} = \frac{x_S}{n} + \frac{1}{k}$$

Therefore, we obtain:

$$P(R_S) = \sum_{x_S=0}^{n_S} P(R_S|X_{S,x_S})P(X_{S,x_S}) \leq \sum_{x_S=0}^{n_S} \left(\frac{x_S}{n} + \frac{1}{k}\right) P(S_{n_S} = x_S) =$$
$$\frac{1}{n}\sum_{x_S=0}^{n_S} x_S P(S_{n_S} = x_S) + \frac{1}{k}\sum_{x_S=0}^{n_S} P(S_{n_S} = x_S) = \frac{n_S p_b}{n} + \frac{1}{k}$$

where the last step leverages the properties of the binomial distribution.

This concludes the second statement of the theorem, i.e., **S** breaks **RPA** with probability

$$P(R_S) \leq \frac{n_S p_b}{n} + \frac{1}{k}$$

.

Adversary A

Finally, consider **A**. Since it can choose the nodes to compromise after the circuit is established, it attempts to break directly only the RPs. Indeed, there is no advantage to compromising other nodes. Therefore, we assume $x_A \leq n_A \leq k-1$ (if it compromises $k-1$ nodes, it identifies the ARP since either it is one of these $k-1$ or it is the other node).

Then, we have

$$P(B^A|X_{A,x_A}) = \frac{x_A}{k}$$

and

$$P(C_i^A|X_{A,x_A}) = \begin{cases} 0 & \text{if } i \neq x_A \\ \frac{k-x_A}{k} & \text{if } i = x_A \end{cases}$$

That leads to

$$P(R_A|X_{A,x_A}) = \frac{x_A}{k} + \frac{1}{k} = \frac{x_A+1}{k}.$$

Finally,

$$P(R_A) = \sum_{x_A=0}^{n_A} P(R_A|X_{A,x_A})P(X_{A,x_A}) = \sum_{x_A=0}^{n_A} \frac{x_A+1}{k}\binom{n_A}{x_A} p_b^{x_A}(1-p_b)^{n_A-x_A} = \frac{n_A p_b + 1}{k}$$

This ends the proof.

Theorem 4.3. *In* B-Tor*:*

1. ***W** breaks **RPA** with probability* $P(R_W) = \frac{1}{k}$
2. ***S** breaks **RPA** with probability* $P(R_S) \leq \alpha$
3. ***A** breaks **RPA** with probability* $P(R_A) \leq \frac{n_A p_b}{r} + \frac{1}{k}$

where $\alpha = \frac{1}{k} + \frac{1}{rl+1} + \frac{(r+l)n_S p_b}{nk} + \frac{rl(2n-1)n_S p_b}{kn(n-1)} - \frac{rl(n_{SG}^2 p_b^2 + n_S p_b(1-p_b))}{kn(n-1)}$.

Proof.

Adversary W

We start considering **W** and analyze the three phases of B-Tor. In the set-up phase the steps 1–3 and 5–7 are the same as L-Tor, then **W**, by **A1**, can identify the AHS only with probability $\frac{1}{k}$.

In Step 4, there are two cases: either the ARP is a BRP b_i or the ARP is a CRP $x_{i,j}$. In both cases, the client builds a circuit until b_i that is extended until b_r with the same approach of L-Tor, then not allowing **W** to detect that a new circuit is established between b_i and b_r (**W** cannot distinguish the new circuit from a standard extension).

Moreover, in both cases, each $b_z \neq b_i$ builds a circuit until $x_{z,l}$. Regarding b_i, in case (1) it performs as the other BRPs (i.e., builds a circuit until $x_{i,l}$), otherwise (case (2)) it builds a circuit until the ARP $x_{i,j}$ that extends (as in L-Tor) the circuit until $x_{z,l}$ thus making case (2) indistinguishable from case (1). Therefore, we conclude that from the set-up phase **W** breaks **RPA** only with probability $\frac{1}{k}$.

Regarding the forward phase, the same two cases may occur. However, in both cases, all the RPs perform in the same way making the two cases indistinguishable for **W**. Indeed, in both cases, each $b_j \neq b_i$ and each $x_{p,q} \neq x_{i,j}$ performs in the same way. Regarding b_i, in case (1) it performs as b_j while in case (2) it forwards the actual request to $x_{i,1}$ in place of a dummy request. However, since the cells are encrypted and change hop-by-hop, **W** does not distinguish these two cases. Regarding $x_{i,j}$, similarly, in case (1) it performs as the other CRPs $x_{p,q}$ while in case (2) it receives an actual request and replaces it with an encrypted dummy request indistinguishable from the actual request.

Finally, for the response phase, several flows of dummy cells originated by the TCRPs travel until the BRPs. The mechanism is the same as L-Tor so that **W** cannot identify if a CRP injects the actual response or just forwards dummy traffic. Regarding the BRPs, each of them receives two flows and forwards a flow to the previous BRP (in the backward path) at the maximum rate between the rates of the two flows (regardless of which flow contains the actual response), so no information can be drawn by **W**. When the actual response is injected (regardless of whether it is injected by a CRP or a BRP), it replaces the dummy cells in such a way that it is indistinguishable from them.

This concludes the first part of the theorem, i.e., **W** breaks **RPA** with probability

$$P(R_W) = P(R_W|X_{W,0}) = \frac{1}{k}$$

.

Preliminary considerations on S and A

We now consider **S** and **A** that have the capability to compromise some nodes. As in L-Tor they obtain an advantage only by compromising an RP. However, differently from L-Tor, four cases have to be considered when an RP is compromised.

The first case is that the RP is the ARP (either a BRP or a CRP). In this case, **RPA** is broken with probability 1.

In the second case, the RP is a CRP (different from the ARP) and the anonymity set is reduced by one unity since the adversary can exclude such an RP.

In the third case, the RP is a BRP (different from the ARP), such that no CRP in its chain is an ARP. In this case, the anonymity set is reduced by $l+1$ nodes. Indeed, the adversary can exclude such a BRP and the l CRPs associated with it, since the BRP knows that the ARP is not in its chain. Finally, in the last case, the RP is a BRP such that the ARP is a CRP in the chain of this BRP. As in the first case, **RPA** is broken immediately since the BRP knows the ARP is in its chain.

Moreover, when more RPs are compromised, we have to take into account the fact that if the adversary breaks the CRP $x_{i,j}$ and the BRP b_i in the same chain, then the anonymity set is reduced just by $l+1$ (and not of $l+2$) since breaking the CRP does not provide any information additional with respect to that obtained by breaking just the BRP.

Adversary S

Consider the adversary **S**. In favor of security, we assume that even though a compromised CRP is in the same chain of a compromised BRP, the anonymity set is reduced by $l+2$. In other words, we search for an upper bound of the actual probability.

Specifically, we define three events: B^S={**S** compromises a CRP that is the ARP or a BRP b_i such that either b_i is the ARP or the CRP $x_{i,j}$ (with any j) is the ARP}, C_i^S={**S** does not compromise the ARP and compromises i BRPs b_s (with $1 \leq s \leq r$) such that the ARP is not a CRP $x_{s,p}$ for any $1 \leq p \leq l$ }. D_j^S={**S** compromises exactly j CRPs different from the ARP}.

Since $P(B^S)$, $P(C_i^S)$, and $P(D_i^S)$ vary according to the fact that the ARP is a BRP or a CRP, we introduce two additional events: E={The ARP is a BRP} and F={The ARP is a CRP} (that is the complementary event of E).

It holds:

$$P(R_S|X_{S,x_S}) \leq P(E)P(B^S|(X_{S,x_S} \cap E)) + P(F)P(B^S|(X_{S,x_S} \cap F)) +$$
$$\sum_{i=0}^{min(x_S,r-1)} P(E)\frac{1}{k-(l+1)i}P(C_i^S|(X_{S,x_S} \cap E)) + \sum_{i=0}^{min(x_S,r-1)} P(F)\frac{1}{k-(l+1)i}P(C_i^S|(X_{S,x_S} \cap F)) +$$
$$\sum_{j=0}^{min(x_S,lr)} P(E)\frac{1}{k-j}P(D_j^S|(X_{S,x_S} \cap E)) + \sum_{j=0}^{min(x_S,lr)} P(F)\frac{1}{k-j}P(D_j^S|(X_{S,x_S} \cap F))$$

where:

- $P(E) = \frac{r}{k}$
- $P(F) = \frac{rl}{k}$.
- $P(B^S|(X_{S,x_S} \cap E)) = \frac{x_S}{n}$
- $P(B^S|(X_{S,x_S} \cap F)) = 1 - \frac{\binom{2}{0}\binom{n-2}{x_S}}{\binom{n}{x_S}} = \frac{x_S(2n-x_S-1)}{n(n-1)}$
- $P(C_i^S|(X_{S,x_S} \cap E)) = \frac{\binom{r-1}{i}\binom{n-r}{x_S-i}}{\binom{n}{x_S}}$
- $P(C_i^S|(X_{S,x_S} \cap F)) = \frac{\binom{r-1}{i}\binom{n-r-1}{x_S-i}}{\binom{n}{x_S}}$
- $P(D_j^S|(X_{S,x_S} \cap E)) = \frac{\binom{rl}{j}\binom{n-rl-1}{x_S-j}}{\binom{n}{x_S}}$
- $P(D_j^S|(X_{S,x_S} \cap F)) = \frac{\binom{rl-1}{j}\binom{n-rl-1}{x_S-j}}{\binom{n}{x_S}}$ for $j \leq rl-1$
- $P(D_j^S|(X_{S,x_S} \cap F)) = 0$ for $j = rl$

At this point, we solve the four sums separately.

First sum.

$$\sum_{i=0}^{min(x_S,r-1)} \frac{1}{k-(l+1)i} P(C_i^S|(X_{S,x_S} \cap E)) = \sum_{i=0}^{min(x_S,r-1)} \frac{1}{k-(l+1)i} \frac{\binom{r-1}{i}\binom{n-r}{x_S-i}}{\binom{n}{x_S}} =$$
$$\frac{1}{k} \sum_{i=0}^{min(x_S,r-1)} \frac{\binom{r-1}{i}\binom{n-r}{x_S-i}}{\binom{n}{x_S}} + \frac{1}{k} \sum_{i=0}^{min(x_S,r-1)} \frac{i(l+1)}{k-(l+1)i} \frac{\binom{r-1}{i}\binom{n-r}{x_S-i}}{\binom{n}{x_S}}$$

The first term is the same as in the proof of Theorem 4.2 with r in place of k, then we obtain:

$$\sum_{i=0}^{min(x_S,r-1)} \frac{1}{k-(l+1)i} \frac{\binom{r-1}{i}\binom{n-r}{x_S-i}}{\binom{n}{x_S}} = \frac{n-x_S}{nk} + \frac{l+1}{k} \sum_{i=0}^{min(x_S,r-1)} \frac{i}{k-(l+1)i} \frac{\binom{r-1}{i}\binom{n-r}{x_S-i}}{\binom{n}{x_S}} \leq$$
$$\frac{n-x_S}{nk} + \frac{l+1}{k} \sum_{i=0}^{min(x_S,r-1)} \frac{i}{r-i} \frac{\binom{r-1}{i}\binom{n-r}{x_S-i}}{\binom{n}{x_S}} \leq \frac{n-x_S}{nk} + \frac{(l+1)x_S}{kn} = \frac{n+lx_S}{nk}$$

where the last sum was solved in the proof of Theorem 4.2 with r in place of k.

Second sum.

$$\sum_{i=0}^{min(x_S,r-1)} \frac{1}{k-(l+1)i} P(C_i^S|(X_{S,x_S} \cap F)) = \sum_{i=0}^{min(x_S,r-1)} \frac{1}{k-(l+1)i} \frac{\binom{r-1}{i}\binom{n-r-1}{x_S-i}}{\binom{n}{x_S}} \leq$$
$$\sum_{i=0}^{min(x_S,r-1)} \frac{1}{k-(l+1)i} \frac{\binom{r-1}{i}\binom{n-r}{x_S-i}}{\binom{n}{x_S}} \leq \frac{n+lx_S}{nk}$$

Third sum.

$$\sum_{i=0}^{min(x_S,rl)} \frac{1}{k-i} P(D_i^S|(X_{S,x_S} \cap E)) = \sum_{i=0}^{min(x_S,rl)} \frac{1}{k-i} \frac{\binom{rl}{i}\binom{n-rl-1}{x_S-i}}{\binom{n}{x_S}} \leq$$
$$\sum_{i=0}^{min(x_S,rl)} \frac{1}{rl+1-i} \frac{\binom{rl}{i}\binom{n-rl-1}{x_S-i}}{\binom{n}{x_S}} \leq \frac{1}{rl+1}$$

The last sum is solved as in the proof of Theorem 4.3 with $rl+1$ in place of k

Fourth sum.

$$\sum_{i=0}^{min(x_S,rl)} \frac{1}{k-i} P(D_i^S|(X_{S,x_S} \cap F)) = \sum_{i=0}^{min(x_S,rl-1)} \frac{1}{k-i} \frac{\binom{rl-1}{i}\binom{n-rl-1}{x_S-i}}{\binom{n}{x_S}} \leq$$
$$\sum_{i=0}^{min(x_S,rl)} \frac{1}{rl+1-i} \frac{\binom{rl}{i}\binom{n-rl-1}{x_S-i}}{\binom{n}{x_S}} \leq \frac{1}{rl+1}$$

where the last sum is solved as in the proof of Theorem 4.3 with rl in place of $k-1$.

Then, it results:

$$P(R_S|X_{S,x_S}) \leq \frac{rl}{k} \frac{x_S(2n-x_S-1)}{n(n-1)} + \frac{n+(r+l)x_S}{nk} + \frac{1}{rl+1}$$

Finally, we obtain:

$$P(R_S) = \sum_{x_S=0}^{n_S} P(R_S|X_{S,x_S})P(X_{S,x_S}) \leq$$
$$\sum_{x_S=0}^{n_S} \left(\frac{rl}{k} \frac{x_S(2n-x_S-1)}{n(n-1)} + \frac{n+(r+l)x_S}{nk} + \frac{1}{rl+1} \right) P(S_{n_S} = x_S)$$
$$= \frac{1}{k} + \frac{1}{rl+1} + \frac{(r+l)n_S p_b}{nk} + \frac{rl(2n-1)n_S p_b}{kn(n-1)} - \frac{rl}{kn(n-1)} \sum_{x_S=0}^{n_S} x_{SG}^2 P(S_{n_S} = x_S) =$$
$$\frac{1}{k} + \frac{1}{rl+1} + \frac{(r+l)n_S p_b}{nk} + \frac{rl(2n-1)n_S p_b}{kn(n-1)} - \frac{rl(n_{SG}^2 p_b^2 + n_S p_b(1-p_b))}{kn(n-1)} = \alpha$$

This concludes the second statement of the theorem, i.e., **S** breaks **RPA** with probability

$$P(R_S) \leq \alpha$$

Adversary A

Finally, consider **A**. Since it can choose the nodes to compromise after the circuit is established, it attempts to break the BRPs directly since there is no advantage to compromising another node in place of a BRP. Therefore, we assume $x_A \leq n_A \leq r$. If $x_A = r$, $P(R_A|X_{A,x_A}) = 1$. Then, we study $x_A \leq r-1$.

If we denote by B^A the event B^A ={**A** compromises a CRP that is the ARP or a BRP b_i such that either b_i is the ARP or the CRP $x_{i,j}$ (with any j) is the ARP}, we have that:

$$P(R_A|X_{A,x_A}) = P(B^A|X_{A,x_A}) + (1 - P(B^A|X_{A,x_A}))\frac{1}{k - x_A(l+1)} =$$
$$\frac{x_A}{r} + \frac{r - x_A}{r}\frac{1}{k - x_A(l+1)} = \frac{x_A}{r} + \frac{r - x_A}{r}\frac{1}{r(l+1) - x_A(l+1)} =$$
$$\frac{x_A}{r} + \frac{r - x_A}{r}\frac{1}{(l+1)(r - x_A)} = \frac{x_A}{r} + \frac{1}{k}$$

Finally:

$$P(R_A) = \sum_{x_A=0}^{n_A} P(R_A|X_{A,x_A})P(X_{A,x_A}) \leq \sum_{x_A=0}^{n_A}\left(\frac{x_A}{r} + \frac{1}{k}\right)\binom{n_A}{x_A}p_b^{x_A}(1-p_b)^{n_A - x_A} =$$
$$\frac{n_A p_b}{r} + \frac{1}{k}$$

This ends the proof.

At this point, we give our result about relationship anonymity by proving that it coincides, in the considered protocols against the considered adversaries, with the recipient anonymity.

Theorem 4.4. *For* Tor, L-Tor, B-Tor, *for any type of adversary* ***W***,***S***,***A***, *the probability to break* ***RLA*** *is the same as the probability to break* ***RPA****.*

Proof.

From [219], **RPA** implies **RLA**. Therefore, if **RPA** holds with probability p, then **RLA** holds with probability $p' \geq p$. Since for **W**, **S**, and **A**, the sender is known with probability 1, we conclude that $p = p'$.

To conclude the section, we compare the three protocols. First, Tor does not offer any anonymity guarantee.

Against **W**, both L-Tor and B-Tor offer the same probability to break **RPA** and **RLA** equal to $\frac{1}{k}$, therefore B-Tor should be adopted due to the benefits in terms of performance.

Observe that, as expected, in both L-Tor and B-Tor, if we consider the probabilities to break **RPA** and **RLA** against both **S** and **A**, they include at least the term $\frac{1}{k}$ since such adversaries have at least the power of **W**.

Furthermore, we obtain that L-Tor offers better resistance to the break of **RPA** and **RLA** than B-Tor.

However, to give a concrete idea of the gain in terms of anonymity obtained by L-Tor and B-Tor, we provide a numeric example.

In particular, we consider the realistic values $n = 7000$, $r = l = 7$, $k = r(l+1) = 56$, $p_b = 0.05$, $LT = 1$ hour, $\delta_S = 6$ hours, $\delta_A = 10$ minutes, $\Delta_S = 30$ days. These values lead to $n_S = 120$ and $n_A = 6$.

Then for L-Tor, we obtain, $P(R_W) \simeq 1{,}79\%$, $P(R_S) \leq 1{,}87\%$, $P(R_A) \simeq 2{,}32\%$ and for B-Tor to $P(R_W) \simeq 1{,}79\%$, $P(R_S) \leq 4\%$, $P(R_A) \leq 6{,}07\%$.

By these results, we can observe that a small difference exists between **W** and **S** (by making B-Tor more suitable for its better performance also against **S**) while there exists a price to pay (we move from 2,32% to 6%) against **A** when adopting B-Tor in place of L-Tor.

Anyway, also in this toy example, we can see that B-Tor offers very good protection against all adversaries.

4.8 Analytical Evaluation

Protocol	Set-up	Forward	Return
L-Tor	$d(2k^2+10k+6)$	$(k+6)d+\frac{w}{B}$	$(k+6)d+\frac{q}{B}$
B-Tor	$d(2r^2+10r+6+2l(l-1))$	$(r+l+6)d+\frac{w}{B}$	$(r+l+6)d+\frac{q}{B}$

Table 4.3: Latency in L-Tor and B-Tor for the three phases.

Through this section, we provide an analytical evaluation of L-Tor and B-Tor, conducted on the basis of some slight approximations. Specifically: (1) we assume that all the nodes have the same bandwidth; (2) we assume all the links have the same delay; and (3) we consider that the cost of the operations in the set-up phase involving the DS is negligible.

Furthermore, in this section, we pursue another goal, i.e., to derive a “design formula" for B-Tor allowing us to set the proper values of r and l to obtain a given anonymity level k.

We denote by d the propagation time (in ms) of any link between two nodes and by B the bandwidth of the nodes. Therefore, to exchange a message of size s between two nodes directly connected to each other, it requires $\frac{s}{B}+d$ ms.

Now, observe that, for both L-Tor and B-Tor, all the messages exchanged end-to-end between the client and each RP in the set-up phase are forwarded in a single cell of size $c = 512$ bytes. For realistic values of B, the ratio $\frac{c}{B}$ is negligible compared to d. Then, we assume that a cell requires xd ms to walk x links.

Consider the set-up phase of L-Tor. To build the L-Tor circuit, the client first builds a Tor circuit of two ORs with the first RP. To do this, the client sends a 1-cell message (including its DH parameter) to the first OR, which replies with a 1-cell message. Then, another 1-cell message is sent to the second OR through the first OR (thus walking two links). Then, the second OR responds with another 1-cell message walking two links too. In sum, this preliminary step requires 6 elementary (i.e., 1-cell) single-hop transmissions. At this point, the client sends a 1-cell message

to the first RP (thus walking 3 links). The first RP replies with a 1-cell message (walking 3 links too). Then, the client replies with a 1-cell message containing the rendezvous cookie to the first RP (again, walking 3 links), which, in turn, replies with another 1-cell message (and, then, walking 3 links). Therefore, to provide the first RP with the secret key, 12 elementary single-hop transmissions occurred and a time of $12d$ ms is required. Now, to exchange the same information with the second RP, 16 transmissions occur since the four 1-cell messages exchanged have to cross the first RP (thus, 1 further hop for each message is required). This pattern is iterated for the other RPs of the L-Tor circuit.

Therefore, the total latency is: $L_L^s = d\left(6 + \sum_{i=1}^{k}(4i+8)\right) = d(2k^2+10k+6)$.

Observe that, this latency increases quadratically with k, exactly as in Tor, with the difference that Tor is not thought to set up circuits with numerous relays. Of course, this is not a good point in favor of L-Tor. For example, for a realistic value of $d = 45$ ms and $k = 50$, we obtain that the set-up phase requires $L_L^s \simeq 4$ minutes, which could be acceptable in some applications, but, in general, results in a perception of low network performance.

Regarding the forward phase, by also considering the size of the request w, the time before the AHS receives the request depends on the position of the ARP. In the worst case, the ARP is in the last position. This leads to a latency of $L_L^f = (k+6)d + \frac{w}{B}$, since there are three links (two relays) between the client and the first RP, $k-1$ links between the first RP and the last RP, and four links (three relays) between the last RP and the AHS.

Finally, for the response phase, independently of its position, the ARP has to wait for the cells originated by the last RP before injecting the response. If we denote by q the size of the response, then we have $L_L^r = (k+6)d + \frac{q}{B}$.

Now, we apply the same reasoning for B-Tor. In the set-up phase, the main advantage is that the client builds a circuit with just r BRPs, and each BRP builds (in parallel) a circuit with l CRPs. Therefore, by considering the two hops from the client to the first BRP, the set-up time for B-Tor is $L_B^s = d(2r^2+10r+6+2l(l+1))$. If we consider $d = 45$ ms and $r = l = 7$ that leads to $k = 56$, we have $L_B^s \simeq 12.9$ seconds, which represents a relevant improvement with respect to L-Tor.

For the forward and response phase, by assuming the worst case in which the ARP is the CRP $x_{r,l}$ (i.e., the last CRP of the last chain), we obtain $L_B^f = (r+l+6)d + \frac{w}{B}$ and $L_B^r = (r+l+6)d + \frac{q}{B}$ since the messages have to walk (round trip) the entire path of CRPs of length l from $x_{r,l}$ until b_r and the entire path of length $r-1$ from b_r to b_1. Moreover, we include the three links between the client and b_1 and the four links between $x_{r,l}$ and the AHS.

All the previous results are summarized in the table in Figure 4.3.

To conclude this section, we derive the design formula to set the parameters r and l in B-Tor.

Specifically, given a privacy level k, we found the optimal values (minimizing latency) of r and l to build a circuit in B-Tor. We can minimize either the set-up latency or the forward latency (equal to the return latency). Regarding the set-up latency, by replacing $l = \frac{k-r}{r}$ in L_B^s and by setting $\frac{\partial L_B^s}{\partial r} = 0$, we obtain the optimal value of r, say $r_o^s \simeq \lceil \frac{1}{4}\sqrt{16k+9} - 3 \rceil$.

Then, given a privacy level k, we can set $r = r_o^s$ and $l = l_o^s = \lceil \frac{k-r_o^s}{r_o^s} \rceil$.

Similarly, to minimize the latency of the forward phase and return phase, we obtain $r_o^f = \lceil \sqrt{k} \rceil$ and $l_o^f = \lceil \sqrt{k} - 1 \rceil$.

Observe that, for increasing k, r_o^s approaches r_o^f, therefore approximately, the value $r = \lceil \sqrt{k} \rceil$ minimizes both the set-up latency and the communication latency. Then, we move from the set-up latency of L-Tor, which increases quadratically with the number of RPs, to the latency of B-Tor, which increases linearly with the number of RPs. Moreover, we move from the communication latency of L-Tor, which increases linearly with the number of RPs, to the latency of B-Tor, which increases according to the square root of the number of RPs. Thus, in any case, we obtain the reduction of a magnitude order.

4.9 Experimental Validation

In this section, we provide a performance evaluation of L-Tor and B-Tor and compare them with the standard Tor protocol when the hidden services mechanism is enabled. The experiments have been conducted through *ns-3* [122], a discrete-event network simulator for Internet systems. Since we are interested in maintaining all the Tor features, we base our implementation on *nstor* [268], a Tor module for ns-3 publicly available on GitHub. As stated in [268], this code is modeled along the lines of the original Tor software, which is an additional guarantee of the accuracy of our results.

4.9.1 Experimental Setup

According to the Tor design [258], Tor relays use a token bucket approach to enforce a long-term average rate of incoming bytes. Therefore, in our simulations, we need to set a realistic average bandwidth value for each relay. Specifically, we configure the relays of the Tor network so that they offer an average bandwidth of 526.097 KB/s per Tor client. This value derives from [111], in which the authors experimented, on the real Tor network, an average bandwidth of 404.69 KB/s per Tor client. However, the paper refers to the value of bandwidth in 2020, and in the last two years, this

value has grown. Then, by the official Tor metrics [226], we found that the total bandwidth value has grown by 30% from 2020 to 2022. Then, we set the relays so that the average bandwidth per client can be considered equal to $404.69 \cdot 1.3 = 526.097$ KB/s.

Once the bandwidth is set, we have to set other network parameters, namely the link delay of the network. To do this, we again refer to the official Tor metrics. Specifically, we consider the average time to download a file in the real Tor network and set the delay of the links so that the same time is obtained in our simulations. This value depends on two factors: the size of the file and the performance of the relays. Regarding the first factor, the Tor metrics report the download time for three sizes: $50KiB$, $1MiB$, and $5MiB$. Regarding the second factor, several clusters of relays (with different throughputs) are reported in the Tor metrics. We consider the cluster op-us6, representing relays offering an intermediate throughput with respect to the other clusters.

Then, starting from the cluster op-us6 and a file size of $50KiB$, we set the average delay of the links in the network, so that the total download time in our simulations is the same as reported in the Tor metrics for $50KiB$. This leads to an average link delay of 110 ms. To confirm the validity of our result, we set this average delay in our simulation environment and measure the download times of the files of sizes $1MiB$ and $5Mib$. The times obtained are very close to those reported in Tor metrics (with an error bounded by 10%). We conclude that the above configuration of the network allows us to obtain a realistic representation of the Tor network when users rely on the hidden services mechanism.

In principle, the same network configuration (i.e., the same values of bandwidth of the relays and delay of the links) should be adopted to simulate L-Tor and B-Tor. However, to be fair, we have to consider a negative effect on the bandwidth in the adoption of L-Tor and B-Tor. Indeed, the circuits of these two protocols involve a greater number of relays than the standard Tor. This means that the average bandwidth that each relay can offer to each client, in the case of L-Tor or B-Tor, is less than the value of 526.097 KB/s considered for Tor. Then, we found a scaling factor to set the proper average bandwidth offered by the relays per client. This scaling factor is obtained as follows. We denote by n the total number of relays of the Tor network and by c the (average) total number of simultaneously active clients forming a circuit. We recall that a standard Tor circuit (with hidden services) involves 6 relays. Then, if the c clients ask for a standard circuit, each relay, on average, will serve $\frac{6c}{n}$ circuits. On the other hand, by denoting by k the number of relays involved in L-Tor or B-Tor, each relay will serve, on average, $\frac{kc}{n}$ circuits. Therefore, when considering

the network conditions of B-Tor and L-Tor, we divide the bandwidth of 526.097 KB/s by a factor $\frac{k}{6}$.

However, this factor reflects the condition in which **all** the standard Tor circuits are replaced by B-Tor or L-Tor circuits. Actually, this is a very unlikely worst case. Indeed, we can assume that only a portion of Tor users are interested in obtaining the high privacy protection of B-Tor and L-Tor, whereas the rest of the clients use the standard Tor. Therefore, in the analysis we perform, we consider different network configurations in which standard Tor circuits and B-Tor\L-Tor circuits coexist. In particular, we consider 3 network configurations in which the B-Tor\L-Tor circuits are: the 100% (worst case), the 75%, and the 50% of the total circuits. In favor of the significance of the experimental results, we did not consider lower percentages, which would be advantageous for our B-Tor\L-Tor. The 100% configuration corresponds to dividing by a factor of $\frac{k}{6}$. For the other configurations, we proportionally scale this factor as follows:

$$x = \frac{k}{6} \cdot \frac{c[k\alpha + 6(1-\alpha)]}{kc}$$

where $\alpha \in [0,1]$ represents the fraction of B-Tor\L-Tor circuits with respect to the total number of circuits and x represents the resulting scaling factor. Indeed, the denominator kc represents the total number of relays involved in the case of 100% B-Tor\L-Tor, while the numerator $c[k\alpha + 6(1-\alpha)]$ represents the total number of relays involved in case the network includes the fraction α of B-Tor\L-Tor circuits.

Therefore, we obtain

$$x = \alpha \cdot \frac{k}{6} + (1-\alpha)$$

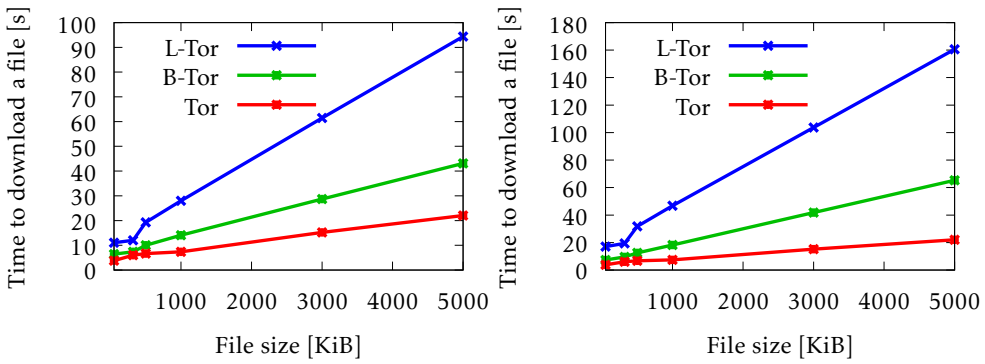
According to the experimental settings, for α we consider only the values 0.50, 0.75, and 1.

However, the so-obtained value for x is not the final one. Indeed, according to the Tor metrics, only $\frac{1}{3}$ of the available bandwidth of the network is currently used by the clients (`https://metrics.torproject.org/bandwidth.html`). Therefore, this unused portion of bandwidth would be employed by B-Tor\L-Tor without resulting in relay bandwidth reduction. Thus, the final bandwidth we set for the simulation of B-Tor \L-Tor will result equal to $\frac{3 \cdot 526.097}{x}$ KB/s.

4.9.2 Results

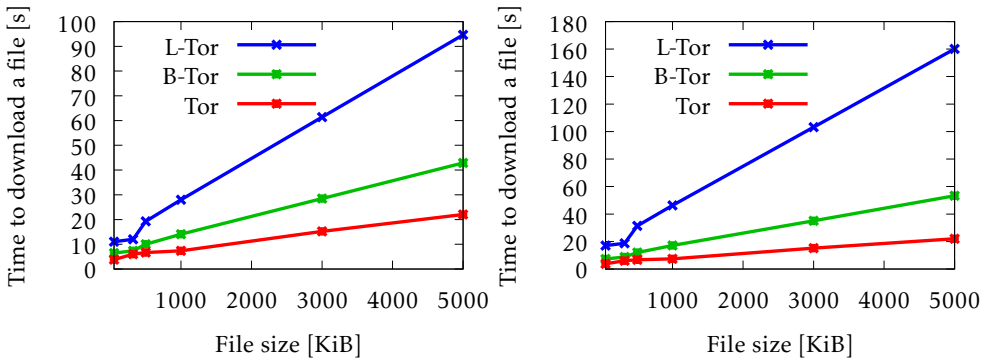
We simulated both B-Tor and L-Tor considering two topologies having $k = 30$ and $k = 56$, respectively. Specifically, for B-Tor, we set $r = 5$ and $l = 5$, in order to obtain $k = 30$, and we set $r = 7$ and $l = 7$, in order to obtain $k = 56$. Indeed, according to the design formula obtained in Section 4.8, to minimize the latency of the forward

and return phase, r and l should be of the magnitude of $\sqrt{k}$. As explained in Section 4.9.1, we consider 3 network configurations in which the B-Tor\L-Tor circuits are: the 100% , the 75%, and the 50% of the total circuits. Therefore, we set the bandwidth for each relay accordingly. Once the parameters for both protocols are fixed, we measure the time needed to download files of different sizes (i.e., $50KiB$, $320KiB$, $500KiB$, $1MiB$, $3MiB$, and $5MiB$) over the network. The results of our simulations are reported in Figures 4.7, 4.8, and 4.9.



(a) $k = 30$. (b) $k = 56$.

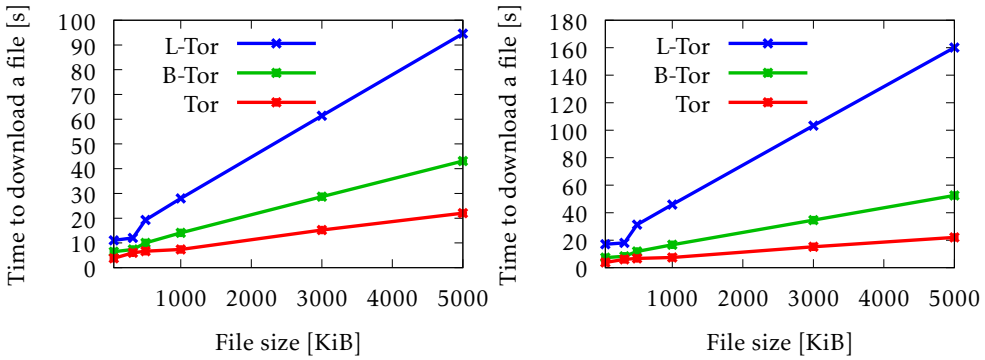
Fig. 4.7: Time to download a file (s) in Tor, B-Tor, and L-Tor when 100% of the total circuits are B-Tor (or L-Tor) circuits.



(a) $k = 30$. (b) $k = 56$.

Fig. 4.8: Time to download a file (s) in Tor, B-Tor, and L-Tor when 75% of the total circuits are B-Tor (or L-Tor) circuits.

If we compare B-Tor to Tor (table in Figure 4.12) we can see that the latency required by B-Tor to download a file is higher than Tor, under the same file size. How-



(a) $k = 30$. (b) $k = 56$.

Fig. 4.9: Time to download a file (s) in Tor, B-Tor, and L-Tor when 50% of the total circuits are B-Tor (or L-Tor) circuits.

ever, the smaller the file to download, the lower the price to pay. Indeed, considering the latency to download $320KiB$, Tor requires 6.02s, while B-Tor (considering a network in which 100% of the circuits are B-Tor circuits) requires 7.38s, when $k = 30$, and 9.55s, when $k = 56$. Hence, moving from Tor to B-Tor to download $320KiB$, leads to an absolute increase of at most 3.53s. Conversely, if we consider the latency to download $5MiB$, Tor requires 22.05s, while B-Tor (considering the same percentage of circuits as before) requires 43.10s, when $k = 30$, and 65.34s, when $k = 56$. Therefore, the price to pay when moving from Tor to B-Tor to download $5MiB$ is more consistent, since the absolute increase in terms of latency is at most 43.29s.

Clearly, the benefits in terms of anonymity provided by B-Tor, with respect to Tor, come at a price in terms of performance. However, if we consider the time required to download a web page (which corresponds to downloading a file of $320KiB$ [142]) this price appears to be acceptable.

As expected, B-Tor always outperforms L-Tor, under the same network conditions and the same k. The difference is most appreciable when considering the latency values for downloading $5MiB$. Indeed, considering the scenario in Figure 4.7a, B-Tor requires 43.1s, while L-Tor requires 94.39s, with an increase by a factor of 2.19. Similarly, considering the scenario in Figure 4.7b, B-Tor requires 65.34s, while L-Tor requires 160.57s, with an increase by a factor of 2.46.

Moreover, B-Tor is strongly advantageous when considering a higher k value. Indeed, considering for instance the scenario in which the B-Tor\L-Tor circuits are the 100% of the total circuits, when $k = 30$ (Figure 4.7a), B-Tor leads to a time to transfer a file lower than L-Tor of at least 4.6s. On the contrary, under the same

percentage of B-Tor\L-Tor circuits, when $k = 56$ (Figure 4.7b), B-Tor leads to a time to download a file lower than L-Tor of at least 9.7s.

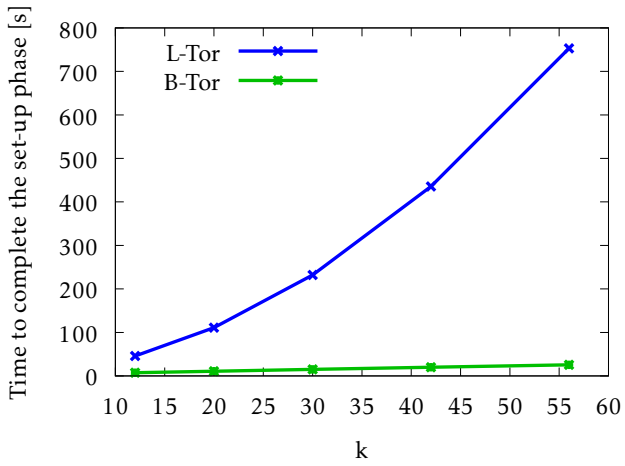


Fig. 4.10: Time to complete the set-up phase (s) in B-Tor and L-Tor varying k.

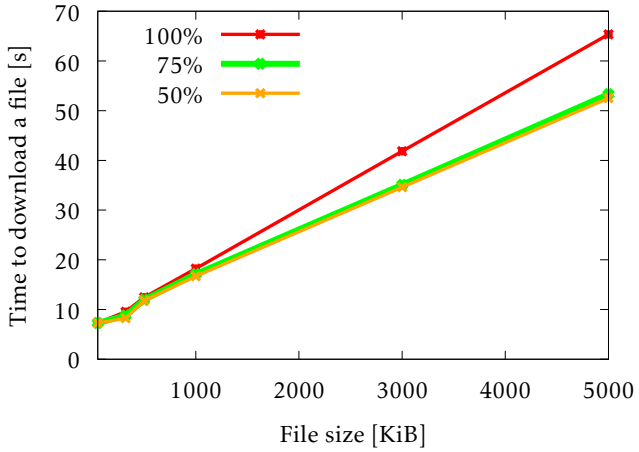


Fig. 4.11: Time to download a file (s) in B-Tor when $k = 56$.

These results can be easily explained by considering the number of hops through which the request and the related response travel. Indeed, for $k = 30$, in L-Tor, the number of hops crossed by request and response, is at most $k+6 = 36$, while in B-Tor the number of hops, is at most $r+l+6 = 16$. Instead, for $k = 56$ in L-Tor the number of hops, is at most $k+6 = 62$, while in B-Tor, the number of hops is at most $r+l+6 = 20$. This clearly leads to an advantage of B-Tor over L-Tor in terms of latency, under the same anonymity set. This result applies also to the set-up phase as reported in Figure 4.10. Therein, we show that, as k increases, in L-Tor the time to perform the set-up phase increases faster than in B-Tor. In particular, the time to perform the set-up

Protocol	320KiB	5MiB
Tor (six-hop)	6.02s	22.05s
L-Tor ($k = 30$)	12.02s	94.39s
B-Tor ($k = 30$)	7.38s	43.1s
L-Tor ($k = 56$)	19.41s	160.57s
B-Tor ($k = 56$)	9.55s	65.34s

Fig. 4.12: End-to-End Latency comparison among Tor, L-Tor, and B-Tor. For the last two, we consider 100% of circuits in the network.

phase increases linearly with k in B-Tor, while it increases quadratically with k in L-Tor, thus confirming the considerations made in Section 4.8.

To complete our reasoning, we compare the time to download a file in B-Tor, considering $k = 56$, when the number of B-Tor circuits in the network represents the 100%, 75%, and the 50% of the total circuits. The comparison is reported in Figure 4.11. The picture shows that for file sizes smaller than 1MiB the three plots are nearly overlapping. Indeed, the time required to download a file of 1MiB when the percentage is 50% is 16.69s, while the time required to download the same file when the percentage is 100% is 18.23s. Thus, the increase of the number of B-Tor circuits in the network impacts the download of relatively small files of at most 1.54s. Conversely, for bigger file dimensions (greater than 1MiB) the difference in the measured latency at the three percentages becomes more pronounced. For instance, the time required to download a file of 5MiB when the percentage is 50% is 52.53s, while the time required to download the same file when the percentage is 100% is 65.34s, with an increase of 12.81s.

4.10 Related Work

Our work is related to the domain of anonymous routing, in which a wide literature is available, which witnesses the strong attention of researchers towards this topic. Specifically, our proposal refers to the Tor protocol. Tor [258] is the de-facto standard implementation of the Onion protocol, originally proposed in [108].

A significant part of the literature focused on the study of Tor security, highlighting many vulnerabilities[7]. Indeed, the Tor overlay network does not provide anonymous guarantees in the severe threat model of a global passive adversary [209], which can observe the entire network traffic. In addition, even though we relax the adversary's power, many attacks are still possible [150, 239, 91].

The class of *traffic analysis attacks* [17, 88, 193, 190], in which the adversary analyzes the traffic to find correlations, is the most famous class of attacks. This class includes both temporal attacks [167, 259, 107], in which the adversary observes the timing of messages arriving and leaving from nodes to find correlations, and traffic confirmation attacks [231], in which the adversary controls and observes two possible end-relays of a Tor circuit to conclude that they actually belong to the same circuit.

Tor is not resistant also to watermarking attacks [136], in which the adversary manipulates the traffic stream by introducing an identifiable pattern. Another category of attacks involves the selection of the relays used to build the Tor circuit. The standard selection is based on network and CPU performance reported by the nodes themselves. This enables self-promotion attacks [247]. Some recent proposals [146, 54, 307] overcome this problem.

Particular attention has been also devoted to Tor and Onion performance [19, 213, 157]. In [51], accurate measurement techniques are proposed and applied to de-anonymization attacks. Some studies are devoted to investigating how to deanonymize Tor by using external information [143].

This work focuses on the weakness of Tor against global adversaries. In particular, we extend Tor to obtain recipient anonymity, which is enough to have relationship anonymity [219], in a threat model considering global eavesdroppers, possibly compromising the secret keys of some nodes of the network. The recipient is hidden among a sufficiently large anonymity set whose size can be a priori configured. Therefore, we refer to the notion of *communication k-anonymity* [274]. Moreover, cover traffic is also enabled. It is well known in the literature [71] that any solution achieving such a goal requires the introduction of *cover traffic*, i.e., dummy traffic exchanged in the network with the aim to hide the real traffic.

To understand how to achieve sender anonymity in Tor, the reader can refer to Chapter 3 and [43].

Extending the perspective to the general landscape of anonymous communication networks, resistance to global adversaries has been obtained in the past by following three approaches: *buses* [124, 21, 296], *mixnets* [98, 61, 158, 279, 165, 61, 243], whose original definition has been given in [59], and *DC-Nets* [245], based on secure multi-party cryptographic protocols.

5

An anonymity protocol for uplink-intensive applications

Sender anonymity in network communication is an important problem, widely addressed in the literature. Mixnets, combined with onion routing, represent certainly the most concrete and effective approach to achieving the above goal. In general, the drawback of these approaches is that anonymity has a price in terms of traffic overhead and latency. On the Internet, to achieve scalability and not require relevant infrastructure and network-protocol changes, only P2P overlay protocols can be adopted. Among these, the most representative proposal is certainly Tarzan, which is designed to obtain strong anonymity still preserving low-latency applications. In this chapter, we propose C-Tarzan, an anonymous overlay routing protocol extending Tarzan. Through experimental analysis, we show that C-Tarzan outperforms Tarzan in the case of uplink-intensive applications.

5.1 Introduction

The most known and used anonymous protocol is Tor [258]. However, unless to implement non-trivial solutions as those discussed in Chapters 3 and 4, anonymity is easily broken under even weak threat models [150]. A challenging goal is to guarantee robust sender anonymity because it is enough to achieve relationship anonymity [219]. By robust, we mean that both passive sniffers and malicious participants cannot distinguish whether a node initiates a message or simply relays it.

The most effective approaches existing in the literature achieving the above goal are based on the concept of mixnet [59] including cover traffic. Mixnet protocols rely on intermediate servers (called *mix-nodes*) that mix the messages coming from different sources to hide the relationship between the incoming messages and the outcoming messages from the mix-nodes. When cover traffic is included in mixnets, serious problems of traffic overhead may arise. While a wide literature regarding mixnets exists, a few proposals mixnet-based oriented to a concrete Internet (low-latency) implementation of the notion of mixnet, including cover traffic, are avail-

able. Among these, if we refer to P2P approaches (thus not requiring infrastructure changes), the most meaningful proposal is certainly Tarzan [98].

Despite its age, Tarzan is the only effective proposed anonymous routing protocol guaranteeing low latency even in large-scale Internet scenarios. Indeed, the protocol allows a client to anonymously contact a server through a tunnel whose length is independent of the number of nodes participating in the peer-to-peer network. As a matter of fact, Tarzan implements a peer-to-peer overlay network at the IP layer, in which peers collaborate with each other to implement anonymous tunnels through which a client may reach a proxy node (called PNAT) from which the server is reached. Another advantage of Tarzan with respect to recent state-of-the-art approaches is that, unlike the emerging mixnets that adopt centralized and explicit shuffling nodes ([223]), the peer-to-peer approach makes the solution more robust against possible attacks on the nodes of the route (or their collusion). Indeed, all the nodes of the network are potentially sender or relay nodes and then there are no explicit targets for the attacker.

The aim of this chapter is to understand whether the change of type of Internet traffic due to various reasons (emerging applications for IoT, M2M, cloud, etc.), for which uplink traffic is increasingly increasing [206, 289, 242, 24], might allow us to find some improvements to the Tarzan approach to make it more suitable to the new scenario.

The study conducted in this chapter leads to the definition of a new P2P overlay anonymous protocol, called *C(yclic)-Tarzan*, which outperforms Tarzan in the case of uplink-intensive applications. The core idea is that the topology of the overlay network allows us to set in the network just unidirectional cover traffic instead of the bidirectional traffic required in Tarzan.

Our study is based on the well-known trilemma, called the *anonymity trilemma* [74], which states the existence of a trade-off between three metrics: the anonymity set size, the latency, and the cover traffic level. Specifically, we show that for uplink-intensive applications, by fixing the same latency and the same cover traffic volume, C-Tarzan offers a greater anonymity set size than Tarzan.

5.2 Background: The Tarzan Protocol

In this section, we provide the technical background about the Tarzan Protocol [98]. We focus just on the main aspects useful to understand the approach we propose in this chapter.

Tarzan is a peer-to-peer anonymous IP network overlay. It offers a degree of anonymity against both a number of malicious nodes and a global adversary able to observe the entire traffic exchanged in the network.

Each node, in order to communicate anonymously with a destination, builds a tunnel composed of a sequence of nodes in which the last node communicates with a special node, called *PNAT*, which acts as a proxy towards the destination.

Each intermediate node of the tunnel acts as a relay by forwarding the messages coming from the previous node. Anyway, since it does not know its position in the tunnel, it is not able to identify the originator of the traffic.

In Tarzan, the construction of the tunnel (i.e., the choice of the intermediates nodes) is not left entirely free to the initiator, which has to satisfy some constraints.

Specifically, each node is associated with a group of nodes called *mimics*. To build the tunnel, the initiator a chooses as the first relay one of its mimics, say b_i. Then, b_i communicates to the initiator the set of its mimics, and a will choose the second relay of the tunnel among the nodes of this set. This procedure is iterated until the tunnel reaches a certain length.

To send messages through this tunnel, the initiator needs to exchange a symmetric key with each node of the tunnel. This procedure is similar to the construction of a virtual circuit in the Tor Protocol [258]. To do this, the initiator first exchanges a symmetric key with the first relay of the tunnel, then it exchanges a symmetric key with the second relay through the first relay, then it exchanges a symmetric key with the third relay through the first two relays of the tunnel built so far, and so on. To exchange a symmetric key with a relay, the initiator encrypts it by using the public key of such a relay. In such a way, each node of the tunnel cannot tell which it is exchanging the key with.

Once exchanged these keys, the messages can be sent through the tunnel encrypted in a layered fashion. This means that the initiator first encrypts the message with the key of the last relay of the tunnel, then encrypts the result with the key of the second-last relay, and so on. A relay receiving a message removes its layer of encryption and forwards the message to the next relay.

A key role in the Tarzan Protocol is played by the selection of mimics. Tarzan relies on a gossip protocol so that each node can discover the other peers of the networks. Among these peers, the node has to select k mimics. Observe that, since each node selects k mimics, we can expect, on average, that it is selected as a mimic from other k nodes. Therefore, the average number of mimics for each node is $2k$.

A node establishes, with each of its mimics, a bidirectional cover traffic flow into which real data can be inserted, indistinguishable, from dummy traffic. To do this, a

symmetric hop-by-hop key is exchanged when a node connects to a new mimic and all the traffic exchanged between these two nodes will be encrypted with such a key.

The bidirectional cover traffic guarantees the anonymity of the senders against a global adversary and traffic analysis attacks. Moreover, the bidirectional flow of traffic allows us to use, for the response, the same tunnel utilized to forward the request. In this case, each node crossed by the response adds a layer of encryption to each message by using a symmetric key shared with the initiator. When the initiator receives the response, it removes all the layers of the encryption.

In Tarzan, each node maintains a three-level hierarchy dynamic hash table (DHT) in which the nodes are inserted in a given position according to their IP addresses.

This table offers a $lookup$ function that, given a string as input, returns as output an IP address of a node of the network. Observe that the input can be any arbitrary string.

To select k mimics, each node a invokes the function $lookup^i(a.ipaddr)$ for $1 < i \leq k+1$ where $a.ipaddr$ represents the IP address of a.

The DHT offers two advantages. First, since the DHT is shared by all the nodes, mimic selection is publicly verifiable and then this prevents an adversary node from selecting more than k mimics. The second advantage is that the mimics for a node are randomly selected in different IP domains so that if an adversary controls an entire domain by generating a huge number of malicious nodes in that domain, it does not increase the probability that a malicious node of such domain is selected as a mimic.

To conclude this section, we discuss the anonymity degree achieved in Tarzan against a malicious node in the tunnel. This degree can be measured in terms of *anonymity set* that is the set of potential initiators of the traffic. The anonymity set size, besides depending on the number of mimics (i.e., *degree*) of the node, increases exponentially with the length of the tunnel.

5.3 Problem Formulation and Basic Approach

In recent years, we observed an increase in uplink traffic demand [206]. A lot of uplink-intensive applications emerged in different fields such as cloud-enabled ecosystem [256], IoT networks [164], sensor and actuator networks [77], and so on.

In this chapter, we address the problem of guaranteeing a measurable degree of anonymity in uplink-intensive applications. A solution could be to apply the Tarzan protocol, discussed in the previous section, in which cover traffic is adopted to offer anonymity guarantees against a global adversary. On the other hand, the cover traffic represents overhead which results in a waste of bandwidth and energy consumption.

In Tarzan, there are three main metrics to consider [74]: latency, the amount of cover traffic, and the size of the anonymity set. Often, the latency is a project constraint as well as the anonymity degree. Therefore, finding a solution that, under the same cover traffic level (that cannot be increased for the above reasons) and a fixed latency, offers a better anonymity degree than Tarzan represents an advancement of the state of the art.

Roughly speaking, we consider, as a measure of the cover traffic, the degree of the nodes. Indeed, the more links in the network the more cover traffic has to be generated. Moreover, Tarzan requires bidirectional cover traffic in each link, otherwise, a significant reduction of the anonymity set arises [98].

Therefore, a challenge could be to eliminate the bidirectionality of cover traffic while preserving the Tarzan-like approach. This is basically the purpose of our proposal.

In principle, unidirectional traffic could still be enabled in the Tarzan protocol just by rearranging node mimics in such a way that they form a cycle. Once mimics are so organized, we can build a tunnel as in Tarzan, but requiring that two adjacent nodes in the tunnel belong to a cycle. This way, the response can be routed by moving back, at each hop between two nodes, by traveling the entire cycle involving these nodes. Thus, no bidirectional traffic is needed.

This idea is sketched in Figure 5.1, in which the red lines represent the forward path and the green lines represent the cycles traveled by the response.

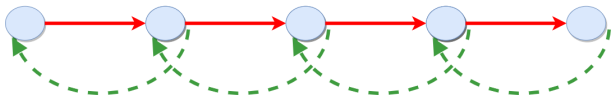


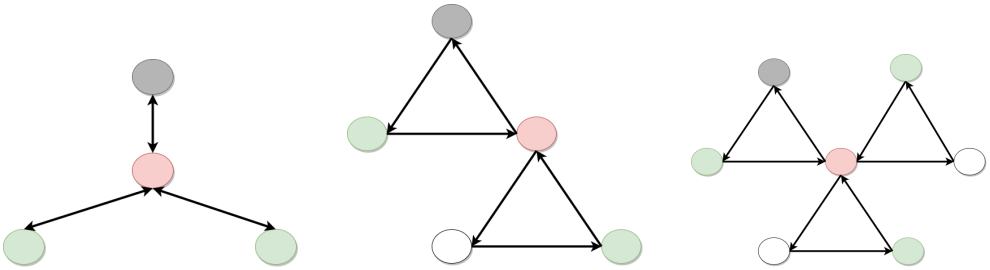
Fig. 5.1: Forward path (red arrow) and return path (green arrow)

However, there might be a price in terms of latency to pay when applying this cyclic approach, since, in general, the response would go through a longer path than the forward path. Instead, in Tarzan, forward and return paths are the same. Therefore, a solution based on the above idea is not trivially applicable.

The first immediate consideration is that it is convenient to minimize the size of cycles. Being Tarzan bidirectional links equivalent to 2-nodes cycles, the minimum dimension for non-trivial cycles is the case of 3-nodes cycles. On the other hand, it is intuitive to understand that no advantage can derive from having bigger cycles.

A much less clear point is to understand whether we have to pay a price also in terms of anonymity set.

This question derives from the following qualitative analysis.



(a) Tarzan topology with in-degree=out-degree=3

(b) Cyclic topology with in-degree=out-degree=2

(c) Cyclic topology with in-degree=out-degree=3

Fig. 5.2: Uncertainty at two hops

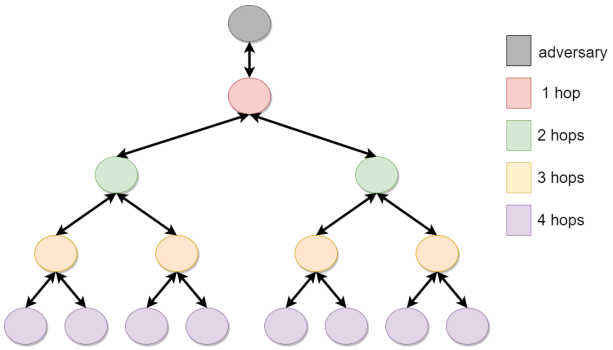


Fig. 5.3: Extension of figure 5.2a

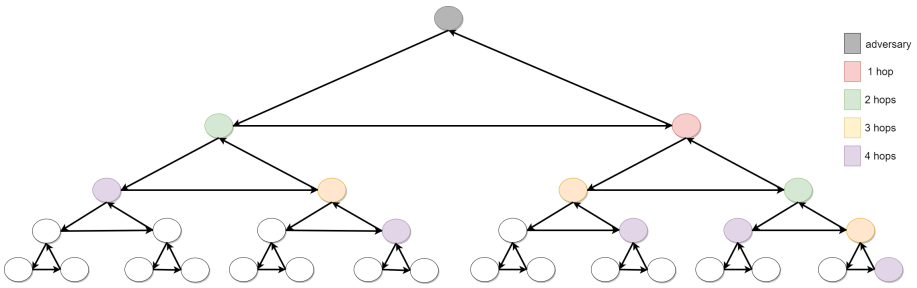


Fig. 5.4: Extension of figure 5.2b

We start by considering the uncertainty at two hops in the standard Tarzan topology and a two-hop equivalent topology in which cycles are enabled. This is represented in Figure 5.2. Specifically, in Figure 5.2a, we represent the standard Tarzan topology in which each node has three mimics. Suppose that the grey node receives a message from the red node. In this case, the candidate senders, at a maximum distance of two hops, are the red node and the two green nodes.

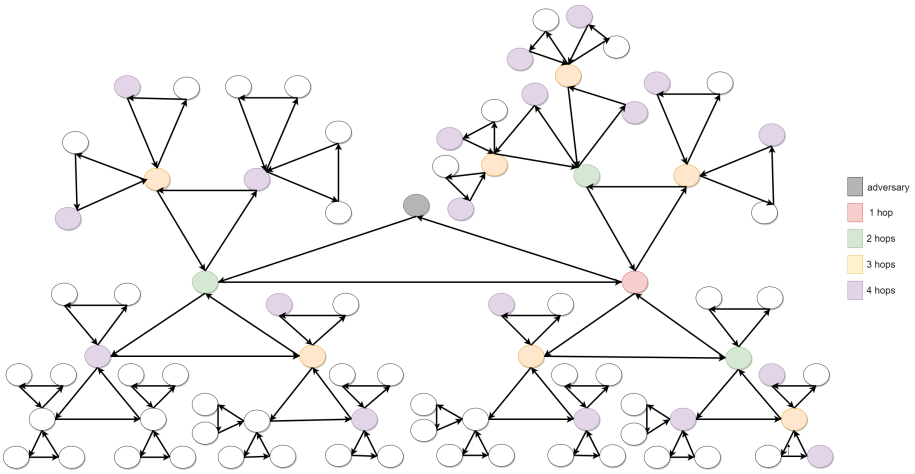


Fig. 5.5: Extension of figure 5.2c

The same uncertainty is obtained in the cyclic topology represented in Figure 5.2b in which, again, the candidate senders, at a maximum distance of two hops, are the red node and the two green nodes.

Regarding the cover traffic, we observe that in Figure 5.2a, we have three bidirectional links while in Figure 5.2b we have four unidirectional links, thus saving two unidirectional links. Therefore, it appears that keeping the same uncertainty, we have a significant reduction in the cover traffic.

Unfortunately, we can realize that the growth of the size of the anonymity set for the cyclic approach is slightly slower than that of standard Tarzan. We can understand this just by considering the case of tunnel length equal to four. To see this, we extend the topologies of Figures 5.2a and 5.2b, in Figures 5.3 and 5.4 respectively, to include tunnels with a maximum length of four hops.

In this case, the anonymity set of Figure 5.3 contains 15 nodes, while the anonymity set of Figure 5.4 contains 11 nodes.

Moreover, we have to take into account also the price in terms of latency required in the cyclic approach. However, the advantage in terms of cover traffic is maintained with respect to Tarzan.

Therefore, it is interesting to understand what happens if we compare the standard Tarzan with the cyclic version by considering two topologies that determine the same cover traffic.

The effect at two hops is highlighted in Figure 5.2c in which there are 6 unidirectional links equivalent to three bidirectional links of Tarzan. Therein, we can see that the candidate senders are the red node and the three green nodes. Therefore, the uncertainty at two hops is increased.

The extension to four hops of Figure 5.2c is represented in Figure 5.5. In this case, the anonymity set contains 30 nodes. Therefore, under the same cover traffic,

the cyclic approach offers a greater anonymity set size. However, the price in terms of latency still remains.

Clearly, in Tarzan, the latency depends only on the tunnel length. In the cyclic approach, it mostly depends on the tunnel length, and in a small measure also depends on the node degree (as explained in Section 5.5). Moreover, the disadvantage of the cyclic version depends also on the balance between downlink and uplink traffic (the more the weight of the downlink, the more the disadvantage).

In fact, the price we pay in terms of latency is related to the downlink traffic for the return path, which is in general longer than the forward path.

Thus, the problem we want to study is the following: In the cyclic approach, can we reduce the tunnel length to reduce latency and still be able to have an anonymity set size greater than Tarzan?

If in general, the answer to this question could be negative, it is interesting to understand what happens when there is an unbalance between the quantity of uplink and downlink traffic.

As we will describe in the sequel of the chapter, the result we achieve is that for uplink-intensive networks, the above approach is definitely advantageous.

5.4 C-Tarzan

In this section, we propose a new protocol, called *Circular Tarzan (C-Tarzan)*, based on the cyclic approach introduced in the previous section.

The idea is to extend the standard Tarzan protocol described in Section 5.2 by moving from bidirectional links to unidirectional links. This is possible if the response is routed through cycles to which mimics belong. As discussed above, we consider cycles of three nodes to minimize the price in terms of latency.

To build the cycles among mimics nodes, we design a new mimic selection algorithm that differs from that of Tarzan.

We assume that the same Tarzan DHT table (with the $lookup$ function) is used in C-Tarzan for the mimic selection.

Each node a chooses k' mimics through the $lookup$ function (see Section 5.2) as in Tarzan. Specifically, a selects $b_i = lookup^i(a.ipaddr)$ for $1 < i \leq k' + 1$. Each chosen mimic b_i can verify the correctness of the selection. Anyway, differently from Tarzan, a unidirectional link directed from a to b_i is established. At this point, each b_i will choose a mimic $c_i = lookup^i(a.ipaddr||b_i.ipaddr)$ and a unidirectional link directed from b_i to c_i is established. Observe that since the function $lookup$ accepts any arbitrary string as input and returns an IP address of a node of the network, it is guaranteed that the node c_i always exists in the network. c_i can verify the correctness

of the mimic selection started by a, involving the node b_i. Finally, to close the cycle, a unidirectional link is established from c_i to a.

It is easy to realize that each node has on average $6k'$ mimics. Indeed, each node A *selects directly* k' mimics $B_1,\dots B_{k'}$ to build k' cycles. In each cycle involving the node B_i, there will be a node C_i that establishes a link with A to close the cycle. Then, A will have further k' mimics $C_1,\dots C_{k'}$, resulting in a total of $2k'$ mimics. At this point, on average, A *is selected directly* by k' nodes to build further k' cycles. This leads to further $2k'$ mimics for A. Finally, on average, A *is selected indirectly* by k' nodes that in turn are selected directly by other k' nodes to build cycles. As before, this results in further $2k'$ mimics for A. Therefore, since unidirectional links are established between pairs of mimics, each node has, on average, $6k'$ unidirectional links ($3k'$ outgoing and $3k'$ ingoing).

We recall that, in Tarzan, if a node selects k mimics, it has, on average, $2k$ mimics and then $2k$ bidirectional links corresponding to $4k$ unidirectional links. Therefore, by considering the number of links as a measure of cover traffic, we have that, to obtain the same level of cover traffic in Tarzan and C-Tarzan, we have to set k' such that $6 \cdot k' = 4 \cdot k$ i.e., $k' = \frac{2}{3} \cdot k$.

At this point, we discuss how the messages are forwarded anonymously to the destination and the latter can reply to the initiator.

As in Tarzan, we assume that a symmetric hop-by-hop key is exchanged preliminarily between mimics.

To enable communication, we need to redefine the entire building process of the tunnel. Specifically, the initiator a selects, as first relay, one of its *outgoing* mimics b_i, i.e., a mimic b_i such that a directed link from a to b_i exists. Similarly to the standard Tarzan protocol, a needs the set of the (outgoing) mimics of b_i and to exchange a symmetric key with b_i. Anyway, since the link between a and b_i is unidirectional, a reply cannot be sent directly from b_i to a, because it would be not covered by dummy traffic.

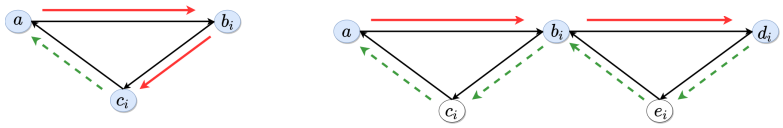
Therefore, to enable the reply, we define the function $C.next$ that can be invoked by a node C. This function receives as input a node B and returns as output the node A, such that there exist: (i) a direct link from B to C, (ii) a direct link from C to A, (iii) a direct link from A to B. Observe that, the *next* function leverages the fact that each node locally stores all the cycles it belongs to. Therefore, for a node C, given a node B as input, it is straightforward to compute the next of the node C (i.e., $A = C.next(B)$) in the cycle $BCAB$.

Then, b_i encrypts the response for a by using the hop-by-hop key exchanged with a and forwards this message to $c_i = b_i.next(a)$. This encrypted message is encrypted, in turn, by b_i with the hop-by-hop key exchanged with c_i. At this point, c_i decrypts

the message, invokes the function *next* to retrieve $a = c_i.next(b_i)$, encrypts the message again with its hop-by-hop key exchanged with a, and forwards it to a. Observe that, even though c_i knows that some real traffic has to be forwarded to a from b_i, c_i does not know the content of it, and then it has no more information than b_i about the fact that a is the actual initiator or just an intermediate node of the tunnel.

Once obtained the outgoing mimics of b_i, a selects a new mimic among them, say d_i, and needs to exchange a symmetric key and the set of outgoing mimics of d_i. Now, two cases may occur. The first case is that $d_i = c_i$ i.e., a, b_i, d_i are in the same cycle and d_i coincides with c_i. In this case, the list of mimics of c_i can be communicated directly through the link between c_i and a.

The second (complementary) case occurs when d_i has no common cycle with a. In this case, the list of mimics has to be forwarded from d_i to a_i through b_i. To enable the communication between d_i and b_i, since no direct link exists from d_i to b_i, we apply the approach discussed above. Specifically, d_i forwards this list through another node $e_i = d_i.next(b_i)$.



(a) Second relay in the same cycle of the initiator

(b) Second relay in a different cycle from the initiator

Fig. 5.6: Second relay selection

These two cases are represented in Figures 5.6a and 5.6b, respectively. Therein, we represent by a red arrow the forward communication between the initiator and the second relay of the tunnel, and by a green dashed arrow the backward communication from the second relay to the initiator.

The building of the tunnel proceeds iteratively until the last node.

Once the tunnel is set, the initiator can communicate with the recipient through this tunnel as in the standard Tarzan protocol.

Regarding the response by the recipient, the approach used to enable the exchange of information between a node of the tunnel and a previous node is applied. Specifically, at each hop of the tunnel starting from the last node until the initiator, if a direct link exists between a node and a previous node of the tunnel, then the response is directly forwarded through this link, otherwise, the response is forwarded through an intermediate node.

Some more detail will be discussed in Section 5.5.

5.5 Latency in Tarzan and C-Tarzan

In the previous sections, we mentioned that our solution introduces a price in terms of latency, assuming the same cover traffic and the same tunnel length in Tarzan and C-Tarzan. To give an answer to the question of Section 5.3, we have to quantify this price.

To perform an analytic analysis, we use as a measure of this metric the number of hops traveled by a message in the forward path and in the return path.

We introduce the following notation. We denote by τ the average delay of the links of the network. We start by evaluating the latency for Tarzan. We denote by h the tunnel length of Tarzan and by L_f and L_r the latency of the forward path and the latency of the return path of Tarzan, respectively.

Since the same tunnel is used both for the request and the response, it is easy to see that $L_f = L_r = (h+2) \cdot \tau$, where the term 2 derives from the fact that there is one hop between the last node of the tunnel and the PNAT and one hop from the PNAT and the destination.

Consider now C-Tarzan. We denote by h' the tunnel length and by L'_f and L'_r the latency of the forward path and the latency of the return path, respectively.

For the forward path, no difference with Tarzan exists and then $L'_f = (h'+2) \cdot \tau$.

On the other hand, for the return path, it is not trivial to estimate the number of hops in the return path, since it depends on the tunnel construction.

We can provide an approximation of the return latency representing an upper bound of its actual value.

The two cases of Figures 5.6a and 5.6b have to be considered. In particular, consider the selection of the first two relays of the tunnel. If the second relay is in the same cycle as the initiator (case a), then the response goes directly from the second relay c_i to the initiator a and this means that two hops in the forward path correspond to just one hop in the return path.

In case (b), the second relay d_i belongs to a different cycle and then the response goes from d_i to the first relay b_i, through an intermediate node e_i (2 hops) and, then, from b_i to the initiator a, through another intermediate node c_i (again, 2 hops).

In other words, for the first two hops of the forward phase, if case (a) occurs, then the response requires one hop, otherwise (case (b)), the response requires four hops.

It remains to estimate the probability that cases (a) and (b) occur.

To do this, we denote by $d = 3k'$ the average number of outgoing mimics of a node.

Obviously, since the mimics are selected uniformly at random, case (a) occurs with probability $\frac{1}{d}$, and case (b) occurs with probability $\frac{d-1}{d}$.

So far, no approximation has been introduced.

If we assume that the third relay of the tunnel is selected in a different cycle than the first relay (it happens with probability $\frac{d-1}{d}$), we can apply the reasoning followed for the first two relays to the third and fourth relays. Therefore, to find an approximation, we neglect the event that the relays in an odd position i of the tunnel are selected in the same cycle of the relay in position $i-2$ (it happens with probability $\frac{1}{d}$ at each choice).

Under this hypothesis, we have that, for every two hops in the forward phase, if case (a) occurs, then the response requires one hop, otherwise (case (b)), the response requires four hops.

It is easy to realize that, if such a hypothesis is not satisfied, then the response requires a lower number of hops and then, our approximation represents an upper bound of the actual latency.

Therefore, for h' even, the latency of the return path of C-Tarzan results: $(\frac{h'}{2} \cdot (\frac{1}{d} \cdot 1 + \frac{d-1}{d} \cdot 4) + 2) \cdot \tau = (h' \cdot (2 - \frac{3}{2 \cdot d}) + 2) \cdot \tau$.

On the other hand, for h' odd, the latency results: $((h'-1) \cdot (2 - \frac{3}{2 \cdot d}) + 4) \cdot \tau$.

By considering equally likely the events that h' is odd and h' is even, we conclude that the return latency for C-Tarzan is: $L'_r = (h' \cdot (2 - \frac{3}{2 \cdot d}) + \frac{3}{4 \cdot d} + 2) \cdot \tau$.

Observe that L'_r increases as d increases. This is due to the fact that, as d increases, the probability that a mimic of the tunnel is selected in a different cycle increases. Then, the response requires more hops, and the return latency increases.

5.6 Experiments

Through this section, we perform an experimental validation of C-Tarzan by highlighting the conditions under which it outperforms the standard Tarzan protocol.

5.6.1 Metrics and Experiment Setting

As already introduced, we consider three metrics: cover traffic, latency, and anonymity set size.

Regarding the cover traffic, we use as a measure the number of ingoing and outgoing links of the nodes, by considering that every link concurs, on average, with the same portion of cover traffic. As discussed in Section 5.4, to obtain the same cover traffic in Tarzan and C-Tarzan, we have to set

$$k' = \frac{2}{3} \cdot k \tag{5.1}$$

Regarding the latency, as seen in Section 5.5, to obtain the same total latency (forward latency plus return latency) we need to set h' such that $L_f + L_r = L'_f + L'_r$ i.e.,

$h' = \frac{2h - \frac{3}{4 \cdot d}}{3 - \frac{3}{2 \cdot d}}$. However, since we are interested in studying what happens when the balance between uplink and downlink traffic varies, we introduce two coefficients w_f and w_r, such that $w_f + w_r = 2$, to associate with the forward latency and the return latency, respectively. For example, $w_f = w_r = 1$ represents balanced traffic between uplink and downlink, while $w_f = 2$ (and $w_r = 0$) represents only uplink traffic.

Therefore, the condition to satisfy is $w_f \cdot L'_f + w_r \cdot L'_r = w_f \cdot L_f + w_r \cdot L_r$, that leads to

$$h' = \frac{2 \cdot h - \frac{3}{4 \cdot d} \cdot w_r}{w_f + \frac{4 \cdot d - 3}{2 \cdot d} \cdot w_r} \tag{5.2}$$

Now, we denote by $AS(k,h)$ the size of the anonymity set of Tarzan obtained as a function of k and h. Furthermore, we denote by $AS'(k',h')$ the size of the anonymity set of C-Tarzan obtained as a function of k' and h'.

Thus, the question now is whether, by setting k' and h' as in equations 5.1 and 5.2, respectively, it holds that AS' is greater than AS. If this is the case, then our approach introduces an advantage with respect to Tarzan.

Due to the complexity of retrieving the analytical formulas for AS and AS', we do this by simulation, leaving the analytical study as future work.

Furthermore, in order to obtain realistic results, we do not use directly the upper bound provided by 5.2 (see Section 5.5), but we find experimentally the values of h and h' leading to the same latency for Tarzan and C-Tarzan, respectively (actually, verifying the results obtained in Section 5.5).

To summarize, we find the values (h,k,h',k') that satisfy the following system.

$$\begin{cases} k' = \frac{2}{3} \cdot k \\ w_f + w_r = 2 \\ w_f \cdot L'_f + w_r \cdot L'_r = w_f \cdot L_f + w_r \cdot L_r \\ AS' \geq AS \end{cases} \tag{5.3}$$

In detail, the simulation has been performed in JAVA as follows. We considered a network of 100,000 nodes. First, we set some values of w_f (and, then, $w_r = 2 - w_f$), k', and h' for C-Tarzan and, then, we generated a topology (the links are obtained considering that each node selects *directly* k' mimics to build cycles).

On this topology, we measured the average degree of each node counting both the actual ingoing and the outgoing links (cover traffic), the actual number of hops that a request and the corresponding response have to cross on a path of height h' (a measure of latency), and the size of the corresponding anonymity set.

We repeated the experiment with the same parameters for 100 rounds (by varying the topology) to obtain steady results.

At this point, by the first equation of the system (5.3), we set $k = \frac{3}{2} \cdot k'$. Then, by using the value $w_f \cdot L'_f + w_r \cdot L'_r$ obtained experimentally for C-Tarzan and by recalling that $L_f = L_r = (h+2) \cdot \tau$, by the second and third equations of the system (5.3), we found the proper value of $h = \frac{w_f \cdot L'_f + w_r \cdot L'_r - 4 \cdot \tau}{2 \cdot \tau}$.

Then, we performed again 100 rounds of simulation with k, h to measure the cover traffic, latency, and anonymity set of Tarzan.

We confirmed that the obtained values of cover traffic and latency are the same as C-Tarzan (with an error of less than 1 % for both). Therefore, we obtain an experimental validation of the fact that the first three equations of (5.3) hold. We discuss the results regarding the anonymity set size in the next section.

5.6.2 Results

In this section, we compare Tarzan and C-Tarzan in terms of anonymity set size, by setting the same cover traffic and same latency.

In the first analysis, we show as the anonymity set size of both protocols varies as the cover traffic increases. We plot in the y-axis the ratio between the size of the anonymity set of C-Tarzan AS' and the size of the anonymity set of Tarzan AS. In the x-axis, we consider the degree d representing the number of outgoing (or ingoing) links in C-Tarzan (as defined in Section 5.5) that is equal to the number of bidirectional links in Tarzan (to obtain the same cover traffic).

The results of this analysis are reported in Figures 5.7,5.8,5.9, for different values of h' and w_f.

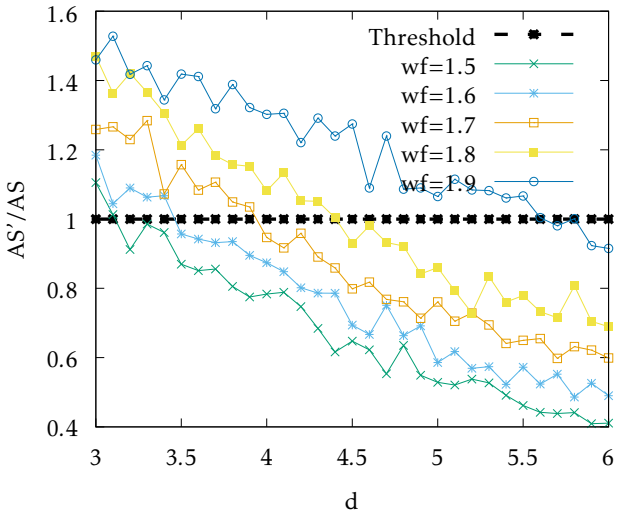


Fig. 5.7: Anonymity set ratio vs cover traffic d with h'=3

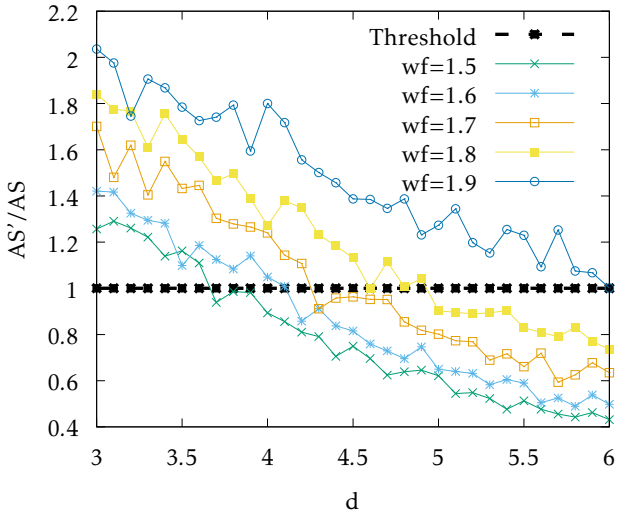


Fig. 5.8: Anonymity set ratio vs cover traffic d with h'=4

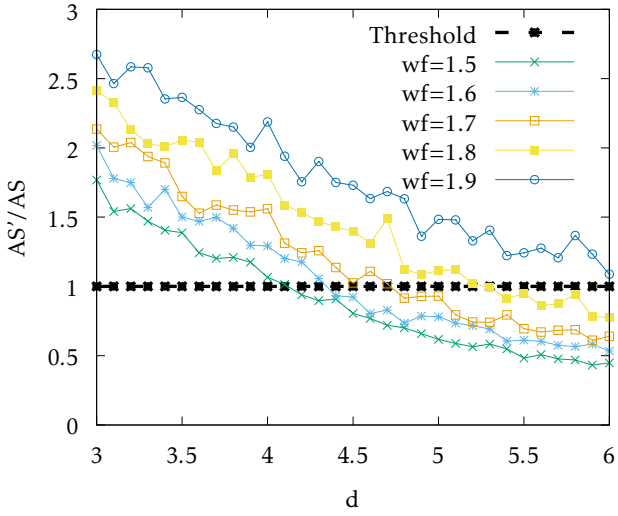


Fig. 5.9: Anonymity set ratio vs cover traffic d with h'=5

We represent with a dashed black line the ratio equal to 1. When the plots exceed this line, C-Tarzan outperforms Tarzan (in terms of anonymity set size).

We observe that our performance (for a fixed h') decreases as d increases.

This happens because, as d increases, the latency of C-Tarzan increases, then the tunnel length of Tarzan h (that offers the same latency of C-Tarzan) increases too. Therefore, the anonymity set size of Tarzan increases.

Even though the anonymity set size of both the protocols has a polynomial growth with d, the exponential growth of the anonymity set size of Tarzan with h is dominant. Therefore, as d increases, the ratio between AS' and AS decreases.

Regarding w_f, as it increases (by considering the same d), the performance of C-Tarzan increases. This happens because an increasing weight w_f represents pre-

dominant uplink traffic that leads to lower total latency for C-Tarzan (since the return path is longer than the forward path). This implies that the tunnel length h of Tarzan, which offers the same latency, decreases, and then AS decreases too.

As a final consideration, we observe that, until a certain level of cover traffic (corresponding to some d), it is advantageous to employ the C-Tarzan protocol, while when this threshold is exceeded, Tarzan is more convenient. Moreover, in the condition of increasing uplink traffic, this threshold also increases by making C-Tarzan suitable within a higher range of cover traffic level.

Observe that lower values of d are desirable since they represent cover traffic injected in the network. On the other hand, the reader might ask whether lower values of d result in acceptable anonymity set size in absolute terms (in relative terms C-Tarzan outperforms Tarzan). The response is affirmative, indeed as we discuss in the sequel, the anonymity set increases exponentially with h and h'. Then, with a small increment of h', we are able to obtain a good anonymity set size still outperforming Tarzan. Just an example, with $d = 4$ and $h' = 4$, we obtain an anonymity set size of about 100.

We conclude this section, by showing as the performances of C-Tarzan vary with respect to Tarzan as h' varies.

The plot in Figure 5.10 shows AS and AS' as h' varies with two different values of w_f and $d = 4$.

As expected, AS' increases exponentially with h'. Moreover, when h' increases, h increases too (to offer the same latency), and then also AS increases exponentially.

Observe that AS' with $w_f = 1.5$ is essentially (modulo experimental error) the same as AS' with $w_f = 1.9$. Indeed, AS' does not depend on w_f.

On the contrary, h depends on the total latency of Tarzan, which is equal to the total latency of C-Tarzan that, in turn, depends on w_f. Therefore, as w_f increases, h decreases and AS decreases too.

To conclude this section, in Figures 5.11, 5.12, and 5.13, we show the ratio between the anonymity set of Tarzan and C-Tarzan as h' varies for different values of w_f and d.

According to the previous analysis, C-Tarzan outperforms Tarzan for low d and for increasing w_f. Regarding h', we observe a fluctuating behavior in which there are some ranges of h in which there is an increasing trend of the ratio and other ranges in which there is an opposite trend. This is due to a compensation effect between the growth of the anonymity set size and the latency. In particular, for C-Tarzan, when h' increases, AS' increases, and the total latency increases too. Anyway, in some ranges, the increment of latency is limited. This leads to an increment of the tunnel length of

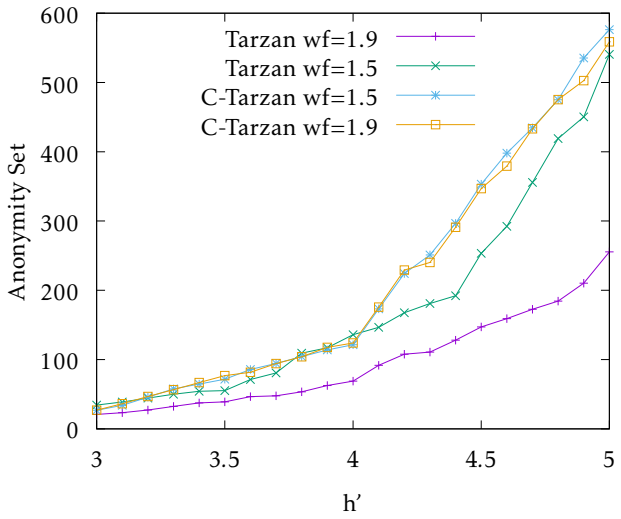


Fig. 5.10: Anonymity set vs h' with d=4

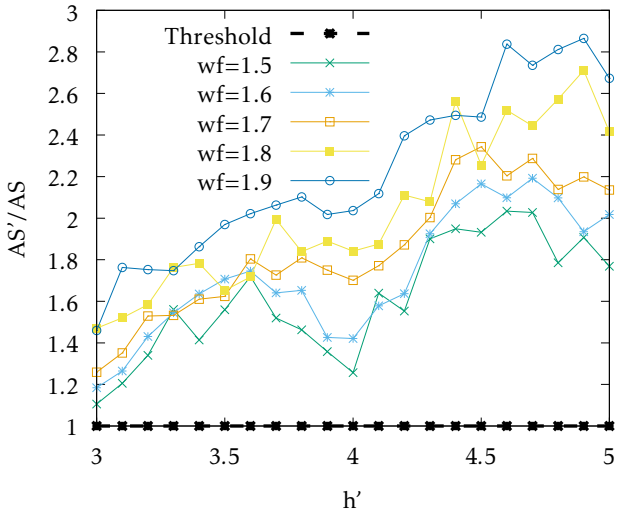


Fig. 5.11: Anonymity set ratio vs tunnel length h' with d=3

Tarzan h that is not sufficient to obtain an anonymity set size AS which compensates for the growth of AS'.

On the contrary, once h' reaches a peak value, the effect of the growth of the latency assumes a more relevant role by leading to values of h corresponding to anonymity set size AS able to compensate for the growth of AS'.

As a final remark, observe that, in this analysis, we show the advantage of our approach just in terms of anonymity set size (under the same latency and cover traffic level). Clearly, this advantage can be translated into an advantage in terms of latency or cover traffic, by fixing the same anonymity set size for both protocols.

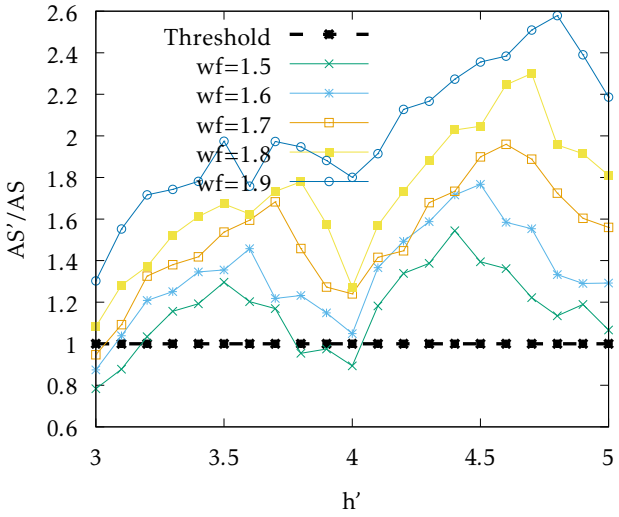


Fig. 5.12: Anonymity set ratio vs tunnel length h' with d=4

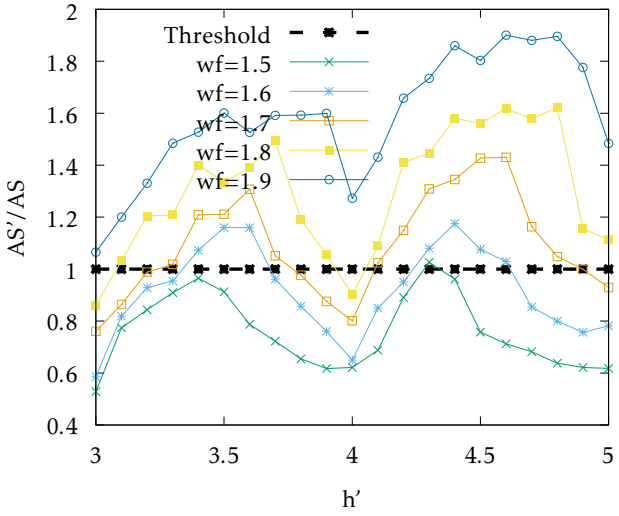


Fig. 5.13: Anonymity set ratio vs tunnel length h' with d=5

5.7 Related Work

Anonymous Communication Networks (ACN) [284, 245] are networks in which users are provided with anonymity services protecting their privacy. An ambitious goal to achieve is to offer anonymity guarantees against passive eavesdroppers (including a global adversary) and malicious participants. As stated in [71], to achieve this goal, dummy traffic needs to be injected into the network to hide the actual traffic.

In the literature, three main approaches leveraging dummy traffic are available. The first is based on *buses* [124, 21, 296]. In this solution, a predetermined route is used by the sender to anonymously communicate with the destination. However,

this technique is not scalable on a large network, since it requires an Eulerian path passing through all the nodes which leads to a prohibitive cost in terms of latency.

A second approach is represented by *DC-Nets* [58], which offer cryptographic guarantees of anonymity, but they suffer from scalability problems as buses [245].

The third approach is represented by the *mixnets* [59, 158, 22] which, in general, offers a lower latency with a price in terms of cover traffic.

Some recent mixnet proposals exist [158, 273, 223, 269]. Anyway, some drawbacks should be taken into account.

For example, as recently stated in [6], the work proposed in [158] suffers from very large communication overhead. Regarding [273], as stated by the authors themselves, the end-to-end latency is about 37 seconds, which may be too high for several applications. Moreover, these approaches rely on a server-oriented architecture, which is known to be less robust against possible attacks on the nodes of the route (or their collusion) [243] and less scalable than P2P architecture [245].

Therefore, the state of the art of P2P approach for low-latency applications is represented by Tarzan [98], which is a work with high impact in the (even current) scientific literature.

Actually, another P2P mixnet proposal, less recent but adopted in practice, is I2P [299]. However, it suffers from different vulnerabilities such as brute-force attacks or timing attacks. Then, as reported on the official website (`geti2p.net`), the authors suggest adopting some mitigations (e.g., constant-rate cover traffic) present in [98].

The work presented in this chapter strongly refers to [98], by proposing an extension improving Tarzan in the case of uplink-intensive applications. To the best of our knowledge, no proposal outperforming Tarzan is available in the state of the art. On the other hand, the considered domain is relevant. Indeed, uplink-intensive applications are becoming more and more common in recent years [206, 289]. Some examples of uplink-dominant applications are represented by M2M [197, 55], Industrial IoT [164], and Wireless-Sensor-Network [77]. Furthermore, intrinsically, cloud-based applications increase the uplink bandwidth demand with respect to traditional client-server applications [256].

Part II

Anonymous communication over an existing application layer

The protocols presented in Part I have the advantage of being independent of the application layer implemented over them. However, they may require heavy infrastructural changes (e.g., the format of the packets) in the underlying network.

Furthermore, there are situations in which anonymity features have to be achieved within applications offering other types of services.

For example, we can consider a social network providing all the traditional social features along with anonymity services. Since these anonymity services may leverage these social features (e.g, communicating users do not know each other but share some interests), they have to be delivered through the communication channels offered by the social network itself (as currently happens in Facebook or Instagram). In these cases, setting external anonymous communication channels is not practical or is not possible at all.

Therefore, it is relevant to design anonymous communication protocols working only on specific application layers that require minimal network infrastructural changes. This is precisely the purpose of this second part of the thesis.

We point out that by designing a protocol ad-hoc for a specific application layer, we can obtain benefits in terms of performance by leveraging specific characteristics of the application itself. For example, in Chapter 6, just a few asynchronous messages have to be exchanged and latency is not a crucial issue. This allows us to design a protocol in which cover traffic is reduced. It corresponds to a reduction of CPU and bandwidth overhead required by users that often rely on smartphones when using the services.

In this part, we propose two protocols.

The first is presented in Chapter 6. Therein, we present an anonymous communication protocol enabling three communication primitives within a social network. The first primitive allows communication between an anonymous sender and an explicit recipient (possibly coinciding with the social network provider). The second primitive enables the response to the first primitive. Finally, the last primitive enables communication from an anonymous sender to an anonymous recipient. The

strong point of this solution is that anonymity is achieved against a global adversary able to observe the entire traffic of messages exchanged in the social network. Actually, it is a necessary condition to obtain a working solution, since the social network provider represents a typical real-life example of a global adversary. An application of this solution to the domain of proximity-based services is also provided.

Another anonymous protocol built over an existing application layer is presented in Chapter 7. In this case, the application layer is MQTT and due to the low-energy consumption requirements, it is crucial that the involved entities exchange messages through this lightweight protocol without relying on external communication channels. In particular, our solution leverages the bridging mechanism natively offered by MQTT to deploy a network enabling anonymous publishing/subscribing to topics. The approach we follow is based on the Crowds protocol [230]. The experimental analysis performed shows that the overhead introduced by our approach with respect to the standard MQTT is negligible considering the throughput and acceptable considering the latency.

6

Anonymous communication in Social networks

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 chapter, we propose an anonymous communication protocol for short communications in social networks, based on a collaborative approach. The results of this approach are included in two research papers [37, 42].

6.1 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 a 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 [245], supporting low-latency and connection-oriented communication.

To the best of our knowledge, there exist a few proposals regarding anonymous communication in social networks [203, 148]. However, [203] only deals with anonymous group communication and [148] does not provide sender anonymity against the global passive adversary, which is the goal we pursue in this chapter.

Indeed, the aim of this chapter is to achieve communication anonymity (in the case of short asynchronous messages) in social networks, against the global passive adversary.

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 an anonymizer, has very limited effectiveness in our threat model, as shown in [276].

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) [71] and are based either on *mixnets* [98, 158, 23], or on *buses* [124, 21, 296].

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 [124, 21] or non-deterministic [296]) 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 the 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 field of proximity services, we can accept that proximity tests are performed within an order of magnitude of minutes.

6.2 Background: Identity-based Encryption

Our protocol leverages the *Identity-based Encryption (IBE)* [56] to encrypt messages. In this section, we provide some background notions about IBE.

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 [229]. 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 favor 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 [53, 33] satisfy our requirements.

6.3 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 an 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 an 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.

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

6.3.1 Identity management

In our solution, the way in which identities are managed is critical, because the goal 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 6.3.7).

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 6.2) 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 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, 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

[84], or a robust Self Sovereign Identity system [110], like that designed in the EBSI framework [271].

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 the 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 at 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 6.1

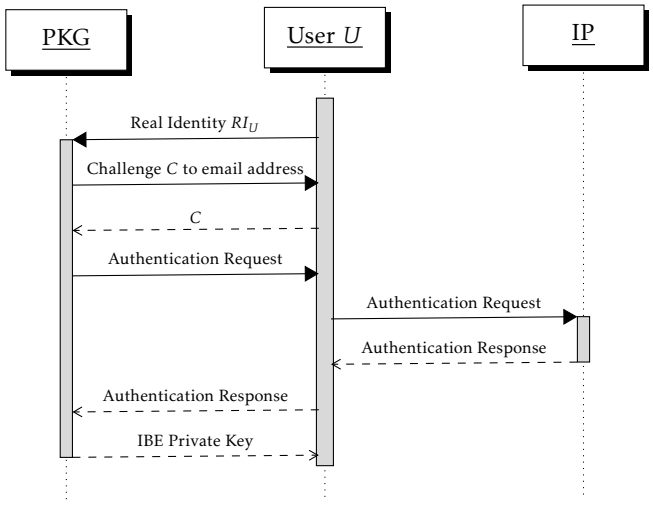


Fig. 6.1: eIDAS-based authentication procedure with PKG to obtain the IBE private key

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 6.3.7, 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 a 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).

6.3.2 Application domains

Anonymity is obtained through a cooperative approach, involving the users of SN. The collaboration is considered 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 6.1. *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 a 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 themselves 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 [178] and t-closeness [169].

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

6.3.3 The ring schema

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

We start with the definition of *ring schema*.

Definition 6.2. *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 to which x belongs. 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 6.3. An example of ring schema for $\alpha = 2$ and $k_A = 5$ (then $n_A = 10$) is reported in Figure 6.2. 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 $ring(9) = 1$.

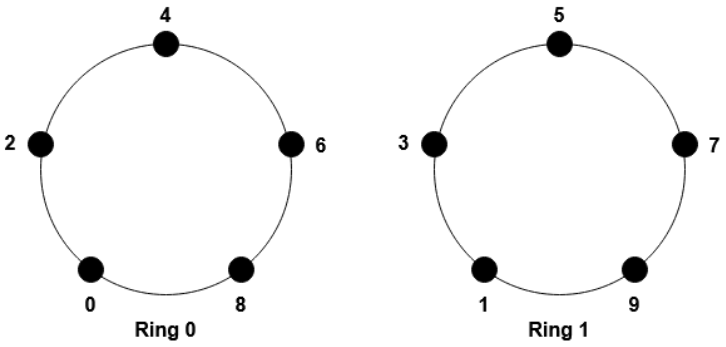


Fig. 6.2: Ring schema for $\alpha = 2$ and $k_A = 5$

From now on, throughout the chapter, 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 6.4. *A user mapping on the α-ring schema of A is any function $h : ID_A \to D_A$ 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 fulfill the condition required by Definition 6.4. 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.

Example 6.5. By referring to Example 6.3 (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 6.3.

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.

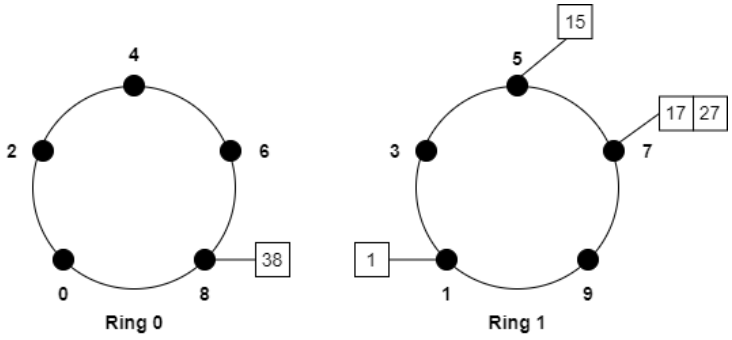


Fig. 6.3: Example of user mapping.

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

Example 6.7. By referring again to Example 6.3, $f_A(2) = 2+2$ **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 6.8. The hash table referred to the user mapping of Example 6.5 is depicted in Figure 6.4.

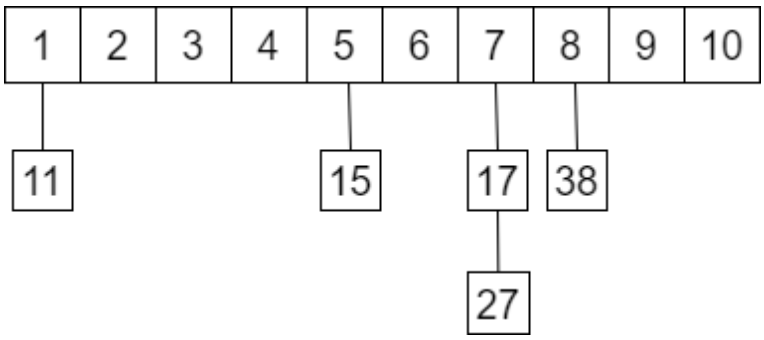


Fig. 6.4: Example of hash table referred to the user mapping in Figure 6.3

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 SI_X is mapped 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 to which the user belongs. 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 in the configuration of the ring is communicated by SN to all the users of the ring, as we will see in Section 6.3.5.

6.3.4 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 a 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 6.9. *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 6.10. *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*.

Example 6.11. Suppose $p = 0.99$ and $\tau = 0.999$. The redundancy level (see Definition 6.9) is $r_A = 1.5$. It is easy to see that the ring schema with the users mapped as in Figure 6.3 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 6.5. Here, $N_A = 19$, $k_A = 5$ and $\alpha = 2$.

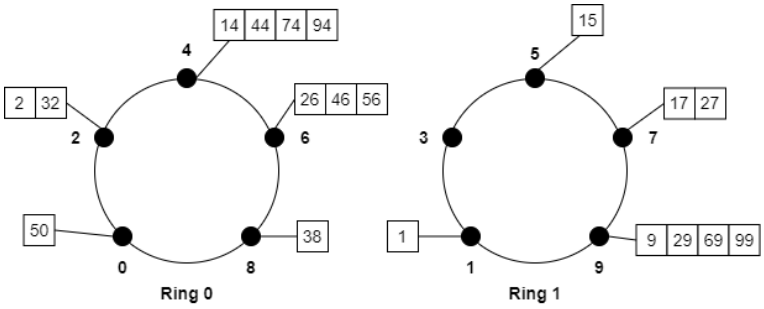


Fig. 6.5: Ring schema τ-safe.

6.3.5 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 6.12. We take the ring schema of Figure 6.5. 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 6.6

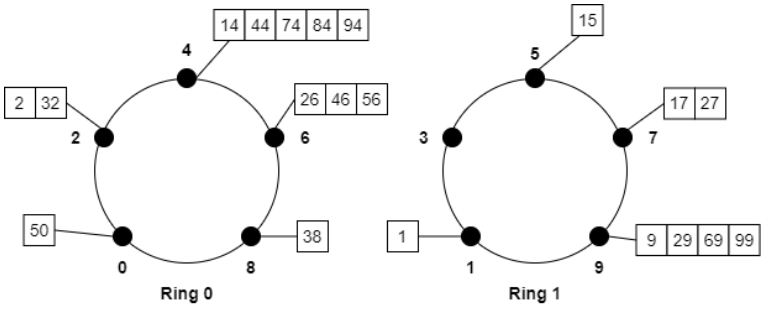


Fig. 6.6: Ring schema after the join of the user with SI 84.

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 that 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, 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 6.13. Consider the ring schema of Figure 6.6. 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 6.7

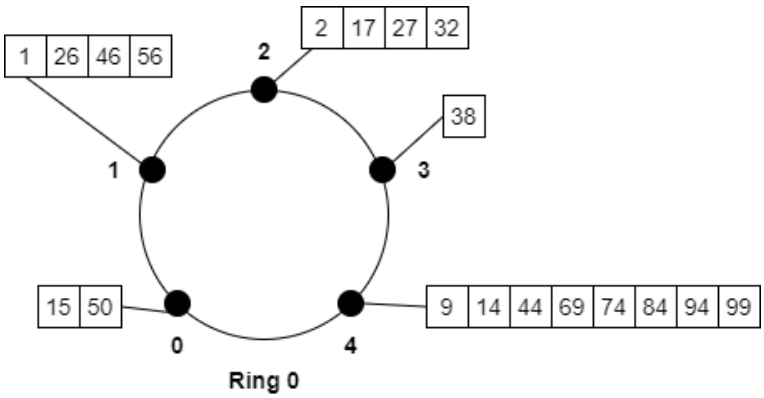


Fig. 6.7: New ring schema after the user with SI 29 leaves the ring 1 of Figure 6.5

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 in the system not corresponding to changes in 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.

6.3.6 Cover-message mechanism

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

Consider a ring $\{v_0, \dots v_{k_A-1}\}$. 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* live user for a given user in a ring.

Definition 6.14. *Given a node v of a ring with at least one live 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 live 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 ,live 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 live user in $h(SI_X)$ with SI higher than SI_X) to the closest node according to the function f_A with live users and, therein, by taking the user with the lowest SI.

According to Definition 6.14, the token is sent by a live 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 live 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 public-key encryption, we mean the above procedure.

In Figure 6.8, an example of a 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 live users, while red circles denote non-live users. The token turns in the ring according to the function *next* of Definition 6.14.

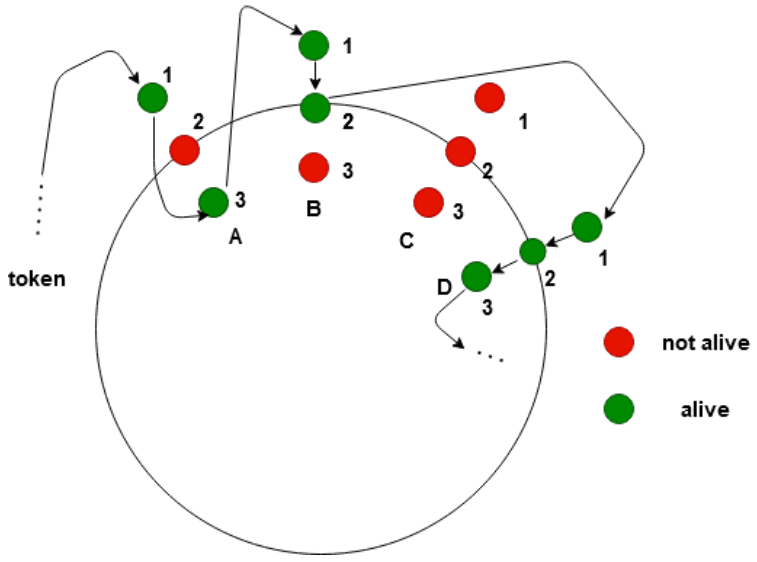


Fig. 6.8: Example of a fragment of the route of a token in a ring.

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 live 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 live 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.

6.3.7 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 the 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 6.6) 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 an 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.

6.4 Comparison with other approaches

After presenting the anonymous protocol, we show, in this section, that the definition of a new specific protocol represents actually an added value. In other words, by considering the attempt to apply existing approaches taken from the field of anonymous communication networks, we show why 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 [124, 21, 296] and mixnets [98, 59].

In deterministic buses [124, 21], the route is a path involving all the nodes of the network. This results in intolerable latency when, as in 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 6.5.3, 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 6.5.3, for a good privacy level (i.e., the cardinality of the anonymity sets) of 10^2 users, we obtain times of the order of magnitude of minute for a worst-case anonymous communication (including a number of exchanged messages).

Consider now non-deterministic buses [296]. 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 [296], 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 the order of magnitude 10^3) produces an average latency time of 20 minutes.

Regarding mixnets, we adopt a simplified yet general model extracted from [98], 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 the 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 6.9, 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 [98], bi-directional cover traffic should be injected, over all the links of the network. For simplicity, we assume

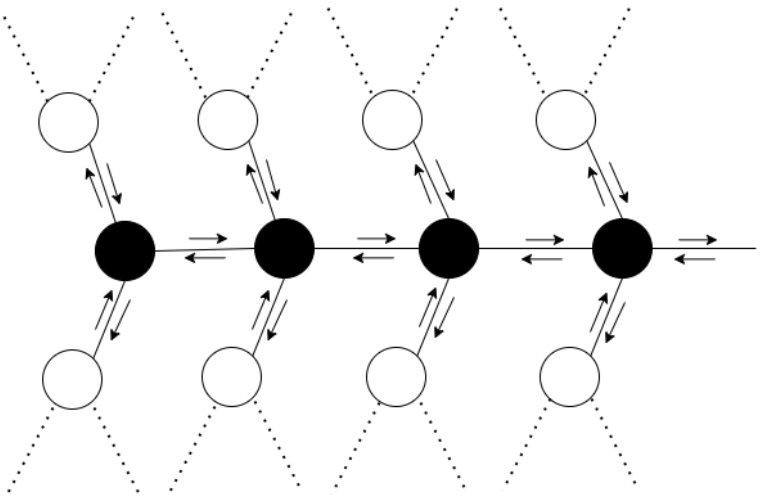


Fig. 6.9: Mixnet with $n = 4$ and $m = 2$.

.

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 6.9.

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 6.9, 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.

Regarding 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 6.5.3 shows that the

privacy-preserving services implemented on top of our anonymous communication protocols are performed in a reasonable amount of time.

6.5 Application to the proximity-based services: Prototype and Experiments

The protocol described in this chapter was applied in [42] to the domain of proximity-based services [181].

In this section, we describe the prototype we developed to validate our protocol. In particular, this prototype implements a service called `KN-Service` and described in [42]. This service leverages the three communication primitives described in Section 6.3.7.

We start by introducing the `KN-Service`.

6.5.1 `KN-Service`

The details about this service can be found in [42] and are outside the scope of this thesis. In this section, we describe just the interface of this service.

The `KN-service` (standing for *known nearby service*) 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*.

In particular, we consider two users knowing each other and 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.

To perform this test, some anonymous communications have to be performed on the social network. In particular, we need user-to-user communications (Primitive **P3**) and user-to-SN communications (Primitive **P2**).

6.5.2 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 [96] 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 and (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 6.3. Specifically, the client app forwards the token to its next in 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, fill it (if empty), and 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.

6.5.3 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 [38, 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 discussed in [42], 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.

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 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 6.10, 6.11, 6.12. 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 6.10, 6.11, 6.12, respectively.

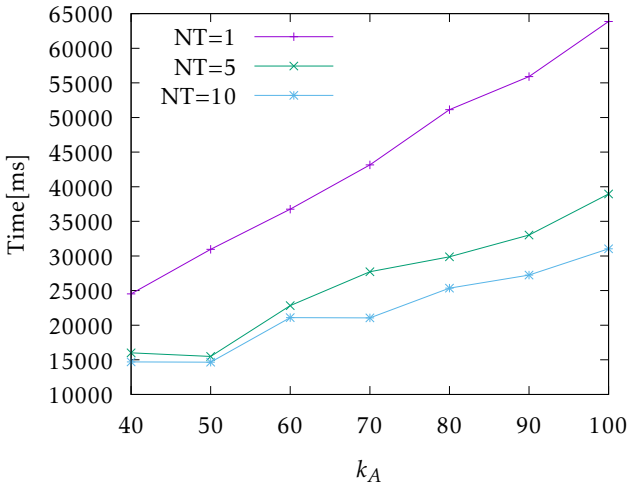


Fig. 6.10: Total proximity-testing time vs privacy level with $\sigma = 0$.

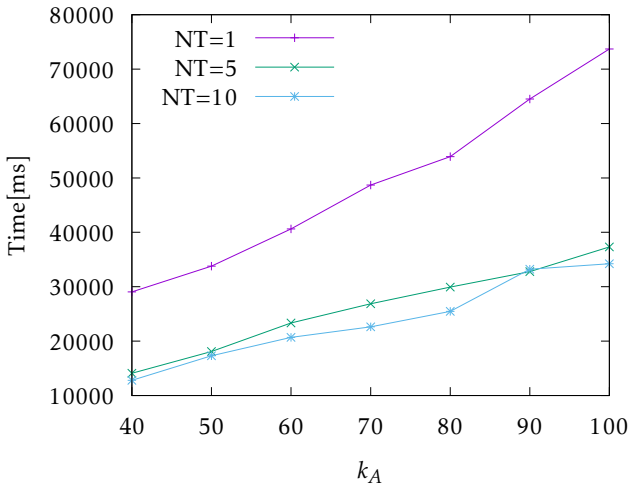


Fig. 6.11: Total proximity-testing time vs privacy level with $\sigma = 0.25$.

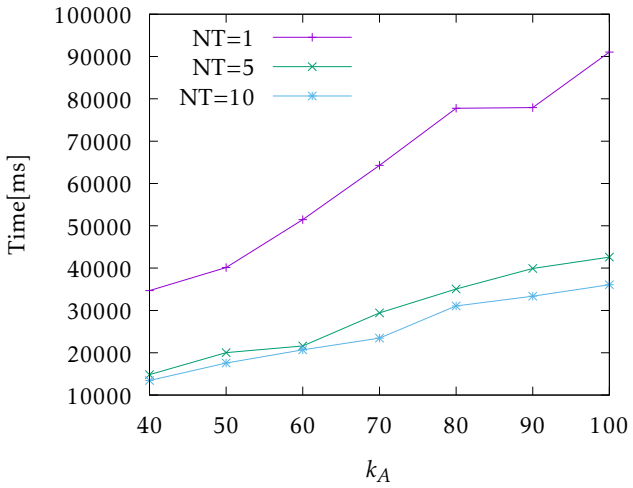


Fig. 6.12: Total proximity-testing time vs privacy level with $\sigma = 0.50$.

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).

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 6.13 the total time of a proximity test with a high privacy level ($k_A = 80$) for $NT = 5, 10$ and $\sigma = 0, 0.25, 0.5$.

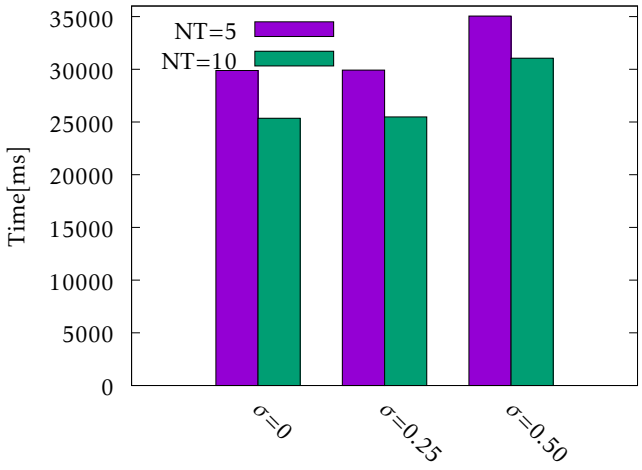


Fig. 6.13: Total proximity-testing time with $k_A = 80$.

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

6.6 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 [42]. **A2** is guaranteed by ring schema updating 6.3.5. 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.

The next three Theorems show how the three Primitives of Section 6.3 guarantee anonymity.

Theorem 6.15. *A user U invoking **P1** can be identified with probability not greater than $\frac{1}{k_A}$ (sender anonymity).*

Proof. 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 at 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 6.16. *A user U receiving a message through Primitive **P2** (as a response to Primitive **P1**) can be identified with probability not greater than $\frac{1}{k_A}$ (recipient anonymity).*

Proof.

This proof follows a similar reasoning to the proof of Theorem 6.15.

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 6.17. *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.

(1) can be proved as in Theorem 6.15. (2) can be proved as in Theorem 6.16.

This concludes the security analysis.

7

A Crowd-based approach to achieve anonymity in MQTT

MQTT is the most popular IoT protocol for communication of constrained devices. Since it is designed to be lightweight, security issues were not natively addressed in MQTT. While security aspects have been extensively studied in the literature, to the best of our knowledge, anonymity issues have received very little attention. In this chapter, we propose a new protocol, called MQTT-A, extending bridging-mode-MQTT to support anonymity of both publishers and subscribers. This task is accomplished through the P2P collaboration of intermediate bridge brokers, which forward the publish/subscribe requests of the clients so that the final broker cannot understand the actual source/destination. Moreover, an anonymity-preserving topic discovery mechanism is also provided. Importantly, all the MQTT-A messages are exchanged through the standard MQTT primitives and by leveraging the bridging mechanism natively offered by MQTT. This allows us not to require changes in the standard MQTT infrastructure to apply our solution, making it proposable from a practical point of view as well. The experimental validation shows that, from the performance point of view, we only pay a reasonable price in terms of latency. No significant impact on goodput occurs.

7.1 Introduction

Internet of things [11] is an evolving paradigm in which smart objects are connected to each other to deliver services. Since IoT devices can be resource-constrained, traditional communication protocols such as HTTP cannot be adopted to connect them. Therefore, researchers proposed new lightweight protocols allowing communication in scenarios in which limited bandwidth is available and energy consumption is a serious issue. MQTT [251] is the most popular protocol in the current IoT scenario. For example, Facebook Messenger uses MQTT for instant messaging [300].

Despite its large adoption, MQTT does not include built-in security. The main reason lies in the fact that complex security solutions require intensive CPU and memory usage, making them not applicable to constrained devices. Then, in the

literature, several solutions have been proposed to the trade-off between security and efficiency [244, 67, 272].

In this chapter, we deal with a specific security aspect, that is anonymity, for which very little attention has been devoted so far (to the best of our knowledge). Specifically, we aim to prevent the identification of publishers and subscribers when sending/receiving data, in every phase of the protocol, including the topic discovery. As a matter of fact, while anonymity in communication networks has been extensively studied for some decades, no specific solution exists in the domain of MQTT.

Therefore, a practical solution MQTT-compliant borrowed from ideas belonging to the field of anonymous communication networks [71] is highly desirable. This is just the contribution of this chapter.

Again, as in the previous chapters, we refer to three types of communication anonymity that we can reach: sender anonymity, recipient anonymity, and relationship anonymity.

In MQTT, publishers play the role of senders and subscribers play the role of recipients. Our solution reaches both sender and recipient anonymity, then relationship anonymity too. Furthermore, the proposed approach can be applied just from one side (publishers or subscribers) achieving just one between sender and recipient anonymity (but still enough to obtain relationship anonymity).

The problem with the standard MQTT approach is that clients communicate directly with the broker so that it is able to infer important information about them [134].

To solve this problem, we take advantage of the bridging architecture that MQTT offers. In this architecture, clients do not communicate directly with the broker but through an intermediate broker called *bridge broker*. The bridging mechanism implements natively a low level of anonymity since devices are not directly connected to the final broker hosting the topics. However, this is not enough. Indeed, the bridge broker simply forwards the requests of the clients without hiding some patterns (number of requests, topics of interest, and so on) sufficient for the re-identification of the clients.

The core of our proposal is to set up an anonymous peer-to-peer (P2P) network composed of the bridge brokers so that the final broker cannot discover which actual bridge broker has started the communication. Our solution takes inspiration from [230]. We chose this protocol since it is lighter than alternative solutions (e.g., [258]) and then, more suitable for IoT applications. Moreover, our solution is designed to be transparent to MQTT clients, leaving all the complexities to the brokers. This leads to two main advantages. First, clients can use our protocol regardless of whether they

implement standard MQTT or MQTT-SN [252]. Second, also resource-constrained clients can leverage our protocol.

The experiments performed in Section 7.7 show that an (acceptable) price in terms of latency has to be paid if both sender and recipient anonymity are desired (and then relationship anonymity too). However, if just one between sender and recipient anonymity is enough (maintaining relationship anonymity too), then the latency required by our protocol is halved. In terms of goodput, no appreciable difference exists.

To provide a complete and effective solution, we did not neglect an aspect not strictly regarding the MQTT communication, but still critically important with respect to anonymity. We refer to the problem of topic discovery, that is, how to allow clients to know which topics are available and which brokers host them. This aspect is not treated in the standard MQTT protocol [225, 153], in which it is assumed that clients know in advance topics and where to find them. This can be true for some applications where publishers can advertise the topics to which they send data or subscribers can advertise their interest in a given topic. However, when anonymity is desired, this is not true anymore. Then, a suitable mechanism has to be implemented.

To summarize, the main highlights of this chapter are the following:

- We provide an anonymity protocol supporting publisher and subscriber anonymous communication with a remote broker. We call this protocol MQTT-A(nonymous).
- We define a discovery protocol allowing the clients to know the available topics and the brokers hosting them.
- All the exchanged information used to implement the above protocols is provided through standard MQTT messages. This avoids infrastructural changes in the architecture and does not require particular effort for clients and brokers.

7.2 Background

MQTT is a client-server publish/subscribe messaging transport protocol [251]. Two kinds of agents are involved in the message exchange: MQTT clients and MQTT brokers. In turn, MQTT clients can be of two types: publisher (producer of information) and subscriber (consumer of the provided information). We stress that an MQTT client can play both roles of publisher and subscriber even at the same time. The MQTT protocol requires that the information provided by a publisher must be associated with a topic, which in general is used to categorize the information itself. Therefore, a subscriber can manifest that it is interested in receiving certain information by just specifying its topic.

A topic can present one or more levels separated by a forward slash ('/'). This way, topics can be organized in a hierarchical structure. Nevertheless, there is no standardized semantic model for MQTT topics, therefore the name of a topic can be chosen freely by publishing or subscribing entities [221].

Unlike the classical client-server architecture, in the MQTT architecture (Fig. 7.1), publishers and subscribers do not communicate directly between them but through an MQTT broker. When a publisher sends a message to the broker, it specifies the information and the topic under which that information should be published. Then, the broker forwards the published information to all the subscribers interested in that topic.

Concerning MQTT messages, they present a limited amount of overhead, since the protocol header is only 2 bytes. Moreover, MQTT limits the payload dimension up to a maximum size of 256 MB. These constraints make MQTT suitable for low-bandwidth networks and resource-constrained clients, such as IoT devices.

Furthermore, in both publish and subscribe messages, an MQTT client can specify the desired Quality of Service (QoS) level. MQTT supports three different Quality of Service (QoS) levels (0,1, and 2). In detail:

- **level 0** implies that messages are delivered *at most* once;
- **level 1** implies that messages are delivered *at least* once;
- **level 2** implies that messages are delivered *exactly* once.

Suppose a publisher sets QoS level QoS_p when it publishes to a topic t. Similarly, a subscriber chooses QoS level QoS_s when it subscribes to the same topic t. Two cases might occur: i) if $QoS_p > QoS_s$, then the broker forwards the data to the subscribing client using QoS_s; ii) if $QoS_p \leq QoS_s$, then the broker forwards the data to the subscribing client using QoS_p.

Another feature provided by the MQTT protocol is represented by the *retained messages*. A retained message is an MQTT message, with the retained flag set to true. When a broker receives a retained message labeled with topic t, it stores it for that topic. Thus, when a client subscribes to the topic t it will receive the retained message immediately after the subscription. A broker can store only one retained message per topic. Therefore, to update the retained message for a topic, it is sufficient for a publisher to send a new message, for that topic, with the retained flag set to true.

A core MQTT feature is represented by the bridging mechanism (Fig. 7.2) that allows two MQTT brokers to connect together so that they can share messages between them.

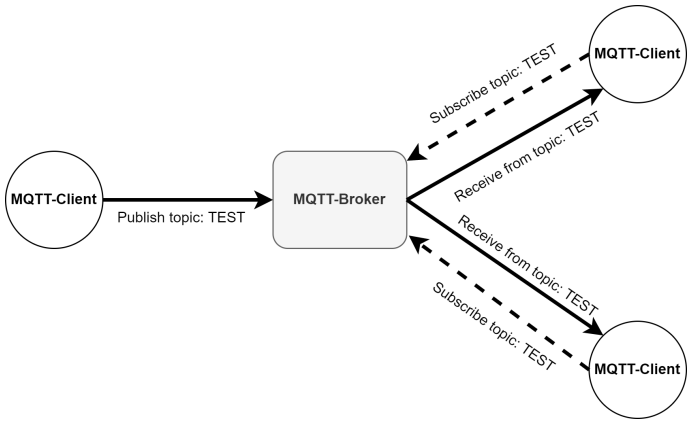


Fig. 7.1: MQTT architecture.

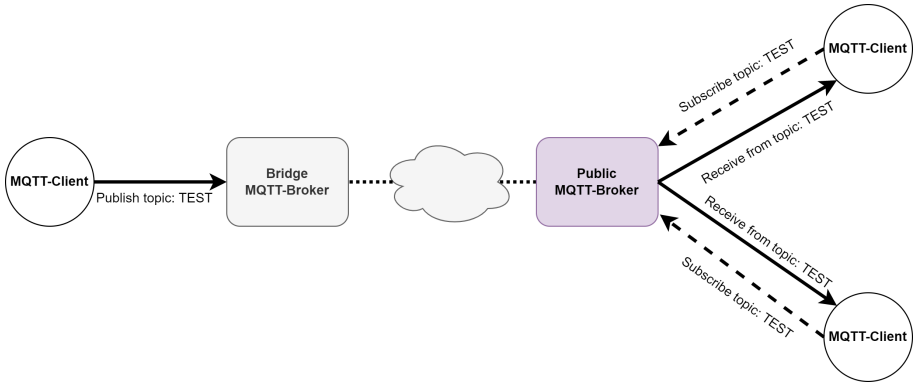


Fig. 7.2: MQTT bridging mechanism.

This configuration is adopted to connect an edge broker (which will act as a bridge) to a public broker. In such a way, the bridge broker will act as an MQTT client for the public broker, thus it can send a subscribe or a publish message to the latter.

Usually, clients are connected to the bridge broker through the local network, and it can decide on which local topics to apply the bridging mechanism. This way, not all the MQTT traffic locally generated is meant to be sent to a remote broker.

Moreover, MQTT allows the remapping of the local topics into public broker's topics. This procedure takes place in the bridge broker, in such a way that it is transparent to MQTT clients. It is worth noting that, through the bridging mechanism, information produced behind different brokers can be aggregated in a single place at a public broker, thus allowing the subscribers to recover it with a single connection.

7.3 Scenario and motivations

The aim of this chapter is twofold. First, we want to offer anonymity guarantees to both publishers and subscribers. Indeed, in a classical MQTT architecture, the

Table 7.1: Notations.

Symbol	Description
$\mathcal{B}_R$	set of the bridge brokers
$\mathcal{B}_P$	set of the public brokers
$\mathcal{B}_D$	set of discovery brokers
p	a generic publisher
s	a generic subscriber
$pub(T,I)$	publish message of the information I on the topic T
$sub(T)$	subscribe message on the topic T
B_R^p	bridge broker directly connected to the publisher p
B_R^s	bridge broker directly connected to the subscriber s
T_N	topic under which new topics available on a certain public broker are published
$T_{N'}$	topic under which new bridge brokers joining the network are published
T_S	topic under which the current set of all the available topics is published
$T_{S'}$	topic under which the current set of all bridge brokers is published
T_F	(*forwarding* topic) topic-prefix labelling information coming from remote bridge brokers
T_A	(*actual* topic) actual topic labelling an information
$REMOTE$	topic-prefix labelling information coming from MQTT clients

broker can observe which subscribers are interested in which topics and which publishers send messages to those topics.

The adoption of end-to-end confidentiality mechanisms (regarding messages and/or topics) is impractical since it would require a key exchange between MQTT clients. This is possible only in restricted scenarios in which publishers and subscribers are predefined and know each other.

Anyway, data-confidentiality-based solutions do not solve the anonymity problem. Indeed, the broker is still able to observe which publishers and subscribers are communicating. Therefore, attacks based on background knowledge about even one client can allow the broker to infer the topic and, as a consequence, information about all the clients communicating on this topic.

On the basis of the above considerations, in this chapter, we choose to achieve the privacy goals by hiding the identities instead of contents. If the identity of clients is really hidden, then the broker cannot link contents with clients. Privacy is then achieved. Consider that the trivial use of a pseudonym for the client is not enough because de-anonymization is always possible, also by taking into account that the source and the destination of the communication (i.e., the IP addresses) are quasi identifiers [68]. Therefore, we have to guarantee that both publishers and subscribers are entities that the broker is not able to identify in the network. This is a goal typically reached in the context of anonymous communication networks [71].

To accomplish the above objective, we refer to the MQTT bridging mode (see Section 7.2) in which clients interact (publish/subscribe) with a public broker through a bridge broker. This natively implements a low level of anonymity, in the sense that clients do not connect directly to a broker but are hidden behind their bridge brokers acting as proxies. However, it is not enough to achieve actual anonymity. First, also the identification of the local bridge could threaten privacy in the case in which the local bridge serves a restricted group of clients. Moreover, two other problems arise. The first is that it might happen that the bridge broker aggregates similar clients (in terms of interests). Therefore, the above-mentioned issues still occur. This problem is related to the concept of *l-diversity* [178]. In general, inference-based attacks are still possible.

The idea we follow in this chapter is to hide the identity of the clients by making anonymous and not identifiable the local bridge from/to which the communication comes/goes, To do this, we implement, in Section 7.5, an anonymity peer-to-peer protocol inspired by [230], in which the peers are represented by the bridge brokers.

To make effective the approach, we also provide a protocol allowing the clients to discover which topics are offered by which public brokers. This is an important aspect to take into consideration even when anonymity features are not required. Indeed, MQTT does not implement any native mechanism to accomplish this task [225]. Furthermore, this becomes a fundamental pillar when publishers remain anonymous. Indeed, being anonymous, a publisher cannot otherwise advertise the topics on which it will publish.

To conclude this section, we provide the notations we use through the rest of the chapter. We denote by $\mathcal{B}_R$ the set of the *bridge* brokers. They form the peer-to-peer network of the anonymity protocol. We denote by $\mathcal{B}_P$ the set of the *public* brokers. They represent the actual brokers in which the clients publish or subscribe to topics anonymously. Finally, we denote by $\mathcal{B}_D$ the set of *discovery* brokers. They implement the discovery protocol allowing the clients to know which public broker offers which topic. Moreover, they allow peer discovery for the anonymity protocol.

Finally, we denote by $pub(T, I)$ the publish message of the information I on the topic T, and by $sub(T)$ the subscribe message on the topic T.

We report in Table 7.1 the notations used throughout the rest of the chapter.

7.4 The discovery protocol

Through this protocol, we offer a discovery service to the bridge brokers so that they can know which topics are available and which public brokers offer them. This service is provided by the discovery brokers. In principle, just a single discovery broker is enough to implement this protocol. Anyway, for scalability, it is more realistic to distribute the service to more brokers so that bridge and public brokers are not connected to a single possible point of failure [175].

Each discovery broker in $\mathcal{B}_D$ maintains:

- a set of pairs $S = \{(T, B_P)\}$ where $B_P \in \mathcal{B}_P$. Each pair $(T, B_P) \in S$ represents the information that the topic T is available on the public broker B_P. The set S has to be the same for all the discovery brokers in $\mathcal{B}_D$.
- a topic T_N that will contain a single pair (T, B_P) denoting that a *new* topic T is available on the public broker B_P.
- a topic T_S that will provide the current set S to the new bridge brokers joining the network. S is stored as a retained message for the topic T_S.

Since S could exceed the maximum size allowed for the content of a topic, we may consider more topics $T_S^1, \dots T_S^k$, each one containing a portion of S. For the sake of presentation, we consider just a single topic T_S, which is also the more realistic case.

The topics T_N and T_S are prefixed strings in the system.

The broker B_D publishes (to itself) the message S with the retained flag set to true. In this way, all the clients subscribing to the topic T_S receive the retained message immediately after subscribing. To update the set S, it is sufficient for the broker to just re-publish again S (with retained flag set to true). Doing so, as explained in Section 7.2, only the last retained message will be stored.

Consider now a new bridge broker $B_R \in \mathcal{B}_R$ joining the system. It randomly selects a discovery broker $B_D \in \mathcal{B}_D$ and sends it the message $sub(T_S)$. In other words, B_R subscribes to the topic T_S hosted by the broker B_D. This way, B_R receives (as a retained message) the current set S. Then, it removes its subscription to T_S so that it does not receive repeated information (i.e., the set S) when other bridge brokers join the network.

However, a mechanism to update S (when a new topic is available on some public broker) is needed to B_R and B_D. This mechanism is based on T_N.

In particular, when the bridge broker B_R joins the network and (randomly) selects the discovery broker B_D, B_R also sends $sub(T_N)$ to it. This subscription is maintained over time.

Consider now a public broker $B_P \in \mathcal{B}_P$ receiving a publish message or a subscribe message to a new topic T^* through the anonymity protocol of Section 7.5. B_P has to advertise the fact that it hosts this new topic.

To do this, B_P randomly selects a discovery broker B_D and sends the message $pub(T_N, (T^*, B_P))$. B_D adds the pair (T^*, B_P) to the set S. Moreover, since some bridge brokers are subscribers to T_N, they also receive (through the standard MQTT approach) the new pair and update the set S. To propagate the new information to all the bridge brokers, B_D sends the message $pub(T_N, (T^*, B_P))$ to all the other discovery brokers. Since each bridge broker is a subscriber on T_N on some discovery broker, all the bridge brokers eventually receive (T^*, B_P) through the standard MQTT protocol.

7.5 The anonymity protocol MQTT-A

The anonymity protocol we propose in this section takes inspiration from the Crowds protocol [230]. Our solution relies on a P2P network formed by bridge brokers collaborating to forward publish/subscribe messages from MQTT clients to the intended public broker. The collaboration among brokers exploits the bridging mechanism, which is natively supported by the MQTT protocol, thus not requiring infrastructural changes in the MQTT architecture.

Similarly to Crowds, our protocol is characterized by a forwarding probability p_f, namely a probability for a message to remain within the peer network. Moreover, as in Crowds, we require that each bridge broker knows all the peers participating in the P2P network. To guarantee this, we propose an MQTT-based peer discovery protocol, thanks to which every peer maintains an updated set of all the bridge brokers forming the network. This protocol is based on the approach presented in Section 7.4. It leverages one or more discovery brokers holding a set of pairs $S' = (IP_{B_R}, port_{B_R})$, where IP_{B_R} and $port_{B_R}$ represent the IP address and the port of the bridge broker B_R participating in the network.

Moreover, each discovery broker stores two additional prefixed topics: $T_{N'}$ and $T_{S'}$. The topic $T_{N'}$ is used by the discovery broker itself to publish any update of the set S' (an update typically happens whenever a new peer joins the network). Instead, the topic $T_{S'}$ holds the set S' as retained message, so that any new peer joining the network, after subscribing to this topic, immediately receives the set S'.

When a bridge broker joins the P2P network, it proceeds as follows. First, it subscribes to the two topics $T_{S'}$ (to receive the set S') and $T_{N'}$ of a randomly selected discovery broker B_D.

Then, the bridge broker sends a publish message for the topic $T_{N'}$ advertising its IP address and its port to B_D. Doing so, every bridge broker subscribed to the topic $T_{N'}$ of B_D receives such a message and, at the same time, B_D can update the set S'. At this point, B_D sends a publish message to all the other discovery brokers for the topic $T_{N'}$, advertising the new pair of IP address and port. Since each bridge broker is a subscriber on $T_{N'}$ of some discovery broker, this procedure allows every bridge broker to learn all other brokers of the network.

We stress that, as described in Section 7.4, every bridge broker, joining the P2P network, must subscribe to both $T_{N'}$ and $T_{S'}$. However, while the subscription to $T_{N'}$ lasts over time, the subscription to $T_{S'}$ is undone once the set S' is received.

This concludes the description of the peer discovery protocol.

In the following, we describe in detail the anonymity protocol we propose.

To describe our solution we set a scenario involving:

- a public broker $B_P \in \mathcal{B}_P$ hosting the topic T_A (where A stands for *actual*);
- a publisher p aiming to publish information to the topic T_A;
- a subscriber s interested in the topic T_A;
- the set of bridge brokers $\mathcal{B}_R$, among which the broker $B_R^p \in \mathcal{B}_R$ represents the bridge broker directly connected to the publisher p, and the broker $B_R^s \in \mathcal{B}_R$ represents the bridge broker directly connected to the subscriber s.

Among other topics, each bridge broker stores two prefixed topics known to all the bridge brokers: a topic T_F, where F stands for *forwarding*, and a topic $REMOTE$.

When a publish or subscribe message is labelled with a topic having prefix $REMOTE$ or T_F, such a message is forwarded to another broker via the bridging mechanism. Nevertheless, there is a slight difference between the two mentioned topics. A broker can only receive a message labelled with the topic $REMOTE/\#$ from MQTT clients directly connected with it. On the contrary, publish or subscribe messages coming from bridge brokers can only be labelled with the topic $T_F/\#$.

Consider now a publisher p that wants to publish the information $DATA$ under the topic T_A. p sends the message $pub(REMOTE/T_A, DATA)$ to the broker B_R^p. B_R^p randomly selects a bridge broker, say B_R^1, among the peers participating in the network. Then, it remaps the topic of the original publish message ($REMOTE/T_A$) to T_F/T_A and sends the message $pub(T_F/T_A, DATA)$ to B_R^1, via the bridging mechanism. We recall that the bridging mechanism allows B_R^p to act as an MQTT client (a publisher, in this case) towards B_R^1.

Once receiving the message, B_R^1 acts as follows. First, it extracts a random number R between 0 and 1. At this point, two cases can occur.

If $R \leq p_f$, then B_R^1 randomly selects a bridge broker B_R^2 among the peers participating in the network. Then, B_R^1 sends the message $pub(T_F/T_A, DATA)$ to B_R^2, via the bridging mechanism. Observe that, no topic remapping is needed in this case.

If $R > p_f$, then the message must be published at the public broker holding the topic T_A, if any. To do this, B_R^1 finds the pair (T_A, B_P) in the locally stored set S. If such a pair exists, then B_R^1 simply sends the message $pub(T_A, DATA)$ to B_P. Conversely, if such a pair does not exist yet, B_R^1 chooses at random a public broker $B_P \in \mathcal{B}_P$ and then sends it the message $pub(T_A, DATA)$. In this case, being T_A a new topic for B_P, B_P must activate the discovery protocol (see Section 7.4) to notify all the bridge brokers of the new topic. This way, each bridge and discovery broker will update the set S.

Consider now the case in which $R \leq p_f$. In this case, B_R^2 receives from B_R^1 the message $pub(T_F/T_A, DATA)$. At this point, the procedure above described is recursively applied. Therefore, B_R^2 again extracts a random number R' and compares it with p_f. If $R' \leq p_f$, the message $pub(T_F/T_A, DATA)$ is sent to a bridge broker B_R^3 chosen at random. Observe that, in this case, no topic remapping is needed. On the contrary, if $R' > p_f$, then B_R^2 sends the message $pub(T_A, DATA)$ to the public broker B_P, following the procedure described above.

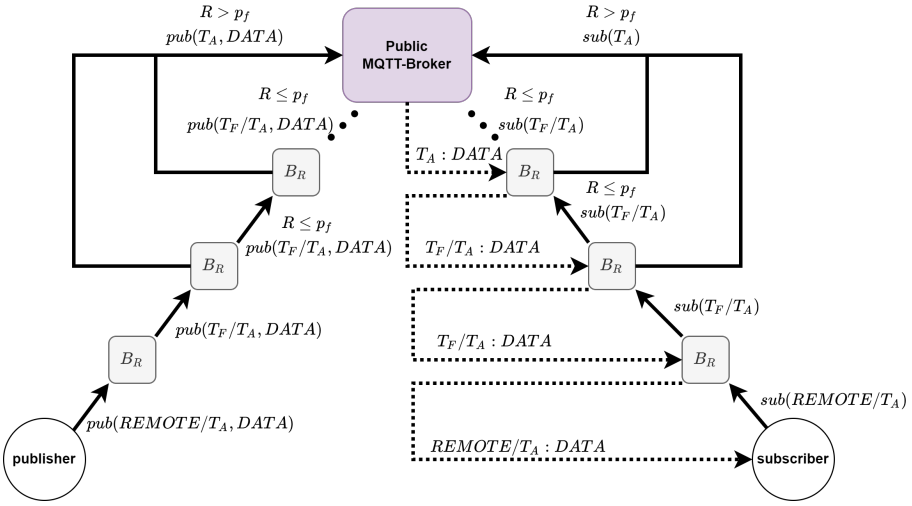


Fig. 7.3: MQTT-A(nonymous).

The mechanism so far described provides anonymity to publishers. Actually, a similar mechanism can be employed so that the same anonymity guarantees are provided to subscribers.

Consider the subscriber s interested in the topic T_A. s sends the message $sub(REMOTE/T_A)$ to its broker B_R^s. Then B_R^s chooses at random a bridge broker, say B_R^4, and sends it the message $sub(T_F/T_A)$, via the bridging mechanism.

Once receiving the above message, B_R^4 acts as follows. First, B_R^4 stores B_R^s in the list of clients subscribed to the topic T_F/T_A, creating this topic if it has not already been locally stored. Afterward, it extracts a random number and compares it to p_f to decide whether to send the subscription message to the public broker B_P (retrieved from its local set S) or to another bridge broker.

In the first case, B_R^4 sends $sub(T_A)$ to B_P and then the procedure stops. Observe that, in this case, the original topic T_F/T_A needs to be remapped to T_A. In the second case, B_R^4 sends $sub(T_F/T_A)$ to a randomly chosen bridge broker, say B_R^5. At this point, B_R^5 recursively repeats the procedure described above. Therefore, B_R^5 stores B_R^4 in the list of clients subscribed to the topic T_F/T_A, possibly creating this topic. Then, B_R^5 extracts a random to decide where to send the received subscribe message. Eventually, after a certain number of iterations, the subscribe message will reach the intended public broker B_P.

Finally, when B_P receives a publish message, say $pub(T_A,DATA)$, it broadcasts the payload $DATA$ to all the subscribers to the topic T_A. Suppose B_R^5 is the bridge broker directly subscribed to the topic T_A at B_P. B_R^5 will receive $DATA$ and, then, it will broadcast this payload to all the clients subscribed to the topic T_F/T_A, among which there is B_R^4. Observe that, being B_R^5 the bridge broker directly connected to B_P, the topic T_A must be remapped to T_F/T_A. Again, following the same mechanism, B_R^4 will send the payload $DATA$ to all the subscribers interested in the topic T_F/T_A, among which there is B_R^s. Once B_R^s receives the payload $DATA$, it will remap T_F/T_A to $REMOTE/T_A$ and, then, it will broadcast $DATA$ to all the subscribers interested in $REMOTE/T_A$, among which there is s.

To conclude this section, we summarize the anonymity protocol in Figure 7.3.

7.6 Path intersection and QoS management

In Section 7.5, to simplify the description of the protocol, we considered the case of a single publisher and a single subscriber. Moreover, we assumed that the paths followed by the publish and subscribe messages labeled with the same topic do not intersect.

Now, we remove the above simplifying assumptions and deal with the consequent impact on QoS guarantees of subscribers.

Two cases deserve to be investigated: (i) the paths followed by at least two subscribe messages labeled with the same topic intersect, (ii) the paths followed by at

least one publish message and at least one subscribe message labeled with the same topic intersect.

As we will see in the following, the problem of path intersection requires ad-hoc strategies to guarantee the QoS level chosen by each subscriber.

We recall that QoS levels 0 and 1 allow duplicate messages, while QoS level 2 requires the message to be delivered exactly once. We stress that both publishers and subscribers can independently choose the desired QoS level for each message being sent or received. Therefore, as an example, it might happen that a client p publishes on a topic with QoS level 1, while there is a subscriber s interested in the same topic but aiming to receive data with QoS level 2. Therefore, s cannot be sure about the fact that it is not receiving duplicate messages.

This QoS mismatching represents an open problem even in standard MQTT. Without loss of generality, in the following, we will consider that clients always publish messages with QoS level 2. This way, any QoS level chosen by subscribers can be met. Moreover, to improve the readability of our discussion, in the following, unless otherwise stated, we will implicitly refer to publish and subscribe messages labeled with the same topic.

7.6.1 Subscribe-Subscribe Path intersection

In this section, we consider a scenario in which a set of subscribers are interested in the same actual topic T_A, hosted by the public broker B_P. Each subscriber might select arbitrarily a certain QoS level for its subscribe message.

Following our protocol, each subscribe message crosses a certain number of bridge brokers before reaching the public broker B_P. Suppose that, before reaching B_P, two or more subscribe messages cross the same bridge broker. In the following, we will call such a broker as *intersection broker*.

At this point, the intersection broker can choose one of the following strategies:

- *blind* strategy;
- *topic-aware* strategy;
- *topic-and-QoS-aware* strategy.

These three strategies are sketched in Figures 7.4,7.5, and 7.6, respectively.

An intersection broker, implementing the *blind* strategy, treats all the subscribe messages the same, regardless of whether they are labeled with the same or different topics. Therefore, subscribe messages labeled with the same topic, in principle, can be routed to different brokers.

An intersection broker, implementing the *topic-aware* strategy, treats the subscribe messages labeled with the same topic differently from all the other messages

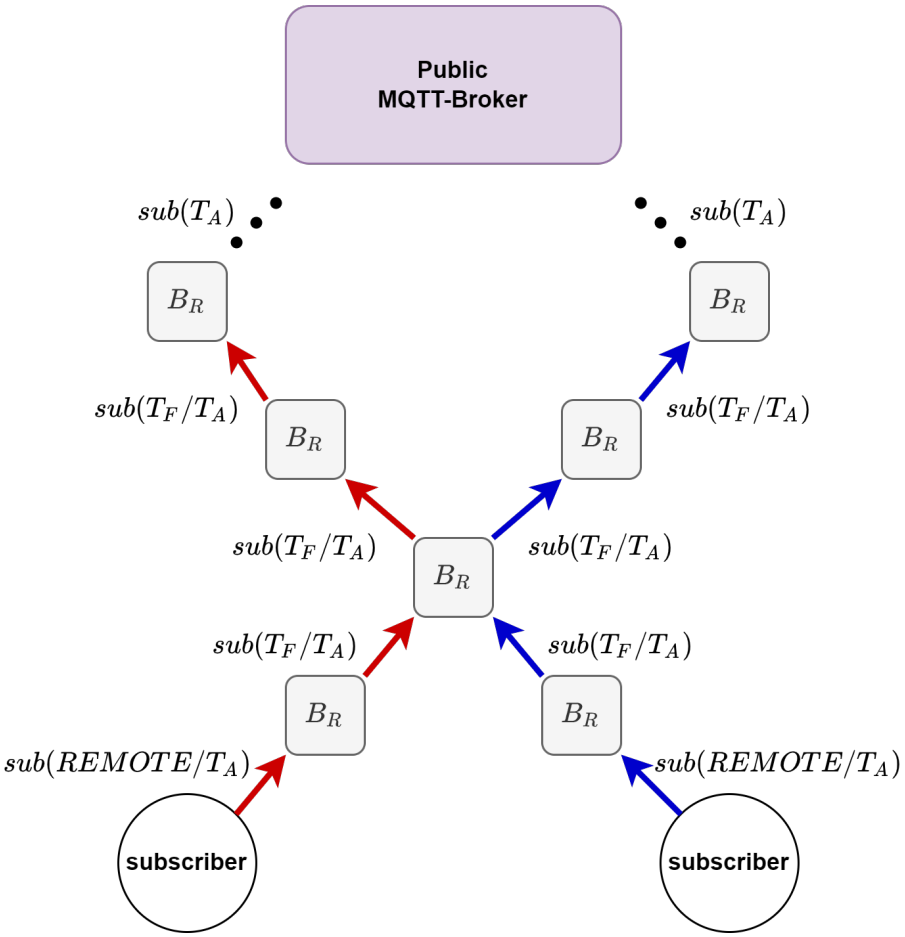


Fig. 7.4: Blind strategy.

crossing it. In particular, referring back to our scenario, just the first received subscribe message to the topic T_F/T_A is forwarded according to the protocol described in Section 7.5. On the contrary, all the other subscribe messages to the same topic that come later are not forwarded further. For such messages, the intersection broker has just to memorize the bridge brokers they come from in the list of brokers interested in topic T_F/T_A. This is enough to guarantee that all the subscribers will be able to receive the data later published to the actual topic T_A. We recall that this strategy is not QoS-aware and therefore messages labeled with the same topic but requiring different QoS are not treated differently.

An intersection broker, implementing the *topic-and-QoS-aware* strategy, distinguishes messages, labeled with the same topic, on the basis of the QoS level they require. To handle the different QoS levels, each bridge broker hosts three forwarding topics $T_F/Q0$, $T_F/Q1$, and $T_F/Q2$. Therefore, subscribe messages sent by clients interested in the actual topic T_A but with different QoS levels will follow different paths in the network. In fact, subscribe messages requiring different QoS levels are treated as messages labeled with different topics (either $T_F/Q0$, $T_F/Q1$, or $T_F/Q2$).

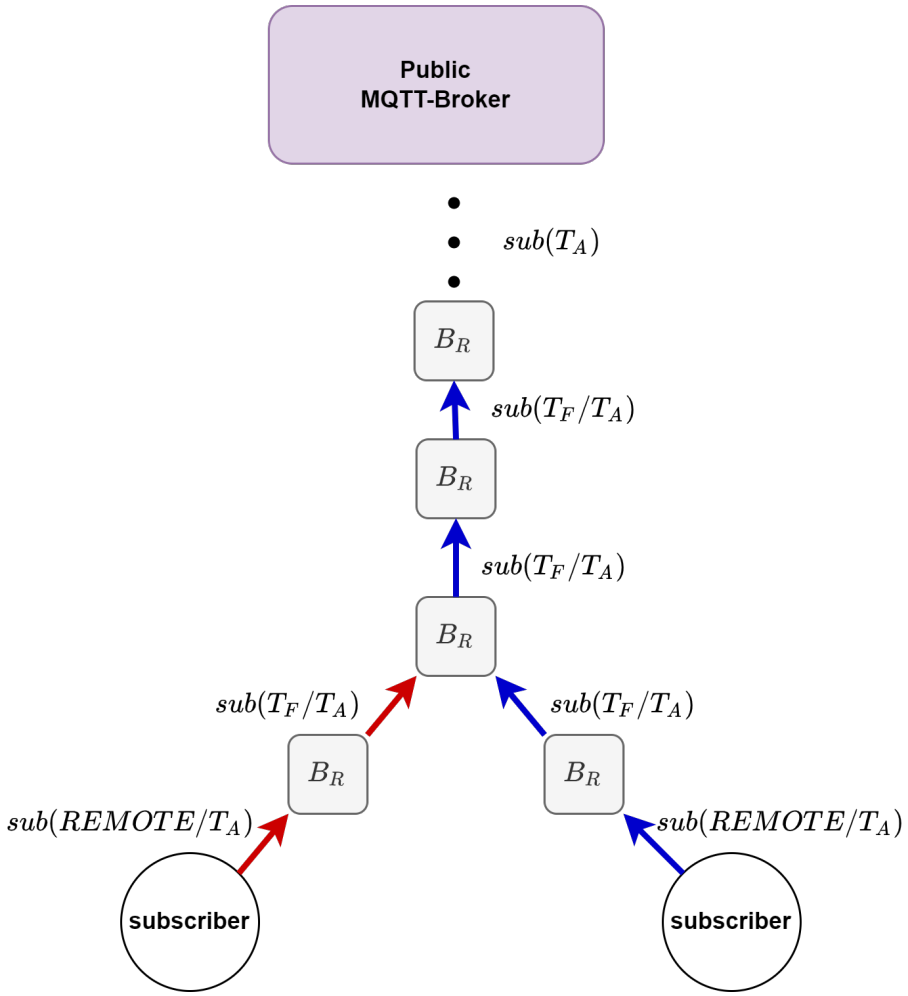


Fig. 7.5: Topic-aware strategy.

On the other hand, concerning messages labeled with the same actual topic and QoS level, the following strategies are applied:

- concerning topics $T_F/Q0$ and $T_F/Q1$, both *blind* strategy and *topic-aware* strategy can be applied (In Figure 7.6, we consider the *topic-aware* strategy);
- concerning topic $T_F/Q2$, only the *topic-aware* strategy can be applied.

7.6.2 Publish-Subscribe Path intersection

In this section, we consider a scenario in which a client aims to publish on the actual topic T_A (hosted by the public broker B_P) and at least one subscriber is interested in the same actual topic T_A.

Following our protocol, each (publish or subscribe) message crosses a set of bridge brokers before reaching the public broker B_P. Suppose that, before reaching B_P, at least one subscribe message and one publish message cross the same bridge broker. As before, we call such a broker an intersection broker.

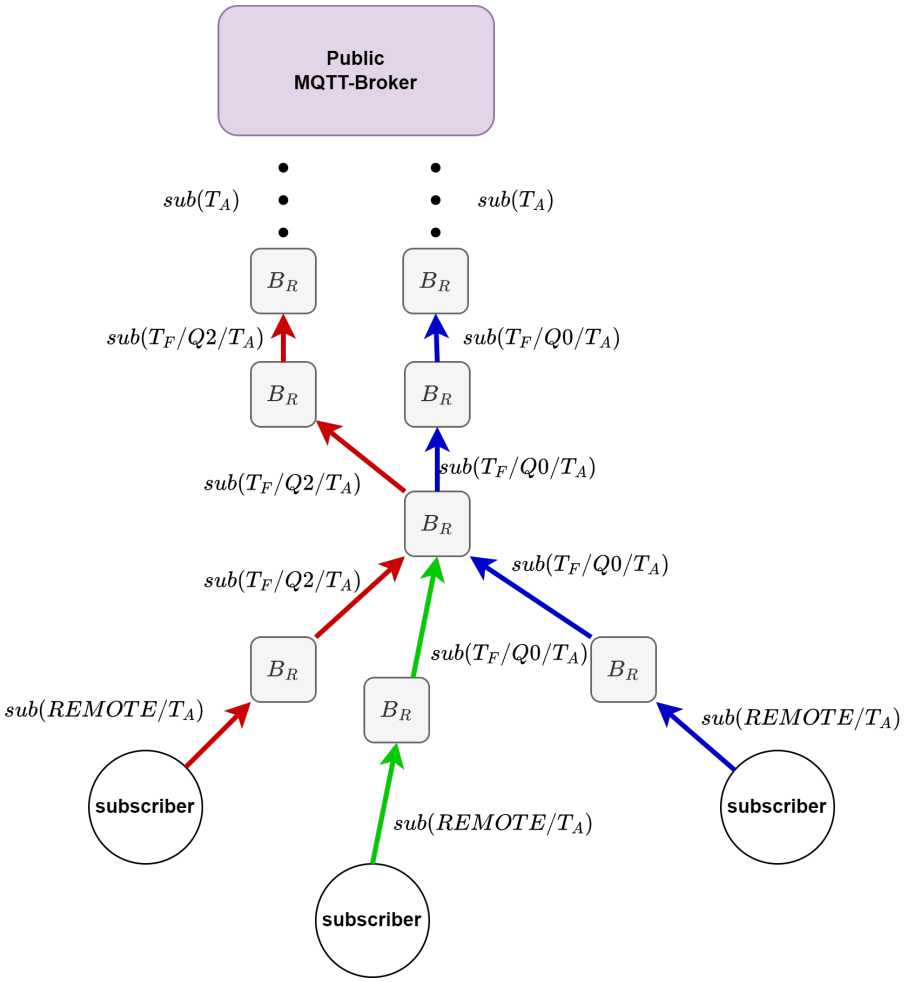


Fig. 7.6: Topic-and-QoS-aware strategy.

At this point, the intersection broker can choose one of the three strategies mentioned above (i.e., either the *blind* strategy, the *topic-aware* strategy, or the *topic-and-QoS-aware* strategy).

An intersection broker, implementing the *blind* strategy, acts as follows. When it receives the publish message it does not publish this message locally, regardless of whether it has locally memorized other bridge brokers subscribed to the same topic labeling the publish message. Therefore, the publish message is further forwarded until it reaches the public broker B_P.

An intersection broker that implements the *topic-aware* strategy acts as follows. When it receives the publish message, it first verifies whether it has locally memorized other bridge brokers subscribed to the same topic of the publish message. If this is not the case, then the intersection broker behaves as described by the *blind* strategy. Conversely, if the above condition is met, the intersection broker publishes the received message locally. This way, all the bridge brokers subscribed to the same topic receive the payload associated with the publish message and, in turn, this pay-

load is forwarded up to the actual subscribers. Observe that, even though the *topic-aware* strategy is applied, the publish message has to be further forwarded to the public broker B_P, since this is the only way that message can reach all the interested subscribers.

As already described, implementing the *topic-and-QoS-aware* strategy requires a slight modification to the forwarding topics hosted by each bridge broker. Indeed, according to this strategy, each bridge broker hosts three forwarding topics: $T_F/Q0$, $T_F/Q1$, and $T_F/Q2$. When a publish message for the topic T_F/T_A reaches an intersection broker, such a broker follows the *topic-aware* strategy. However, differently from before, the message is locally published only under topics $T_F/Q0$ and $T_F/Q1$.

7.6.3 Strategies Comparison

To conclude this section, we examine the advantages and drawbacks of the presented strategies. Overall, the *blind* strategy represents the easiest solution to implement. However, adopting this strategy leads to higher bandwidth consumption than other strategies since messages intersecting in the same broker are simply propagated into the network without being aggregated [249]. On the contrary, the *topic-aware* strategy requires the least bandwidth of the three strategies. However, the main drawback of this strategy is that it is not QoS-aware. To explain why this may represent an issue, we consider the following example. Suppose there are two subscribers s_1 and s_2 both interested in the topic T_A but requiring different QoS, 0 and 2 respectively. We consider that the paths followed by the two subscribe messages in the network intersect. According to the *topic-aware* strategy, the intersection broker further forwards only the first received subscribe message. This way, all the brokers (including the intended public broker B_P) crossed by such a message will memorize the QoS level that it requires. Such QoS level can be either 0 or 2 depending on which subscribe message comes first at the intersection broker. Observe that, since the public broker will be aware of just one subscription out of the two, applying the *topic-aware* strategy also turns out to be advantageous anonymity-wise speaking, not just bandwidth-wise speaking.

Therefore, all the data published on the topic T_A will cross all the bridge brokers from the public broker to the intersection broker respecting the QoS level required by the subscribe message. Observe that, this may be an issue, either for s_1 or s_2. Indeed, supposing the s_1 subscription (requiring QoS level 0) comes first at the intersection broker, the QoS level required by s_2 cannot be satisfied, since it is higher than 0. On the contrary, supposing the s_2 subscription (requiring QoS level 2) comes first at the intersection broker, the QoS level required by s_1 can be satisfied, since it is lower than 2. However, this comes with a cost. Indeed QoS level 2 is satisfied in the

path from the public broker to the intersection broker, while QoS level 0 is satisfied in the path from the intersection broker to s_1. In general, the overhead introduced by the QoS level 2, in the first part (i.e., from the public broker to the intersection broker) of the path, may not be acceptable for s_1.

A similar case might happen when the paths followed by a publish and a subscribe message intersect. Suppose there are a publisher p and a subscriber s, both requiring QoS level 2. According to the *topic-aware* strategy, the subscriber s receives the publish message by the intersection broker. Actually, s will receive the same publish message also by the public broker. Therefore, despite the fact that s chose the QoS level 2, s will receive duplicate messages. This is due to the fact that, regardless of whether or not a path intersection has occurred, the publish message, as well as the subscribe message, must reach the public broker. This way, all the interested subscribers can receive the publish message and the subscriber s can receive all the messages published on the topic by all the publishers in the network, not just by p.

As the occurrence of the two situations above described is not acceptable, bridge brokers can implement the *topic-and-QoS-aware* strategy. Indeed, referring back to our last example, when the publish message reaches the intersection broker, it is locally published just under topics $T_F/Q0$ and $T_F/Q1$, which are also the QoS levels that can tolerate duplicate messages. Such a strategy ensures that the QoS level required by subscribers is always met, provided that publishers guarantee an appropriate QoS level. However, this comes with a cost. Indeed this strategy requires more bandwidth consumption than the *topic-aware* strategy.

7.7 Experiments

Through this section, we perform an experimental validation of MQTT-A and compare it with the standard MQTT protocol. To obtain the anonymity features, MQTT-A introduces a communication overhead. Therefore, the aim of this section is to show that the price we pay is tolerable and the performance results acceptable.

7.7.1 Experimental Setting

For implementation and testing, we relied on HiveMQ CE [125], which is a Java-based open source MQTT broker. We customized the bridge brokers to implement our solution and deployed a P2P network leveraging also Digital Ocean cloud platform [205] to have remote peers. No change is required for clients and public brokers. In detail, for our experiments, we consider two network configurations, one for MQTT-A and the other for MQTT. The aim is to compare the two protocols by measuring latency and goodput.

MQTT-A Configuration In MQTT-A, we have the following components:

- **Clients**: They are deployed through the Java library provided in [126] with no change. They are connected to the bridge brokers through a local network.
- **Bridge Brokers**: They are implemented by customizing the Java library [125] and deployed to form a P2P network. In particular, they are partially deployed on standard laptops and partially deployed on the Digital Ocean cloud platform [205]. To obtain realistic results, the communication bridge-to-bridge brokers always happens through the Internet.
- **Public Brokers**: They are deployed on HiveMQ Cloud, a cloud-based platform hosting MQTT brokers. They are standard brokers and do not require any implementation change. Clearly, the communication bridge-to-public brokers always happens through the Internet.

MQTT Configuration In MQTT, the scenario is a simplification of the previous scenario. In particular, clients are connected to bridge brokers through the local network and bridge brokers are *directly* connected to public brokers through the Internet.

Regarding MQTT-A, we chose to adopt the *blind* strategy (see Section 7.6), which is the simplest one but the worst in terms of performance. As a measure of performance, we considered goodput and latency.

7.7.2 Goodput

For goodput, we mean the number of *useful* information bits received by a subscriber in the unity of time.

To measure it, we fixed a sending rate for the publishers and observed the corresponding goodput for the subscribers. We considered three different sending rates: 1 KBytes/s, 10 KBytes/s, and 100 KBytes/s. We study as the goodput varies as the forward probability of MQTT-A varies. Since the higher the forward probability the longer the paths, we expect the goodput of MQTT-A decreases as the forward probability increases. Obviously, the goodput of MQTT does not depend on the forward probability.

We performed our experiment by considering two levels of QoS (0 and 2). Since QoS level 2 requires additional communication overhead with respect to QoS level 0, the goodput of both MQTT and MQTT-A is lower than QoS level 0.

The results with QoS level 0 are reported in Figures 7.7, 7.8, and 7.9, for sending rate of 1 KBytes/s, 10 KBytes/s, and 100 KBytes/s, respectively. The results with QoS level 2 are reported in Figures 7.10, 7.11, and 7.12, for sending rate of 1 KBytes/s, 10 KBytes/s, and 100 KBytes/s, respectively.

Regarding QoS level 0, we observe that for all the sending rates and almost all the forwarding probabilities, the percentage difference between MQTT and MQTT-A ranges from 3 to 5%. The worst case corresponds to a forward probability equal to 0.9 and sending rate of 100 KBytes/s. The corresponding percentage difference is less than 8%.

Considering QoS level 2, MQTT-A shows a slight worsening. Indeed, for almost all the sending rates and almost all the forwarding probabilities, the percentage difference is between 7-10%. The worst case corresponds to a forward probability of 0.9 and sending rate of 100 KBytes/s in which the percentage difference is less than 18%.

However, high forwarding probabilities cannot be adopted also for security reasons (see Section 7.8). Therefore, we can conclude that the price in terms of throughput is not relevant. For completeness, we show in Figure 7.13, the goodput with a forwarding probability equal to 0.67 and sending rate of 10 KBytes/s. We can observe that the goodput is essentially the same for MQTT and MQTT-A and it is very close to the sending rate.

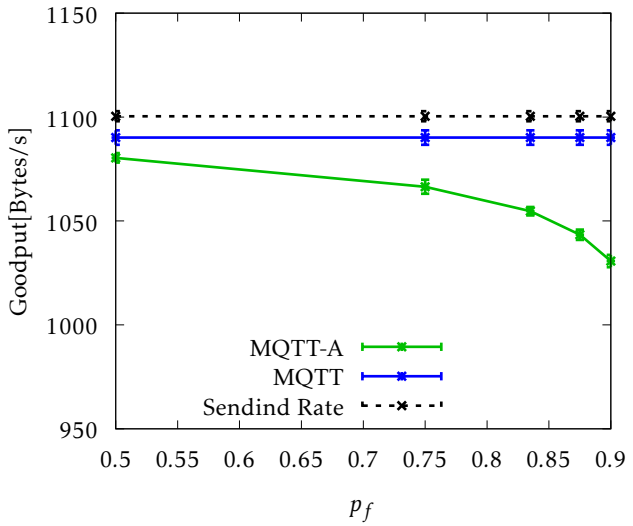


Fig. 7.7: Goodput with sending rate 1 KBytes/s and QoS level 0.

7.7.3 Latency

The second considered metric is the end-to-end latency measured between the instant in which a message is sent by the publisher and the instant in which it is received by the subscriber. We study how this latency varies as the forwarding probability of MQTT-A varies. We measured the latency for three different message sizes

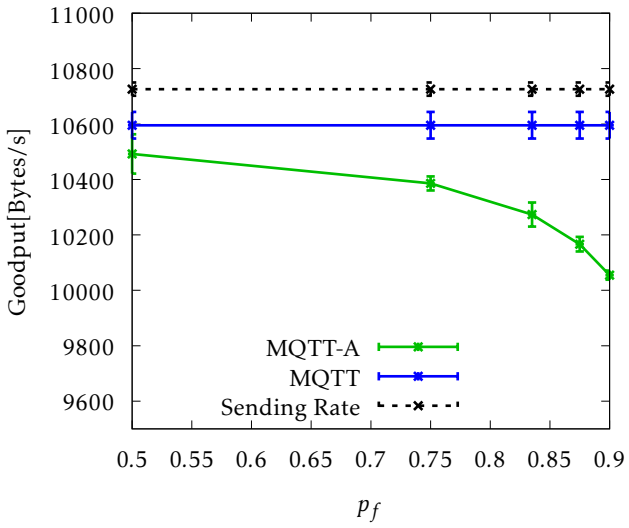


Fig. 7.8: Goodput with sending rate 10 KBytes/s and QoS level 0.

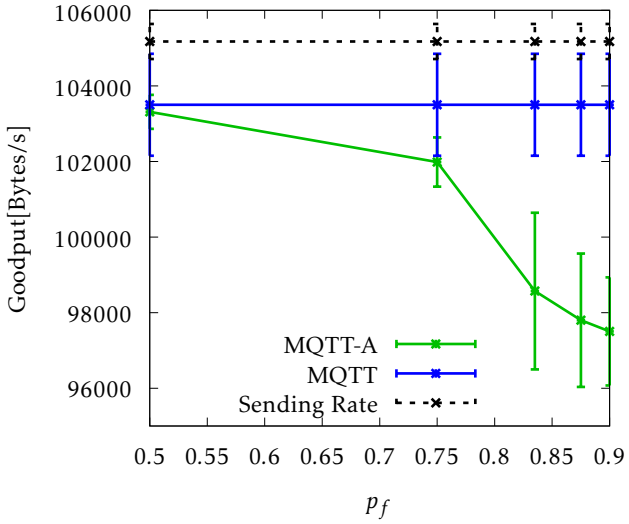


Fig. 7.9: Goodput with sending rate 100 KBytes/s and QoS level 0.

(100 Bytes, 1000 Bytes, and 10000 Bytes), considering two QoS levels (0 and 2). The results are reported in Figures 7.14, 7.15, and 7.16, respectively.

As a first observation, we see that there is no appreciable difference when the packet size changes in both protocols.

Second, latency increases for both protocols by approximately the same factor (1.4-1.5) when the QoS level goes from 0 to 2.

Finally, as expected, the latency of MQTT-A increases with the forward probability (corresponding to longer paths). This is the main drawback of the proposed approach.

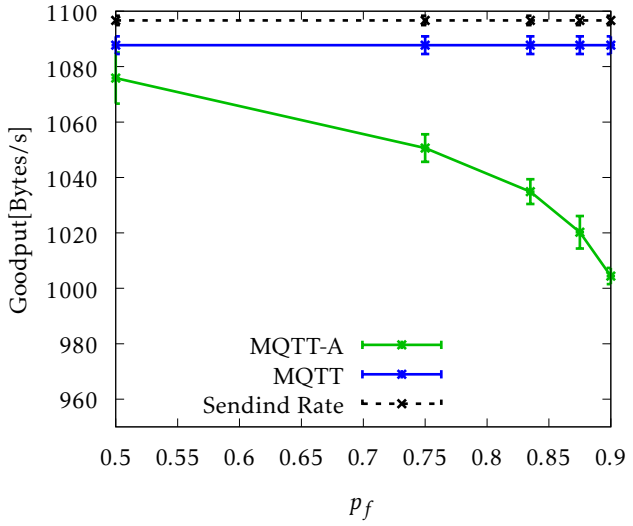


Fig. 7.10: Goodput with sending rate 1 KBytes/s and QoS level 2.

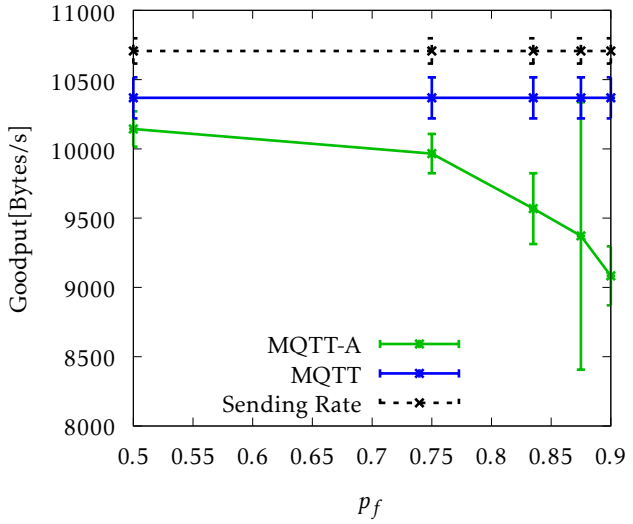


Fig. 7.11: Goodput with sending rate 10 KBytes/s and QoS level 2.

However, we have to consider that this condition guarantees both sender and recipient anonymity. If we relax this constraint and require just one of the two, the latency of MQTT-A is halved.

In Figure 7.17, we set the forward probability to 0.67 and show the value of latency of MQTT (in blue) and MQTT-A when both sender and recipient anonymity are achieved (green) or when just one of two properties is supported (red).

We observe that in the first case (green bar) the ratio between the latency of MQTT-A and MQTT is less than 3, while in the second case, it is less than 1.5. This applies with minimum differences for the two QoS levels and packet sizes.

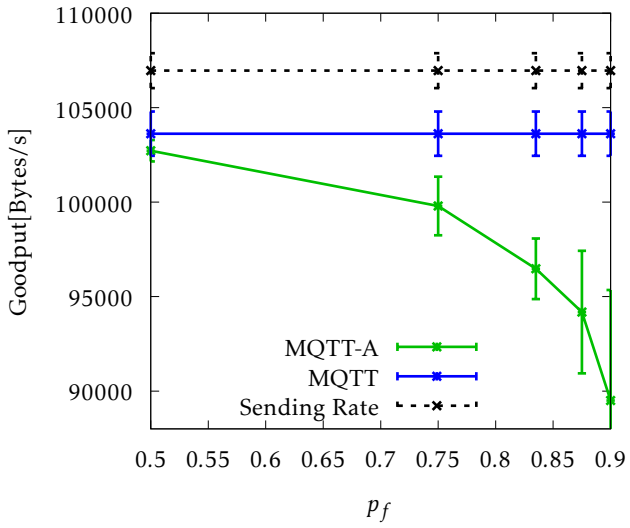


Fig. 7.12: Goodput with sending rate 100 KBytes/s and QoS level 2.

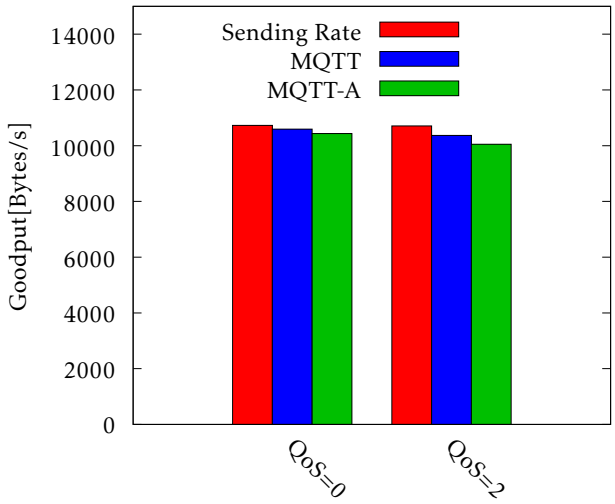


Fig. 7.13: Goodput with $p_f = 0.67$ and sending rate 10 KBytes/s.

7.8 Threat Model and Security Analysis

Since our anonymity protocol is based on [230], we consider the threat model of the original paper. However, a very significant difference exists. In [230], the recipient of the communication to protect is an end server that in our solution corresponds to a public broker. Anyway, such a public broker is not the actual recipient that, instead, is a subscriber to a topic on this public broker. Then, this difference has to be taken into account in our analysis. For example, in Table 1 of [230], when considering the protection of the recipient against the end server as an attacker, it results N/A since the recipient is the end server itself. On the other hand, in our solution, it is not true anymore.

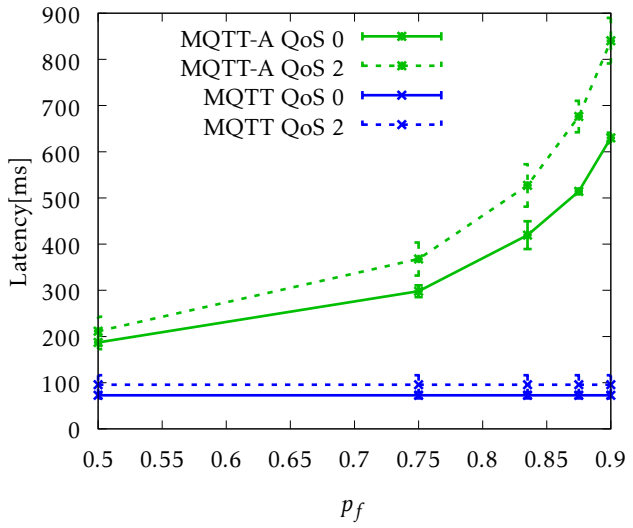


Fig. 7.14: Latency with packet size of 100 Bytes.

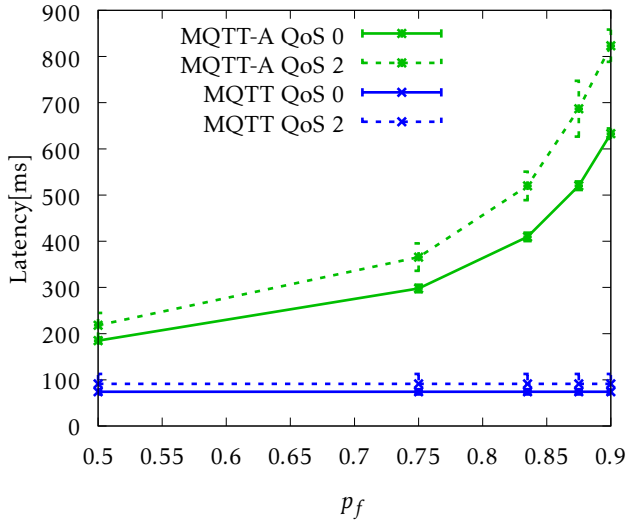


Fig. 7.15: Latency with packet size of 1000 Bytes.

We now describe the considered threat model.

We consider the same adversaries and security properties of [230]. Clearly, they are properly adapted (and contextualized) to our approach to take into account the fact that the recipient of the communication is not a central server (i.e., the public broker), but a subscriber connected to a bridge broker participating in the P2P network.

Attackers.

- **Local Eavesdropper**: An attacker that compromises the bridge broker directly connected to a publisher or subscriber.

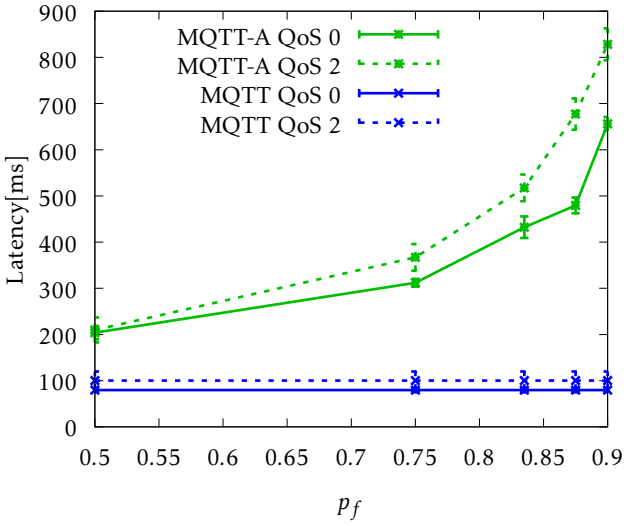


Fig. 7.16: Latency with packet size of 10000 Bytes.

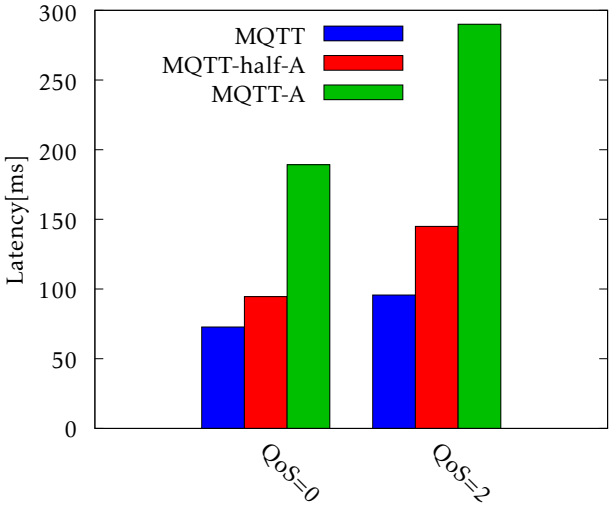


Fig. 7.17: Latency with $p_f = 0.67$ and packet size of 100 Bytes.

- **Collaborating bridge brokers**: A set of bridge brokers participating in the P2P network that collaborate to identify publishers and subscribers.
- **Public broker**: The public broker hosting the actual topics in which MQTT clients are interested. It represents the end server of [230].

Clearly, as [230], our proposal is not oriented to the protection against a global adversary able to observe the entire flow of messages exchanged in the network. As a matter of fact, MQTT (and our proposal too) is designed for wide-area networks, in which the existence of a global adversary is unrealistic.

Security properties [219].

- **Sender Anonymity**: The identity of the publishers is hidden.

- **Recipient Anonymity**: The identity of the subscribers is hidden.
- **Relationship Anonymity**: The attacker cannot discover that a publisher and a subscriber are communicating with each other.

By [219], it is sufficient to guarantee either sender anonymity or recipient anonymity to obtain relationship anonymity.

Now, we analyze how the attackers perform against the security properties. To do this, we consider the following scenario.

We have a publisher p directly connected to a bridge broker B_R^p that publishes data on a topic T_A hosted by the public broker B_P. Similarly, we consider a subscriber s directly connected to a bridge broker B_R^s interested in the topic T_A hosted by B_P.

In favor of security, we neglect the low level of anonymity introduced natively by the bridging mechanism and assume that sender anonymity is broken when the adversary identifies B_R^p (in place of p). Similarly, we assume that recipient anonymity is broken when the adversary identifies B_R^s (in place of s).

Resistance against the local eavesdropper. In this case, the attacker compromises either B_R^p or B_R^s. When the attacker is B_R^p, it is able to observe directly the incoming publish messages coming from p, then sender anonymity is not achieved. However, B_R^p is not able to identify the subscriber s interested in the offered topic. Indeed, even in the case B_R^p receives a subscribe message to forward towards B_P on the same topic, it cannot distinguish B_R^s from the other bridge brokers of the network. Therefore, if the P2P network is sufficiently huge, recipient anonymity is guaranteed and then also relationship anonymity.

Similar considerations can be applied when the attacker is B_R^s. In this case, since it is able to observe directly the incoming subscribe messages coming from s, then recipient anonymity is not achieved. However, B_R^p cannot be identified by B_R^s, thus preserving sender anonymity and then relationship anonymity.

Resistance against collaborating bridge brokers. Regarding this type of adversary, in [230], it is provided a detailed probabilistic analysis that can be applied also to our solution. For the sake of presentation, we report directly the two main results of the analysis and do not repeat the calculations.

The first result is that, given n nodes forming the peer network, we obtain *probable innocence* with respect to sender anonymity against c collaborators, if $n \geq \frac{p_f}{p_f - \frac{1}{2}}(c+1)$.

Probable innocence means that, from point of view of the attacker , the sender appears no more likely to be the originator of a message than to not be the originator.

According to this result, a higher probability p_f allows us to resist a higher number of corrupted nodes.

The second result is that if n is sufficiently high, we obtain *absolute privacy* for sender anonymity with a probability approaching 1. However, the growth of probability can be slow if p_f is large since it is more likely to involve a corrupted node in the path. Therefore, there exists a security trade-off regarding the value of the probability p_f.

Absolute sender privacy against an attacker means that the attacker can in no way distinguish the situations in which a potential sender actually sent communication and those in which it did not.

Clearly, in our application, the role of the sender considered in [230] is played both from B_R^p and B_R^s. As a consequence, if we have a sufficiently high number of bridge brokers, then our solution offers both sender and recipient anonymity and then relationship anonymity.

Resistance against the public broker. In this case, the attacker is B_P. Similarly to [230], sender anonymity is achieved since the publish message sent by B_R^p cannot be distinguished by B_P from a publish message originated from any other bridge broker. Unlike the previous adversary, this result does not depend on the probability p_f.

Regarding recipient anonymity, while in [230] it is not applicable since the recipient is the server itself, in our application we can consider the recipient from the point of view of B_P simply as a sender of a subscribe message. Then, by applying the same reasoning done for B_R^p, also B_R^s cannot be distinguished from any other bridge broker.

Then, we obtain sender, recipient, and relationship anonymity against this adversary.

7.9 Related Work

Security in MQTT is an open problem [215, 217, 184]. On the one hand, implementing security mechanisms is crucial to protect end-to-end clients. On the other hand, since MQTT is adopted when constrained devices are involved [263], complex security solutions cannot be applied. Therefore, MQTT does not provide any built-in security mechanism.

A first issue regards *confidentiality*. The basic approach consists of using TLS to establish secure channels [80]. However, it has a negative impact on performance and energy consumption [224]. Therefore, more advanced solutions have been proposed in the literature [182, 253, 2, 244, 237, 81, 204, 138].

Often, the confidentiality mechanisms are adopted to reach also *authorization* (with a focus on *access control*) [198, 187]. A lot of works pursuing both authorization

and confidentiality are based on CP-ABE or KP-ABE [116, 31, 185, 172]. Clearly, these schemes differ from the standard ABE schemes since they are tailored for being used by constrained devices.

Another security feature investigated in MQTT is *authentication* [48, 30, 15]. For example, in [46], a multi-factor blockchain-based solution for authenticating clients is provided.

The problem of *privacy* in MQTT [9, 133] is more related to our work. An interesting solution aimed to obfuscate the topics when public brokers are involved is provided in [97]. However, it requires pre-shared keys between clients making this solution not compliant with more general scenarios (such as that described in this paper) in which publishers and subscribers do not know each other. The same consideration applies for [133]. In principle, the above techniques aim to protect the content of communication, while our proposal is devoted to protecting peer identities. Clearly, the two approaches are not in contrast and could be combined to obtain a higher level of privacy.

The exact context in which our paper falls is *anonymity*, in which the aim is to prevent the identification of the clients publishing or subscribing to some topics. In the literature, to the best of our knowledge, no relevant and complete proposal is available in this direction. The only approach that presents some similarities with our proposal is [227]. Therein, a Tor-like [258] solution is designed in which clients connect to the brokers through a path of intermediate brokers that forward messages encrypted in Onion fashion. However, in the short paper, only the rough idea of the approach is presented without providing any implementation or experimental validation. Furthermore, no discovery protocol and no security analysis are included.

Part III

Anonymous Service delivery

When a given service is provided on the basis of some personal data, the protection of the communication between the user and the service provider may not be enough to protect the identity of the users.

A typical example is provided by location-based services (LBS) in which a user obtains a service after disclosing their position. As well-known in the literature, the position is a quasi-identifier [240], easily linkable with user's identity. In this case, also by relying on a perfect anonymous communication network, the identity of the user may be retrieved by a service provider receiving the position. Then, new challenges arise.

The goal of this third part of the thesis is twice. On the one hand, we want to provide users with privacy-preserving services so that their identity is kept secret from all the non-trusted parties. This is achieved in Chapter 8, in which we propose a hierarchical LTS system offering protection against a global adversary. The services we consider in this solution are location-based services and the aim is to provide the users with guarantees about the fact that the provider is not able to identify their positions. As with many state-of-the-art approaches, our solution relies on the presence of some location-trusted servers (LTSs) that build, starting from the position of a user u, a cloaking area including at least $k-1$ different users indistinguishable from u. However, in standard LTS-based solutions, the LTS is a centralized entity with a global view of all the positions of the users. In our approach, we reduce the trust required of a single LTS by splitting its competence on a given set of positions among many LTSs, each possibly managed by an autonomous organization.

The second goal of this part is to give an answer to the following question. It is possible to enable new useful features, by relaxing the anonymity requirements. The answer to this question is positive and it is the objective of Chapters 9 and 10.

In Chapter 9, we present a solution to the trade-off between accountability and anonymity. In particular, we consider a scenario in which a user is known to the service provider just by means of a pseudonymous username. To enable the possibility to re-identify the user in case of malicious or illegal behavior, we require the collab-

oration of three parties. This is the main advantage with respect to other solutions in which just the collaboration of two parties is enough. This solution can be adopted to support an anonymous social network in which users can interact anonymously also with respect to the service provider itself. However, in the case of acts of cyber-bullying or incitement to hate, a law-authorized party can identify the user with the collaboration of the three parties involved in our protocol.

Regarding Chapter 10, we present a solution that enables the anonymous linkage of open data only by some authorized parties. Specifically, we are in a smart city scenario in which a user interacts with several subsystems by producing data published in the form of open data. Even though these data are anonymized for privacy reasons, our solution enables their linkage only to authorized parties which, however, are unable to discover the real identity of the user to whose data refers.

8

A hierarchical LTS system offering protection against a global adversary.

Privacy-aware location-based services (LBS) can be obtained by protecting the user's identity so that queries cannot be linked with users. A way to do this is to use a Location Trusted Service (LTS, for short), to transform the exact position of the user into a cloaking area including at least k users (thus, obtaining k-anonymity). In wide-area scenarios, a centralized LTS organization might represent a serious threat in terms of security and privacy. This chapter proposes a hierarchical multi-provider distributed LTS organization, which mitigates the above security privacy risks by splitting competence areas and minimizing access to punctual data. This hierarchical organization of the LTS enables also the possibility to gain protection against a global passive adversary without flooding the network as would occur with a centralized approach. Finally, the hierarchical organization traces the road for a possible edge-cloud-based implementation, whose benefits are also highlighted by our experiments. Some preliminary results of this approach are published in a research paper [39].

8.1 Introduction

Location-based services (LBS) occupy an important position within pervasive, ubiquitous, and wide-area computing systems. There are many types of location-based services, such as navigational, resource-discovery (typically, points of interest), traffic, news, weather, emergency, advertising, location-based games, etc. [255, 28, 174, 298, 154]. They can be continuous (such as navigational services), may require different localization precision, and can be delivered as push services. LBS may represent a serious threat to people's privacy. Indeed, the link between the content of the required service and the location itself may allow an honest-but-curious provider to link the user's identity with sensitive information like habits, health state, religious, or sexual orientation. This is a very well-known problem in the literature, and basically, it depends on the fact that location data are *quasi-identifiers*, allowing

the adversary to discover the identity of the victim if combined with background knowledge or through collusion among different adversary parties.

One of the approaches used to contrast the above problem is to protect the user's identity. The goal is to prevent location-based queries from being linked to users' identities. This can be obtained by relying on a Trusted Third Party, named *Location Trusted Service* (LTS, for short). The role of the LTS is to mediate the queries coming from the users proxying them to the LBS provider. However, each query is not forwarded as it is. Instead of the exact user position, the LTS builds a *cloaking area* including at least k users and such that, for each of those k users, the associated cloaking area is the same. This way, *location k-anonymity* [102, 145, 113] is achieved. The stability condition of the cloaking area, called *reciprocity* [149, 106, 63] is fundamental, because it prevents reverse-engineering attacks [63] that can reduce actual anonymity. Cloaking areas should also satisfy the *effectiveness* property [106] that requires the minimization of their extension.

A problem with the LTS-based approach is that the concrete implementation of the LTS in wide-area scenarios is a complex task. Indeed, implementing the LTS as a centralized entity might represent a serious threat in terms of security and privacy. This is due to the fact that the LTS keeps the entire location data of users and tracks them every time and everywhere. Moreover, if the LTS is compromised, it will pose user information in jeopardy.

This chapter deals with the above problem by providing a concrete location-trusted service system based on a hierarchical multi-provider distributed organization. The proposed system is distributed and hierarchically organized, in such a way that the competence is split among many LTSs, each possibly managed by an autonomous organization. Moreover, multiple services and possible overlapping of regions are managed.

Clearly, a hierarchical approach can really mitigate the above security privacy risks only if the providers covering wide areas (thus, high in the hierarchy) do not manage exact location data, but only aggregate values.

Therefore a problem to study for an effective hierarchical LTS implementation, is how to exploit any existing cloaking-area-construction algorithm that works on exact positions to build cloaking areas on aggregate data preserving both reciprocity and effectiveness.

In this chapter, we propose a solution to the above problem.

Observe that the LTS-based approach is completely nullified if a global passive adversary is allowed, able to monitor the flow of messages, as the source of the query can be identified among the anonymity set of users. In real-life contexts, this is for example the case in which we want to provide LBS completely within a mobile so-

cial network. In this case, the LTS could be an entity that interacts with the users and the LBS provider (possibly, the social network itself) by using communication mechanisms provided by the social network (and thus completely observable by the provider). Concerning this aspect, a nice feature of our approach is that the hierarchical organization of the LTS enables the possibility to use position notification as cover traffic to hide queries and multicast to hide responses against the global adversary, without flooding the network as would occur with a centralized approach.

In this chapter, we also highlight that our hierarchical distributed approach can benefit from the edge-cloud paradigm [151, 171] for a real-life implementation. Indeed, the lower the LTSs in the hierarchy, the closer to the user they can be implemented, thus reducing network latency for local queries and better distributing the computational load.

8.2 Background

In this section, we provide the background knowledge useful for the comprehension of the rest of the chapter. Specifically, we give the fundamentals of the location-trusted-service-based approach used to protect privacy in location-based services.

Location-based services (LBS) are services based on the location of users. They may lead to serious threats to privacy. There are different approaches used to contrast this problem. Our proposal refers to the approach aimed to protect the user's identity. This approach is based on the presence of a *Trusted Third Party* (TTP), called *Location Trusted Service* (LTS), which plays the role of anonymizer of the user's requests towards the LBS provider. The LTS receives the query from the user and, instead of the exact position, sends a cloaking area to the LBS provider including at least k users (including the requester), in such a way that *location k-anonymity* [113] is achieved. This means that the probability for the adversary (i.e. the LBS provider) to identify the user requesting a location-based service is at most $\frac{1}{k}$, provided that an additional feature (detailed below) is adopted.

To give the flavor of the approach based on the location k-anonymity, let's consider the following example. Suppose that Bob wants to know the points of interest close to his position by relying on an LBS provider. To prevent an honest-but-curious LBS provider from learning information about Bob related to the query, a possible way is to protect Bob's identity, so that the LBS query cannot be linked to him. Unfortunately, the use of anonymous IDs is not enough. Indeed, the location itself is a *quasi-identifier* and, therefore, allows the LBS provider to identify Bob if the location data are combined with other public data, background knowledge, or through collusion with external parties. To avoid this, the LTS is placed in the middle, between

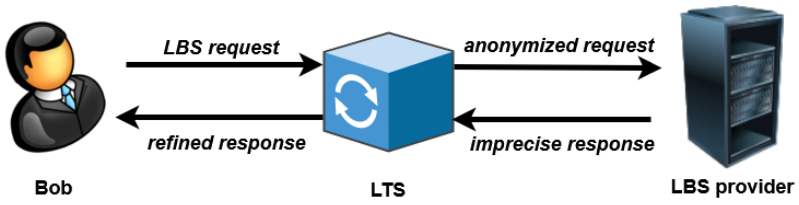


Fig. 8.1: An example of construction of a cloaking area.

.

Bob and the LBS provider. The query is submitted by Bob to the LTS (playing as Trusted Third Party), along with his exact position. The LTS builds a cloaking area, including an anonymity set of users (AS), to which Bob belongs. If the required privacy level is k, then the cardinality of the anonymity set must be no smaller than k. These users are selected in such a way that, on the basis of the knowledge available for the LBS provider, they are indistinguishable to the latter. The LTS removes the user ID and submits the query to the LBS provider along with the cloaking area, instead of the exact position. This is done with the objective to obtain that the probability for the adversary (i.e., the LBS provider) to identify Bob and link him to the query is not higher than $\frac{1}{k}$, which is acceptable for a sufficiently large value of k. The LBS sends the answer to Bob's query to the LTS, which can filter it based on Bob's exact location, and sends the refined response to Bob to minimize client-side communication overhead. The above scheme is summarized in Figure 8.1.

More formally, we define the following.

Definition 8.1. Consider a user u issuing a query with privacy level k. A *cloaking area* (built by the LTS) is any area A such that a set of users AS (called, *anonymity set*) can be found by the LTS within A such that $u \in AS$ and $|AS| \geq k$.

Unfortunately, a cloaking area so defined is not enough to guarantee the required privacy level, even though users inside AS are really indistinguishable to the adversary. Indeed, a technique only satisfying the above requirements is vulnerable to reverse-engineering attacks [63].

To prevent this problem, the following property has to be required.

Definition 8.2. (Reciprocity Property [106]). Consider a user u issuing a query with privacy level k, associated by the LTS with a cloaking area A, and involving the anonymity set of users AS. We say that A satisfies *reciprocity* if every user in AS also generates the same anonymity set AS for the given privacy level k. A cloaking-area-construction algorithm is *reciprocal*, if every returned cloaking area A (for each possible user, and each possible privacy level k) satisfies reciprocity.

Clearly, the condition stated in Definition 8.2 is also satisfied if every user in AS also generates the same cloaking area A (which then involves the same anonymity set AS). As a matter of fact, this is the condition satisfied by reciprocal algorithms proposed in the literature, starting from the first proposal given in [106]. On the other hand, different returned cloaking areas, even including the same AS, could enable reverse-engineering attacks if not adequately built.From Definition 8.1, immediately follows that a cloaking area satisfying reciprocity satisfies also location k-anonymity. However, Definition 8.2 entails that k-anonymity ensures equal probability $\frac{1}{|AS|}$ for the adversary to identify a user. As shown in [106, 63], without reciprocity, reverse-engineering attacks are possible allowing the adversary to reduce the above probability below the aimed level $\frac{1}{k}$. Observe that reciprocity has been independently formulated as the *k-shared* property in [63].

8.3 Approximate Cloaking Areas

Our approach relies on any existing reciprocal cloaking-area-construction algorithm (see Definition 8.2). Since reciprocity, for a given privacy level k, implies location k-anonymity, for a reciprocal cloaking-area-construction algorithm and for the returned cloaking areas, we say that they satisfy the *k-reciprocity* property, to highlight the privacy level k. Recall that if the returned cloaking area includes at least k users and is the same for all those users, then the algorithm is reciprocal. As highlighted in Section 8.2, coherently with the reciprocal algorithm proposed in the literature, Definition 8.2 is applied in our approach in the sense that the returning cloaking area is the same for every user in the anonymity set AS. Moreover, throughout the chapter, we implicitly assume that the anonymity set is the set of all the users belonging to the cloaking area.

Consider a geographical area A that is a set of GPS coordinates, called *positions*, identifying the portion of a territory with a certain degree of granularity. We denote by $P \subseteq A$ the set of the positions associated with all the users located in A. P is called *actual position set*.

Definition 8.3. *A Cloaking Oracle $\mathcal{O}_A$ (for the area A) is a partial function that receives as input a privacy requirement k, a set of positions Q, and a position $p \in Q$, and returns, if any, a cloaking area $C_Q \subseteq A$ such that $p \in C_Q$ and satisfies k-reciprocity.*

According to the definition of reciprocity recalled in Section 8.2, from the above definition it follows that there exist at least k positions $p_1, \ldots, p_k \in C_Q \cap Q$ including p such that $\mathcal{O}_A(k, Q, p_1) = \ldots = \mathcal{O}_A(k, Q, p_k) = C_Q$.

Observe that, the Cloaking Oracle is a partial function because, depending on k, on a certain input, it could happen that there is no way to build the cloaking area.

Anyway, this is not a limit of our work, which considers an *ideal* algorithm. This case only refers to the intrinsic non-compliance of the distribution Q with the privacy requirement (for example, if $|Q| < k$).

Now, consider a partition Z_A of A identifying sub-areas of the territory. Even though the partition we consider in Section 8.4 divides A into a grid of equi-sized squares, the next result works for any shape of the sub-areas. Each element of Z_A is called *cell*.

We introduce the following two definitions.

Definition 8.4. *The aggregation mapping (on P) is a function $D_P : Z_A \to \mathbb{N}$ which, for each cell, returns the number of positions of P in it included. Formally, $D_P(z) = |z \cap P|$, for each $z \in Z_A$.*

Definition 8.5. *Given D_P, a set of positions $P' \subseteq A$ is said equivalent to P, if $D_P = D_{P'}$ (i.e., $|z \cap P| = |z \cap P'|$, for each $z \in Z_A$).*

In words, P' is any redistribution of the users (starting from the actual distribution represented by P) preserving the number of users occurring in each cell. We denote by $\mathcal{R}(D_P)$ a given fixed *generation schema* (i.e., a deterministic function) of a set of positions equivalent to P. Therefore, given D_P, any party that is aware of this schema can deterministically generate the same set of positions $P' = \mathcal{R}(D_P)$ equivalent to P. From now on, throughout the chapter, consider given the geographical area A, the actual position set P, the partition Z_A, and the generation schema $\mathcal{R}$.

Given a cloaking area C_Q, for a certain set of positions Q, we denote by $S_{C_Q} = \{z \in Z_A | z \cap C_Q \neq \emptyset\}$. S_{C_Q} represents the set of cells that are involved by C_Q.

Definition 8.6. *We define the approximate cloaking area (of C_Q) as $\overline{C}_Q = \bigcup_{z \in S_{C_Q}} z$.*

Observe that $\overline{C}_Q$ identifies a sub-area of A including C_Q.

Given a set of positions P' equivalent to P, consider the cloaking area $C_{P'}$ obtained as output of $\mathcal{O}_A(k, P', p)$. The idea is that, from just the knowledge of D_P, any agent may construct a cloaking area by satisfying the k-reciprocity requirement, just by calling the Cloaking Oracle on a set of positions P' equivalent to P.

We are ready to define the algorithm that an LTS can follow to build a valid clocking area (with respect to a given privacy requirement k), by relying only on the aggregation mapping D_P, instead of P.

The Algorithm 1, called *approximate cloaking-area construction* (denoted as $\mathcal{A}_{cloak}(k, D_P, p')$), has input: (1) the privacy requirement k, (2) an aggregation mapping D_P, and (3) $p' \in \mathcal{R}(D_P)$, and output either *fail* or an approximate cloaking area. $\mathcal{A}_{cloak}$ proceeds as follows: (1) Relying on D_P and $\mathcal{R}$, it builds the position set $P' = \mathcal{R}(D_P)$ (recall that P' is equivalent to P); (2) It calls $\mathcal{O}_A(k, P', p')$. If the Oracle is not able to respond (i.e., $\mathcal{O}_A(k, P', p')$ is not defined), then $\mathcal{A}_{cloak}$ returns *fail*,

Algorithm 1 Approximate cloaking-area construction

Notation $\mathcal{O}_A$: the Cloaking Oracle for the area A
Notation $\mathcal{R}$: the generation schema
Input k: privacy requirement
Input D_P: aggregation mapping
Input p': a position in $\mathcal{R}(D_P)$
Output *fail* or $\overline{C}_{P'}$
1: $P' = \mathcal{R}(D_P)$
2: **if** ($\mathcal{O}_A(k, P', p')$ is not defined) **then**
3: **return** *fail*
4: **else**
5: $C_{P'} = \mathcal{O}_A(k, P', p')$
6: **return** $\overline{C}_{P'}$ of $C_{P'}$
7: **end if**

else ($C_{P'} = \mathcal{O}_A(k, P', p')$), the execution goes to the next step; (3) $\mathcal{A}_{cloak}$ returns the approximate cloaking area $\overline{C}_{P'}$ of $C_{P'}$.

In Figure 8.2, an example of execution of the approximate cloaking-area construction is reported.

Specifically, map **(a)** represents the geographical area A divided into cells by Z_A, including a certain set of users with actual positions P, and a given position (in red), which the LBS request comes from. To understand the meaning of the red and green areas, we need first to describe the other maps. Suppose that the privacy requirement is k =12, so that the final cloaking area must include at least 12 users. Map **(b)** denotes the application of the aggregation function on P. Finally, map (c) shows the redistribution $P' = \mathcal{R}(D_P)$ of users (preserving the aggregation mapping), the cloaking area $C_{P'}$ (in green) returned by the Cloaking Oracle, and the approximate cloaking area $\overline{C}_{P'}$ (in red). The example shows that even though the Cloaking Oracle returns on P' the green cloaking area fulfilling the privacy requirement (as it includes k=12 positions), being P' a set of dummy positions, the cloaking area itself could be not valid if applied to the actual position set P. This is the case we are representing. Indeed, when projecting the green cloaking area from map (c) to map (a), we see that the actual involved positions are less than the required number (as they are 10 positions).

In contrast, the red approximate cloaking area, which, by construction, includes the same number of positions if applied to either P or P' (for P and P' equivalent), satisfies the privacy requirement as the number of actual (or dummy) included positions equal to 16.

For simplicity, in this example, we did not consider the reciprocity property.

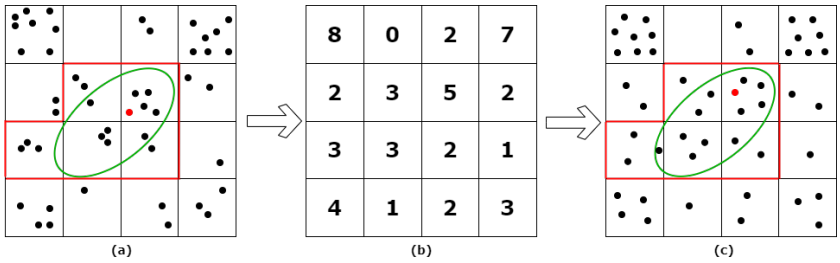


Fig. 8.2: An example of construction of an approximate cloaking area.

.

8.4 The Distributed LTS

In this section, we describe the LTS organization. The area A is divided into n rectangles each containing a number of square cells in Z_A. Let denote by $A_Z = \{A_1, \dots A_n\}$ the set of such rectangles. The choice of the rectangle as a shape of an elementary group of cells is done only for the sake of simplicity. Any other shape could be in principle utilized.

In our system, multiple hierarchical LTSs are adopted, each possibly managed by an autonomous organization. They are organized as follows. Each rectangle, called 0-zone, is under the responsibility of an LTS of level 0. Each LTS of level 0 responsible for the 0-zone A_i (with $1 \leq i \leq n$) knows a subset of actual positions $P_i \subseteq P$ representing the actual positions of the users in the 0-zone A_i.

The LTSs system basically implements a forest of tree spatial indices. Specifically, a number of LTSs of level 0 managing adjacent 0-zones constitutes the set of children of an LTS of level 1. Such an LTS is responsible for a 1-zone obtained as the union of the 0-zones which each of its children is responsible for and knows only the restriction of the aggregation mapping D_P to the subset of P involved by the 0-zones forming the 1-zone of its competence.

In general, a number of LTSs of level i managing adjacent i-zones constitutes the set of children of an LTS of level $i+1$. Such an LTS is responsible for an $(i+1)$-zone obtained as the union of the i-zones which each of its children is responsible for and knows only the restriction of the aggregation mapping D_P to the subset of P involved by the 0-zones mapped to the $(i+1)$-zone of its competence. In other words, each LTS of level $i > 0$ has competence on a number of rectangles of A and, then, on the involved cells but knows only, for each cell, the number of users positioned in it and not the exact position. This happens also for the root of each tree, which is set in such a way that the size of the map of its competence is feasible. This is the reason why we do not have a single tree, but a forest of trees. We assume that all the LTSs share the generation schema $\mathcal{R}$.

The LTS hierarchical organization is sketched in Figure 8.3.(a). In detail, there are three 0-zones, colored in green, red, and blue, respectively, and their LTSs of level 0. Users are also represented. The red user is submitting an LBS query. The union of these three 0-zones forms the 1-zone of competence of the black LTS of level 1. Figure 8.3.(b) reports the view from the black LTS of this 1-zone (which includes only the aggregation mapping). The red dot corresponds to the position which the query comes from. The red-line ellipse and the black-line region will be described later.

8.4.1 Registration

When a user enters a 0-zone, it performs a pseudonymous registration to the LTS of level 0 responsible for such a zone. Each LTS of level $i \geq 0$ (in a tree of the forest) knows its parent LTS (of level $i+1$) according to the LTS hierarchy defined above. Similarly, each LTS of level $i \geq 1$ knows all its children (LTS of level $i-1$).

8.4.2 Position Notification

Periodically, with a given frequency, each user located in a 0-zone A_i sends their position to the LTS of level 0 responsible for such a 0-zone. Therefore, such LTS, say L, knows the set of positions P_i of the users in the 0-zone. The above component of the position notification process is called *detailed position notification*.

Now, we describe how aggregate positions are notified to the higher levels of the LTS hierarchy. We call this task *aggregate mapping notification*. Let's consider the LTS L again. We denote by D_{P_i}, the restriction of D_P to P_i. L can compute $P_i' = \mathcal{R}(D_{P_i})$. Then, it builds and maintains a one-to-one correspondence $\mathcal{S}$ between the sets P_i and P_i' in such a way that any actual position $p \in P_i$ is associated with a dummy position $\mathcal{S}(p) \in P_i'$ such that $p, \mathcal{S}(p) \in z$ for some $z \in Z_A$ (i.e., p and $\mathcal{S}(p)$ are in the same cell). As will be clarified in Section 8.6, this is done to guarantee k-reciprocity (not with the purpose of hiding the actual position).

Periodically, L sends D_{P_i} to its parent LTS (of level 1). This counts the number of users for each cell contained in its 0-zone. At this point, the LTS of level 1, by merging the information coming from its children, is able to know the restriction of D_P to P^*, where P^* is the set of the positions of the users in the 1-zone which such an LTS is responsible for. Observe that, it does not know P^*. Periodically, this restriction of D_P is sent to its parent LTS (of level 2). The process is iterated until the root of the tree is reached.

8.4.3 LBS Request Processing

Suppose a user in position p performs an LBS request. To prevent the global adversary can identify that a user submits a query to the LBS provider, the user replaces one of the messages of the detailed position notification (intended for the LTS of level 0 responsible for the 0-zone which the user is located in) with the LBS request including the position p and an on-the-fly key K. Suppose that such 0-zone is A_i and the set of positions of the users including in it is P_i, this LTS invokes the Cloaking Oracle $\mathcal{O}_A(k, P_i, p)$.

We distinguish 2 cases. The first case is when the Cloaking Oracle outputs a cloaking area C_{P_i}, then the LTS can directly perform the request to the LBS provider and after receiving the response and (possibly) filtering it, sends the response encrypted with K in multicast to the users included in the cloaking area.

The second case occurs when the Cloaking Oracle fails. If this happens, the LTS of level 0 forwards the request to its parent LTS (of level 1), by replacing p with $p' = \mathcal{S}(p)$.

Again, to hide that a request is sent, the LTS of level 0 replaces one of the messages of the aggregate mapping notification (intended for its parent) with the request itself. At this point, the LTS of level 1 invokes $\mathcal{A}_{cloak}$ by passing as input the privacy requirement k, the restriction of the aggregation mapping to the positions set included in the 1-zone of its competence, and the position p'.

Since $\mathcal{A}_{cloak}$ might positively respond or fail, we have to distinguish two cases again. If $\mathcal{A}_{cloak}$ succeeds, then it returns an approximate cloaking area and the LTS of level 1 performs the request to the LBS provider. When it receives the response, filters it, and starts what we define as the *cloaking multicast* mechanism. This mechanism first requires the selection of the children LTSs of level 0 which the filtered response has to be sent to. The selection is done by finding the LTSs of level 0 which are responsible for at least one cell included in the approximate cloaking area. Then, the filtered response is multicasted to such LTSs encrypted with K. In turn, each LTS sends the response in multicast to all the users belonging to the involved cells.

Otherwise (i.e, $\mathcal{A}_{cloak}$ fails), the request is forwarded (by replacing, as usual, an aggregate-mapping-notification message) by the LTS of level 1 to its parent. The process is iterated until the root of the tree. The LBS request can be satisfied only if, eventually, an LTS is able to positively respond.

In Figure 8.3.(b), an example of an approximate cloaking area built by an LTS of level 1 is depicted. The red-line ellipse represents the cloaking area returned by the Cloaking Oracle invoked by the black LTS. The black-line region is the approximate cloaking area of the previous cloaking area.

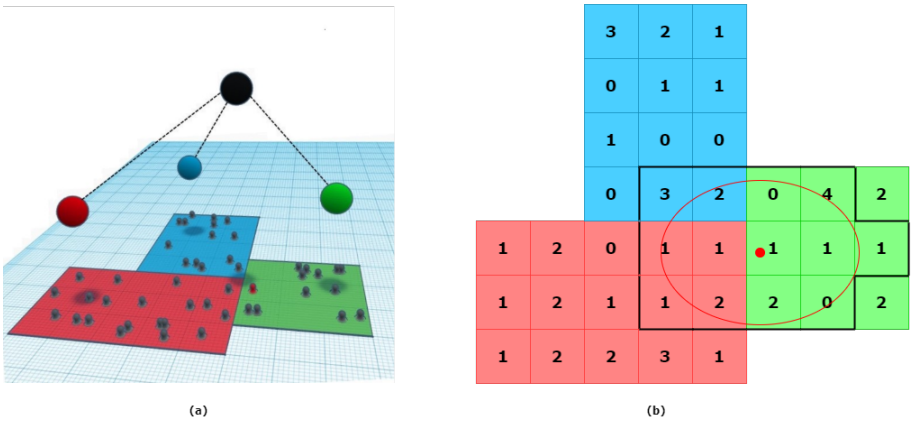


Fig. 8.3: Example of LTS hierarchy and approximate cloaking area for the LTS of level 1.

.

8.5 Service Management, LTS overlapping, and Implementation Aspects

The aim of this chapter is to provide a concrete solution based on a theoretical background to implement a real-life LTS system. Then, in this section, we provide some enhancements to the model, to capture some practical aspects. Specifically, we introduce the dimension of the *service*, to allow an LTS to be involved only in some specific services from those available in the territory.

We start by defining the role of the organizations involved in the system. The users belong to the set U. We assume that they are identified by their positions in P.

- An organization, called SP, providing the set of location-based services S_A in the area A.
- A set of organizations $L^i = \{L^i_1, \ldots, L^i_{m_i}\}$ $(0 \leq i \leq i_{max})$ interested in providing the LTS services at level i. i_{max} represents the highest level of the LTS hierarchy (possible real-life values for i_{max} are 3 or 4) and m_i represents the cardinality of the set of organizations at level i.
- The telephone service provider TSP supporting communications.

An agreement between SP and TSP exists, to implement an edge-cloud solution sketched at the end of this section. Let ST be the entity representing this agreement. Observe that this entity can play the role of global passive adversary, because it has the technological capabilities to monitor the entire traffic generated and received by the users.

We define now a number of mappings associating services with 0-zones, LTSs, and users, and defining the LTS hierarchy.

- A mapping returning the set of services on which a given LTS of level 0 works in a certain 0-zone: $services : L^0 \times A_Z \rightarrow 2^{S_A}$. To define this mapping, every LTS of level 0 establishes which services it wants to support. A given LTS of level 0 keeps only the restriction of the function *services* regarding itself. This allows us to store the information just close to the user, coherently with the edge-cloud paradigm. Obviously, this information is notified to all the registered users.
- A mapping returning the set of LTSs of level 0 working on a given 0-zone: $zero\_lts : A_Z \rightarrow 2^{L^0}$. This mapping is managed by ST, and, again, notified to all the users occurring in a given 0-zone, concerning its restriction to this 0-zone.
- A mapping returning the set of LTSs of level $i+1$ with which a given LTS of level i ($0 \leq i \leq i_{max} - 1$) has established an agreement to forward users' requests: $up_i : L^i \rightarrow 2^{L^{i+1}}$, and a mapping returning the set of LTS of level $i-1$ with which a given LTS of level i has established an agreement to receive users' requests: $down_i : L^i \rightarrow 2^{L^{i-1}}$.
- A mapping defining the set of services on which a user does not want to collaborate for the construction of the cloaking area: $deny\_s : U \rightarrow 2^{S_A}$. This mapping is defined by the user and is notified to and kept by the LTSs of level 0 to which the user is registered.
- A mapping returning, for a given LTS of level 0, say L, a service $s \in S_A$, and a 0-zone $A_j \in A_Z$, a set of positions P_j^L corresponding to users belonging to A_j of competence of L, registered with the LTS L, who did not deny the service s and $s \in services(L, A_j)$: $position : L^0 \times S_A \times A_Z \rightarrow 2^P$, where P, we recall, is the set of all the positions as defined in Section 8.3.

  The set P_j^L, referring to the 0-zone A_j, will be used by any LTS of level 0 L of competence of A_j, to build a cloaking area only for a query regarding the service s. Each LTS of level 0 keeps the proper restriction of the above mapping, which is populated thanks to the detailed position notification described in Section 8.4.2. Given a cell $z \in Z_A$ included in A_j, we denote by $P_j^L[z]$, the subset of the positions of P_j^L belonging to the cell z.

Since the LTSs of level i may require the collaboration of LTSs of level $i+1$ to satisfy a request on a given service, we require that each LTS of level i (with $i > 0$), say L_x^i, supports all the services supported by the LTSs in $down_i(L_x^i)$.

The inclusion of multiple services and the changes introduced in the LTS hierarchy do not have an impact on the mechanism by which the cloaking area is constructed, provided that this is done by selecting just the proper dimension (i.e., the service), to avoid intersection attacks. For example, suppose that in the 0-zones A_i the services s_1 and s_2 are provided, and, in the adjacent 0-zone A_j the services s_2 and s_3 are provided. Now, consider a user belonging to A_i who submits a query regarding

the service s_1 that cannot be resolved by the LTS of level 0. If we allow the LTS of level 1 to use all (independently of the service) the positions in A_i and A_j to build the cloaking area, then the adversary can infer that the actual anonymity set of users is restricted to A_i thus breaking k-anonymity. To avoid this, for a query on a service s, the positions that can be used for this service at any level of the hierarchy must come from 0-zones in which s is provided. This allows us, for the security analysis provided in Section 8.6, to consider the simpler model of a single service.

The introduction of the new features in the system model affects only the registration phase and the position notification.

8.5.1 Impact on the Registration Phase

The registration phase defined in Section 8.4.1 is affected by the inclusion of multiple services and LTS competence overlap basically due to the mapping $deny_s$ and to the fact that the user can choose the LTSs to contact for any service.

Specifically, the registration is modified as follows. Each user performs a *global* registration (in pseudonymous form) with ST and obtains a *global* pseudonymous ID.

When a user requires the collaboration of an LTS of level 0 to obtain a service, they perform a *local* registration with such an LTS and provide it with their global pseudonym. Therefore, all the LTSs of level 0 to which the user is registered will own its global pseudonymous ID. As it will be clear in the next section, we need a unique identifier for the users because the higher-level LTSs should have a global view (even in aggregate form) of the user distribution. Therefore, users who are registered with multiple LTSs of a given 0-zone, should be counted once, and then their records should be reconciled. The exact mechanism to obtain this goal without breaking users' privacy is explained in Section 8.5.2.

Given a user u belonging to a 0-zone A_j, u can perform the local registration with one or more LTSs of level 0 belonging to the set $zero\_lts(A_j)$.

For each LTS L_i^0 chosen by the u, we assume *by default* that u provides their positions to build cloaking areas for all the services in $services(L_i^0, A_j)$. Anyway, u can specify the services on which they do not want to adhere and notify them to the selected LTSs of level 0. This latter operation represents the way in which the function $deny$ is updated.

8.5.2 Impact on the Position Notification

The revision of the basic model provided in the above section has an impact on the position notification defined in Section 8.4.2. This basically happens due to the presence of multiple services and to LTSs competence overlapping.

The presence of multiple services induces a trivial change. Indeed, when a user denies a service, all the LTSs of level 0 in which the user is registered should make aware of this denial. Therefore, the position of that user is not considered for that service. Moreover, the view of the positions of the users is specific for a certain service, as induced by the definition of the mapping *positions* introduced earlier.

From now on, consider implicitly given a service s, and only those users who did not deny this service.

Concerning the detailed position notification, the only difference with respect to the basic definition is that the set of positions that is updated by an LTS of level 0, say L, working on a 0-zone A_j, is just the partial view $P_j^L = positions(L, s, A_j)$ of the overall set of positions of users belonging to A_j. Therefore, two LTSs A and B of level 0 working on the same 0-zone, may have a different view of the positions of the users occurring in this 0-zone. In other words, it may happen that $P_j^A \neq P_j^B$, because there could be a difference between the set of users registered with A and registered with B.

Also the aggregate mapping notification to the higher level is affected by the overlapping of LTSs of level 0. For instance, it might happen that on a given cell $z \in Z_A$, belonging to the 0-zone A_j, there are two LTS of level 0 of competence, say A and B. In this case, some users located in the cell z could be registered only with A, some users could be registered just with B, and other users could be registered with both A and B. Let denote by k_A (k_B, respectively) the number of users registered only with A (B, respectively) and let k_{AB} the number of users registered with both A and B. It happens that, the actual restriction of the aggregation mapping to notify to the proper LTSs of level 1 should be $k_A + k_B + k_{AB}$. Unfortunately, thanks to the detailed position notification, the only information available to A is $|P_j^A[z]| = k_A + k_{AB}$, whereas the information available to B is $|P_j^B[z]| = k_B + k_{AB}$. To overcome this drawback, the two LTSs should know the cardinality of the intersection between the $|P_j^A[z]|$ users registered with A and the $|P_j^B[z]|$ users registered with B. To avoid unwanted leakage of privacy, this task can be accomplished by using an efficient protocol for *multi-party private set intersection cardinality (MPSI-CA)*, like those proposed in [155, 76], or an approximate technique, to achieve better efficiency, like that proposed in [83] (in this case, an extra value of the privacy level k should be set to guarantee that the anonymity threshold $\frac{1}{k}$ is fulfilled). MPSI-CA is a secure multi-party-computation (SMPC) protocol, allowing any party among a group of participants, each owning a private set of items, to compute the cardinality of the intersection of these sets, without learning anything about the sets of the other participants. This way, in our case, A knows nothing about users of B (not registered to A) and vice versa.

To generalize the above case (and then, also to understand why we need to compute private set intersection cardinality among multiple parties –more than two), we introduce the following notions.

Definition 8.7. *Given a cell $z \in Z_A$, we denote by L_z the set of LTSs of level 0 supporting the service s in z. Given a set of LTSs of level 0 $X \subseteq L_z$, we define now the following recursive notion:*

$$c_\emptyset = \left|\bigcap_{Y \in L_z} P_j^Y[z]\right|$$
$$c_X = \left|\bigcap_{Y \in L_z \setminus X} P_j^Y[z]\right| - \sum_{Y \subset X} c_Y$$

The next result holds.

Theorem 8.8. *Given a cell $z \in Z_A$, and a set of LTSs of level 0 $X \subseteq L_z$, c_X represents the cardinality of the set of the users who are registered with all and only the LTSs of level 0 belonging to the set $L_z \setminus X$.*

Proof. We proceed by induction on the cardinality of the set X, by proving that the property stated in the theorem holds when $|X| \leq k$, for any $k \geq 0$.
Basis. (i.e., $|X| \leq 0$). In this case, $X = \emptyset$, and then, trivially, $c_\emptyset = \left|\bigcap_{Y \in L_z} P_j^Y[z]\right|$ represents the set of the users who are registered with all and only the LTSs of level 0 belonging to the set L_z.
Induction. We have to show that if the theorem statement holds for any set X with $|X| \leq k$, it also holds for any set X with $|X| \leq k+1$. According to Definition 8.7, $c_X = \left|\bigcap_{Y \in L_z \setminus X} P_j^Y[z]\right| - \sum_{Y \subset X} c_Y$.

The term $\left|\bigcap_{Y \in L_z \setminus X} P_j^Y[z]\right|$ counts all the users who are registered with *all* the LTSs of level 0 belonging to the set $L_z \setminus X$.

However, the above summation also includes those users who are registered also with LTSs in X. To prove induction, we have to show that c_X does not also count these users. This actually happens thanks to the subtractive second term. Indeed, by the inductive hypothesis, for any $Y \subset X$ (and, thus, $|Y| \leq k$), c_Y represents the set of the users who are registered with all and only the LTSs of level 0 belonging to the set $L_z \setminus Y$. Then, $sum_{Y \subset X} c_Y$ represents the number of users that are registered with all the LTSs of level 0 belonging to the set $L_z \setminus X$ and *at least one* LTS of level 0 in X.

Therefore, the difference $\left|\bigcap_{Y \in L_z \setminus X} P_j^Y[z]\right| - \sum_{Y \subset X} c_Y = c_X$ represents the number of users that are registered with all the LTSs of level 0 belonging to the set L_z and are not registered with any LTSs of level 0 in X. The proof is then concluded. □

Thanks to the above result, we describe now how each LTS of level 0 in L_z can compute the number of users belonging to the cell z to notify it to the higher LTS level. Observe that, this number, denoted by n_z, is independent of the LTS counting it because it counts the overall number of users not denying the service s registered with any of the LTSs in L_z.

Assume that, preliminary, in each 0-zone A_j, all the LTSs of level 0 operating on it (L_z, for each z in A_j) know each other (this task can be supported by ST). The protocol can proceed as follows, cell by cell (in parallel) in the 0-zone:

- For each subset of LTSs in L_z, the MPSI-CA (multi-party private set intersection cardinality) protocol is executed among the corresponding sets of registered users to obtain the cardinality of the intersection of such sets. The overall effort of this step is $2^{|L_z|} - |L_z| - 1$ MPSI-CA executions. Observe that, this is not prohibitive because we expect, in real-life situations, very few LTSs per 0-zone. Moreover, due to the parallel execution of the intersections, the overall computation is bounded by one MPSI-CA execution among $|L_z|$ sets.
- Each LTS in $L \in L_z$ notifies to all the LTSs of level 1 in $up_0(L)$ the following information: (1) $|P_j^L[z]|$, and (2) the result of any executed intersection in which L is involved. To avoid duplication of communications, a P2P protocol could be established among the LTSs involved. For simplicity, we omit this aspect. Eventually, each LTS of level 1 is aware of all the needed information to compute c_Xs, for each subset $X \subseteq L_z$.
- At this point each LTS of level 1 can compute n_z. This is done by choosing any LTS of level 0, say A, and then by computing: $n_z = |P_j^A[z]| + \sum_{\{Y \in 2^{L_z} \mid A \in Y\}} c_Y$. This immediately follows from Definition 8.7 and Theorem 8.8, due to the fact that, for two distinct subsets of LTSs X_1 and X_2, the corresponding c_Xs refer to disjoint sets. Therefore, the second term of the above expression counts all the users not registered with A.
- The value n_z is forwarded by the LTSs of level 1 to the higher level, and so on.

In the next example, we give an instance of the application of the above protocol to a case with three LTSs of level 0 and 1 LTS of level 1.

Example 8.9. Suppose that, for a given cell z in the 0-zone A_j, there are three LTSs of level 0 operating on it, i.e., $L_z = \{A, B, C\}$. Suppose that $up_0(A) = up_0(B) = up_0(C) = \{D\}$, that is the three LTSs of level 0 are down just 1 LTS of level 1. Now, we show how the above protocol is executed. For simplicity, we denote by $I_X = P_j^X[z]$, for any LTS $X \in L_z$. The overall number of MPSI-CAs to perform is 4 (i.e., $2^3 - 3 - 1$). They are: (1) $I_{A,B,C} = |I_A \cap I_B \cap I_C|$; (2) $I_{A,B} = |I_A \cap I_B|$, (3) $I_{A,C} = |I_A \cap I_C|$, and (4) $I_{B,C} = |I_B \cap I_C|$. Once they are executed, $I_{A,B,C}$, $I_{A,B}$, $I_{A,C}$, $I_{B,C}$, $|I_A|$, $|I_B|$, and $|I_C|$ are notified to D.

At this point, D is able to compute the following:

$$
\begin{aligned}
c_{\emptyset} &= I_{A,B,C} \\
c_{\{A\}} &= I_{B,C} - \sum_{Y subset \{A\}} c_Y = I_{B,C} - I_{A,B,C} \\
c_{\{B\}} &= I_{A,C} - \sum_{Y subset \{B\}} c_Y = I_{A,C} - I_{A,B,C} \\
c_{\{C\}} &= I_{A,B} - \sum_{Y subset \{C\}} c_Y = I_{A,B} - I_{A,B,C} \\
c_{\{A,B\}} &= |I_C| - \sum_{Y subset \{A,B\}} c_Y = |I_C| - (c_{\{A\}} + c_{\{B\}} + c_{\emptyset}) = \\
&= |I_C| - (I_{B,C} - I_{A,B,C} + I_{A,C} - I_{A,B,C} + I_{A,B,C}) = \\
&= |I_C| - I_{B,C} - I_{A,C} + I_{A,B,C} \\
c_{\{B,C\}} &= |I_A| - \sum_{Y subset \{B,C\}} c_Y = |I_A| - (c_{\{B\}} + c_{\{C\}} + c_{\emptyset}) = \\
&= |I_C| - (I_{A,C} - I_{A,B,C} + I_{A,B} - I_{A,B,C} + I_{A,B,C}) = \\
&= |I_C| - I_{A,C} - I_{A,B} + I_{A,B,C} \\
c_{\{A,C\}} &= |I_B| - \sum_{Y subset \{A,C\}} c_Y = |I_B| - (c_{\{A\}} + c_{\{C\}} + c_{\emptyset}) = \\
&= |I_B| - (I_{B,C} - I_{A,B,C} + I_{A,B} - I_{A,B,C} + I_{A,B,C}) = \\
&= |I_B| - I_{B,C} - I_{A,B} + I_{A,B,C}
\end{aligned}
$$

Now, D can compute n_z, by choosing any LTS, say A:

$$
n_z = |I_A| + \sum_{\{Y \in 2^{L^2} \mid A \in Y\}} c_Y = |I_A| + (c_{\{A\}} + c_{\{A,B\}} + c_{\{A,C\}})
$$

8.5.3 Implementation Aspects

As highlighted earlier, our hierarchical distributed LTS system enables the possibility to adopt the emergent edge-cloud architecture for mobile applications, like that presented in [264]. Indeed, the above paper is a state-of-the-art architecture that deploys cloud servers at the network edge and designs the edge-cloud as a tree hierarchy of geo-distributed servers. Therefore, this architecture perfectly fits our case. Indeed, even though the use of cloud resources appears necessary for any LTS-based solution, to serve the spatial peak loads from mobile users, our LTS organization enables a hierarchical architecture of edge cloud, which allows incremental local management of the peak loads across different tiers of cloud servers to better distribute the mobile workloads and reduce network latency. Differently from the approach presented in [264], in which a workload placement algorithm is defined to hierarchically distribute the computation load, we expect that the decentralization of computation is a *for-free* side-effect of the inherent locality of cloaking areas for even high privacy levels. Indeed, supposing to consider a 5G scenario [260], if we implement the LTSs of level 0 at the *small cells*, and then, the higher levels at the higher tiers (by using the various levels of *macro cells*), placing the highest LTSs at the (traditional) cloud, we expect that the most queries can be resolved by LTSs of low level (0 or 1), with a significant positive impact both in the overall computational workload and, importantly, from the side of service latency.

Even though, in this proposal, we do not provide an edge-cloud implementation of our system, the aim of this section is to highlight this implementation aspect as a relevant nice feature of our approach. In Section 8.7, we analyze the impact on the latency of a 4-tiers edge-cloud implementation against a traditional 1-tier implementation by simulating users' requests in different configurations, obtain a confirmation of the above expectation.

8.6 Security Analysis

In this section, we provide the security analysis of our solution. We start by defining the threat model.

Actors. We consider the following actors:

Users: They submit LBS queries.

LTSs: They process the queries of the users, build the cloaking areas and contact ST. Finally, they provide the answers to the users.

ST: It represents the agreement between TSP, providing the communication infrastructure, and SP, which receives the requests from LTSs and replies to them.

As discussed in Section 8.5, for a query submitted by a user on a service s, the positions that can be used for this service at any level of the hierarchy must come from 0-zones in which s is provided. This allows us, for the security analysis provided in this Section, to consider the case of a single service.

Assumptions. We make the following assumptions.

A1: All the messages exchanged between users and LTSs and between LTSs themselves have the same size and are encrypted by using a secure probabilistic encryption scheme.

A2: The LTSs are considered trusted.

A3: The Cloaking Oracle is reciprocal, according to Definition 8.3.

Observe that, regarding the Assumption **A2**, the LTSs of level 0 know the exact positions of the users but only limited to the 0-zone of their competence. This is the standard assumption of each LBS system employing LTSs. However, the LTSs of level $i > 0$ have less power than the LTSs of level 0. Indeed, they know only the number of users for each cell of the i-zone of their competence.

Adversaries. We consider as adversary ST playing also as a global passive adversary. Indeed, ST (1) is able to monitor all the exchanged messages (this is performed by TSP that controls the communication infrastructure), (2) can maliciously exploit the received information (this is accomplished by SP when receives the queries along with the cloaking areas), and (3) thanks to the background knowledge (SP) and/or

physical tracking of the users in the cells through the antennas (TSP), it is able to know the positions of the users in the territory.

Observe that, due to Assumption **A2** no other adversary can exist, because a user does not access any information regarding other users.

We guarantee the following security property.

SP: It is not possible for the adversary to link an LBS request with the identity of the user performing it with probability greater than $\frac{1}{k}$ (this property coincides with the classical notion of *location k-anonymity*).

It is easy to see that **SP** would be immediately broken if ST, playing as a global adversary able to monitor all the traffic in the system, can identify the source of the query. However, this is not possible according to the following reasoning. Indeed, the requests are hidden into the position-notification messages and, due to Assumption **A1** and **A2**, the adversary is not able to know when a user sends a request instead of a detailed-position-notification message. Similarly, the adversary is not able to know when an LTS forwards a request to an LTS of a higher level. However, the adversary detects that a request has been originated, because it receives such a request (forwarded by a certain LTS). The adversary can identify from where the query has been originated by observing the response. Clearly, if the coverage of the response that the adversary can track reduces someway the cloaking area, then the adversary is able to break **SP**. It is easy to see that this is not the case. Indeed, thanks to the multicast mechanism, the response messages are directed to all the users belonging to the constructed cloaking area.

Therefore, it remains to prove that our method preserves the property fulfilled by the Cloaking Oracle stated in Assumption **A3** (i.e., k-reciprocity). This guarantees Property **SP**.

To do this, recall that the cloaking area is the result of either (i) the invocation of the Cloaking Oracle $\mathcal{O}_{\mathcal{A}}$ (if the area can be constructed by an LTS of level 0) or (ii) the invocation of Algorithm $\mathcal{A}_{cloak}$. In case (i), due to Assumption **A3**, the above statement is clearly satisfied. In case (ii), the cloaking area is built by an LTS of level greater than 0. Recall that, in this case, Algorithm $\mathcal{A}_{cloak}$ returns an approximate cloaking area $\overline{C}_{P'}$ of $C_{P'} = \mathcal{O}_A(k, P', p')$ where $P' = \mathcal{R}(P)$, and $p' = \mathcal{S}(p) \in P'$ where $p \in P$ is the actual position of the user performing the request.

The next theorem states that our approximate construction-cloaking-area technique is reciprocal.

Theorem 8.10. *$\mathcal{A}_{cloak}$ is reciprocal.*

Proof. We have to prove that there exist at least k positions $p_1, \ldots, p_k \in \overline{C}_{P'} \cap P$ such that $\overline{C}_{P'} = \mathcal{A}_{cloak}(k, D_P, x)$, for each $x = \mathcal{S}(p_i)$, for any $1 \leq i \leq k$.

By Assumption **A3**, being $C_{P'} = \mathcal{O}_A(k, P', p')$, there exist at least k positions $p'_1, \ldots, p'_k \in C_{P'} \cap P'$ such that $C_{P'} = \mathcal{O}_A(k, P', p'_i)$, for any $1 \leq i \leq k$. Since $p'_i \in P'$ (for any $1 \leq i \leq k$), by definition of $\mathcal{S}$ (that is a one-to-one mapping) there exist at least k positions $p_1, \ldots, p_k \in P$ such that $p'_i = \mathcal{S}(p_i)$ for each $1 \leq i \leq k$. Since $C_{P'} = \mathcal{O}_A(k, P', p'_i)$, for any $1 \leq i \leq k$, by definition of $\mathcal{A}_{cloak}$, $\overline{C}_{P'} = \mathcal{A}_{cloak}(k, D_P, x)$, for each $x = p'_i = \mathcal{S}(p_i)$, for any $1 \leq i \leq k$. Indeed, $\overline{C}_{P'}$ depends only on $C_{P'}$. Therefore, it suffices to prove that $P_r = \{p_1, \ldots, p_k\} \subseteq \overline{C}_{P'}$. Let q be a position in P_r. We have to prove that $q \in \overline{C}_{P'}$. By definition of $\mathcal{S}$, there exists $q' \in P'$ such that $q' = \mathcal{S}(q)$ and $q, q' \in z$, for some $z \in Z_A$. Therefore, we show that $z \subseteq \overline{C}_{P'}$. By definition of $C_{P'} = \mathcal{O}_A(k, P', q')$, $q' \in C_{P'}$, and then $z \cap C_{P'} \neq \emptyset$. Therefore by definition of $S_{C_{P'}}$, $z \in S_{C_{P'}}$. Finally, by definition of $\overline{C}_{P'} = \bigcup_{y \in S_{C_{P'}}} y$, therefore $z \subseteq \overline{C}_{P'}$. The proof is then concluded. □

8.7 Experiments

Through this section, we perform a deep experimental analysis of the proposed solution to validate it in terms of both cloaking-area minimality (i.e., effectiveness) and network-performance advantages of its decentralized architecture.

8.7.1 Experimental set-up

We start by recalling that our approach is parametric with respect to any cloaking-area-construction algorithm. In the theoretical framework, we called Cloaking Oracle the underlying cloaking-area-construction algorithm. For the experiments, we consider a state-of-the-art approach, called Casper [189].

Casper is an LTS-based system aimed to achieve location k-anonymity. The Casper system architecture consists of two components: the *location anonymizer* and the *privacy-aware query processor*.

The location anonymizer plays the role of LTS. It is responsible for the construction of cloaking areas. The user, at registration time, sets a *privacy profile*, which is defined as a pair (k, A_{min}). k is the privacy level, while A_{min} represents the lower bound of the size of the returned cloaking area (too small cloaking areas can threaten users' privacy). Mobile users can change their privacy profile at any time. The anonymizer maintains the locations of the clients using a pyramid data structure, similar to a Quad-tree, in which minimum cell size corresponds to the anonymity resolution. The privacy-aware query processor is embedded inside the LBS provider. It handles anonymous queries and cloaking areas. The privacy-aware query processor returns a list of candidate answers to the location-based query via the location anonymizer. The size of the candidate list largely depends on the user's privacy profile (for example, a strict privacy profile would result in a large list of candidates). Casper is

set in our experiments with a very high granularity level (indeed, as mentioned earlier, it allows granularity modulation), in such a way that negligible approximation is introduced by Casper itself.

To execute Casper, we exploited its JAVA implementation provided in [277]. We modify this JAVA project also to implement our solution. Due to the computational effort required to conduct the experiments, to reduce the time to complete experiments, we employ three personal computers in parallel equipped with 1.8 GHz Intel i7-8850 CPU and 16 GB of RAM, 2.5 GHz Intel i7-6500 CPU and 12 GB of RAM, and 1GHz Intel i5-1035G1 CPU and 8 GB of RAM respectively. As in [297], in our experiment, the user data are generated by using the Thomas Brinkhoff data generator [34]. Furthermore, we also consider the mobility pattern, provided by the generator, to study the stability of our approach when users move into the map.

According to different metrics that we will explain next, we execute our technique by using Casper as an underlying technique, by varying different parameters. In other words, Casper plays the role of Cloaking Oracle.

We chose to conduct our experiments on a portion of the center of the city of Reggio Calabria (Italy) in which a huge presence of users is possible in real life. In fact, we also analyze the performance as the number of users varies. In Figure 8.4, the considered 2 $km \times 2$ km map provided by OpenStreetMap (OSM) [117] is depicted, including the distribution of 6000 users generated by [34] at a given instant.

As for the implementation of our approach, in favor of the clarity of the analysis, we referred to the basic tree-like LTS hierarchy, with a single-service assumption and square zones. We considered a four-layer LTSs hierarchy.

Specifically, there is a single 3-zone of size 2 $km \times 2$ km. This 3-zone is divided into four 2-zones of size 1 $km \times 1$ km. Each 2-zone includes twenty-five 1-zones of size 200 $m \times 200$ m. In turn, each 1-zones includes twenty-five 0-zones of size 40 $m \times 40$ m. Finally, we consider cells of size 8 $m \times 8$ m. Therefore, the 0-zones include 25 cells, the 1-zones include 625 cells, the 2-zones include 15,625, and the 3-zone include 62,500 cells.

Observe that such configuration of zones and LTSs is aligned to the edge-cloud implementation discussed in Section 8.5.3. Indeed, the 0-zones are of the order of 5G small cells.

8.7.2 Metrics

In our experiments, we consider three metrics: *Cloaking Area Size*, *Stability*, and *Transmission Latency*.

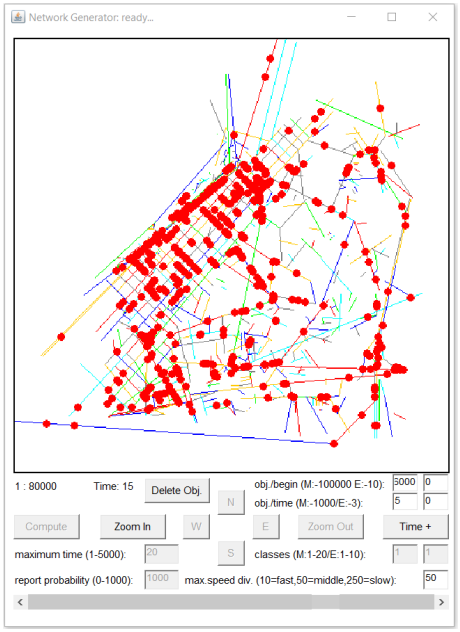
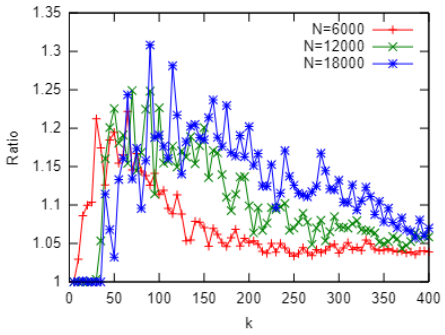
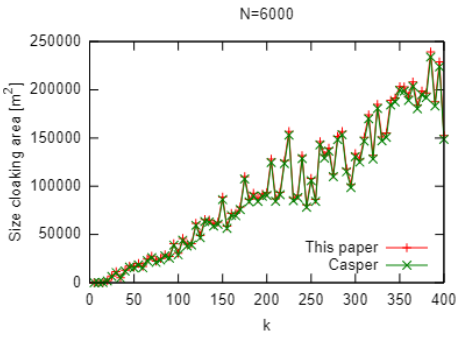


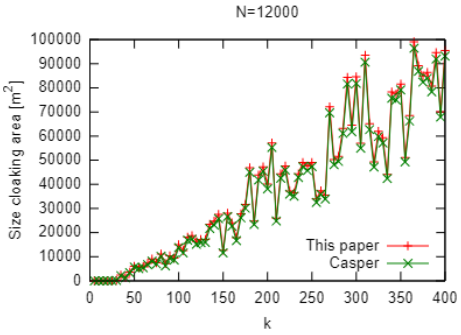
Fig. 8.4: Distribution of the users in the selected area of Reggio Calabria.



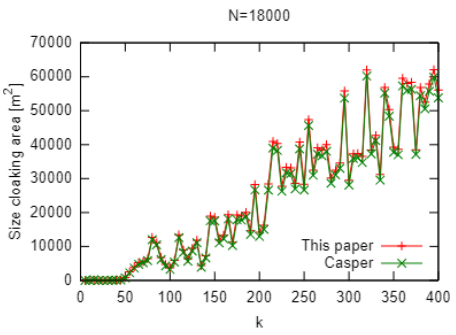
(a) Ratio between the size of the cloaking area returned by our approach and the size of the cloaking area returned by Casper as k varies.



(b) Size of cloaking area of our approach and Casper with $N = 6000$.

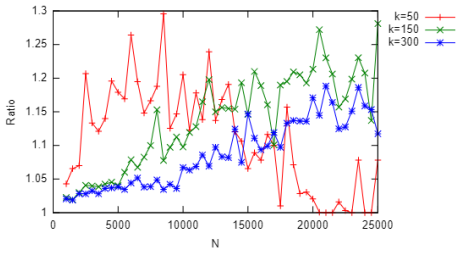


(c) Size of cloaking area of our approach and Casper with $N = 12000$.

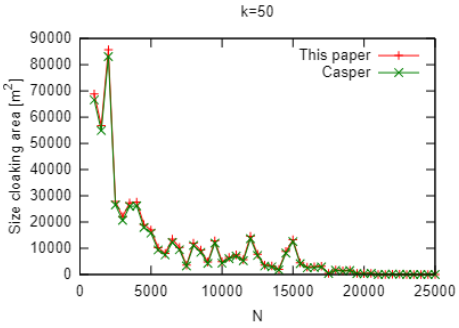


(d) Size of cloaking area of our approach and Casper with $N = 18000$.

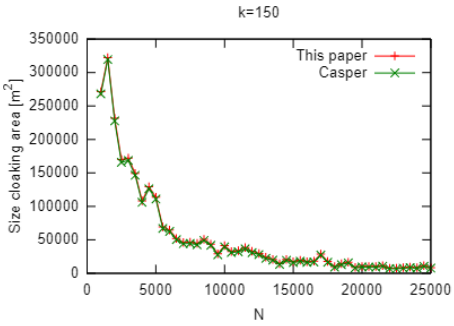
Fig. 8.5: Performance as k varies.



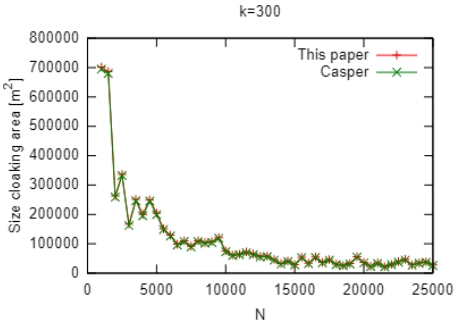
(a) Ratio between the size of the cloaking area returned by our approach and the size of the cloaking area returned by Casper as N varies.



(b) Size of cloaking area of our approach and Casper with $k = 50$.



(c) Size of cloaking area of our approach and Casper with $k = 150$.



(d) Size of cloaking area of our approach and Casper with $k = 300$.

Fig. 8.6: Performance as N varies.

Cloaking Area Size

Since, in our approach, the approximate cloaking area includes the cloaking area returned by the Cloaking Oracle (considering a redistribution of the users), we expect that our performances are worse than Casper regarding this metric. Indeed, once a privacy level k is fixed, to minimize the cloaking area including at least k users is in general aimed. In the literature, this property is called *effectiveness* [106]. As a matter of fact, large cloaking areas incur high processing overhead from the side of LBS and network cost, due to the high numbers of candidate results to return to the LTS. In our business model, coherently with what is stated in the literature [106], the users are charged depending on the required privacy level, but also on the overhead that this requirement imposes on the system.

The aim of our experiments is to show that the extra size of the cloaking area returned by our method with respect to the underlying cloaking-area-construction algorithm is very limited. Therefore, the price we pay in terms of minimality is acceptable.

First, we measured the size of the cloaking area as the privacy requirement k varies. Specifically, since the cloaking area varies according to the user performing the query, we repeat the query, for each value of k, 50 times for different users and calculate the average value. The plots in Figure 8.5a show as the ratio between the size of the cloaking area returned by our approach and the size of the cloaking area returned by Casper as k varies.

The three plots correspond to three different numbers of total users in the considered area of $2\ km \times 2\ km$. We denote by N such a number. We observe that for small values of k, the ratio is almost 1. The reason is that such queries can be resolved (in most cases) by the LTSs of level 0 and, therefore, the approximate cloaking area is equal to the cloaking area returned by Casper. As the value of k increases, the queries involve also the LTS of level $i > 0$. Thus, the ratio increases too. Anyway, as k reaches a certain value, the queries are resolved (in most cases) only by the LTSs of higher level (2 or 3). In this case, the areas returned by Casper are bigger and the approximation introduced by our cells has a smaller impact so that the ratio decreases. Regarding the differences between the total number of users on the map, we observe that, basically, the plots are translated. Indeed, if the density of the users increases for the same value of k, the queries can be resolved by lower LTSs. Therefore, as N increases, we require a greater value of k before the ratio rises from 1 and a greater value of k before the ratio decreases. We observe that the maximum ratio is about 1.3 and it is obtained for a limited range of values of k. For most of the values of k the ratio ranges from 1 to 1.1. Therefore, our performance regarding such a metric appears acceptable.

The actual size of the average cloaking areas (for our approach and Casper) as k varies are shown in the plots in Figures 8.5b,8.5c,8.5d for different values of N. Clearly, as expected, such size increases as k varies. We observe that, by fixing a value of k, the value of the size of the cloaking area for both approaches decreases as N increases. Again, this is due to the fact that the more users occurring in the map, the lower the level of the LTSs able to construct the cloaking areas is.

Dual to the first analysis, we fixed the value of k and studied the size of the cloaking area as N varies. The result is reported in Figure 8.6a.

We expect that this plot is mirrored with respect to the plot in Figure 8.5a. Indeed, for small values of N, the queries are resolved by the LTSs of higher level that results in greater areas with a smaller approximation. Therefore, the ratio assumes small values (even if greater than 1). As N increases, the LTSs of lower levels start to get involved in the resolution of the queries and the ratio increases. When N reaches a certain threshold only the LTSs of level 0 are involved, therefore the ratio approaches 1. This is evident for the plot with $k = 50$. For the other

plots ($k = 150, k = 300$), this is less evident since we considered the maximum value of $N = 25000$ and a greater value of N is required. Anyway, we did not consider $N > 25000$ since it is not realistic in the city of Reggio Calabria inside the considered area ($2\ km \times 2\ km$). For completeness, in Figures 8.6b, 8.6c, 8.6d we report the actual size of the cloaking area as N varies.

Clearly, in accordance with the plots in Figures 8.5b,8.5c,8.5d, such size increases as k increases and decreases as N increases.

Stability

Another metric we considered in our experiment is the *stability* of our approach with respect to Casper when users moving into the map (a real-life case mapping this situation is for example when the user in a vehicle wants to continuously visualize the gasoline stations available on the map).

Specifically, we follow a user in different time instants and evaluate the size of the cloaking area obtained for a query coming from this user with our approach and with Casper. We consider three mobility models of the users provided by [34]: *Slow*, *Middle*, and *Fast*.

In our experiment, we measure the cloaking area size of 80 users (randomly selected) when they move on the map. Since each user experiments a different cloaking area instant by instant (according to their position) we selected a *representative* user and show as the size of the cloaking area obtained with our approach and with Casper, varies in the time domain.

We consider two approaches to selecting the representative user. The first representative user is an *average* user, in which the cloaking area at each instant is given by the average of the cloaking area of the 80 users at the same instant. The limit of this approach is that the average user is not any real user and could obtain cloaking areas very different from any real user.

Another approach to select the representative user is to find a real user who experiments a median standard deviation σ of the distribution of ratios (in the time) between the size of the cloaking area returned by our approach and the size of the area returned by Casper. This way, we select a user well representing the sample from the stability point of view because the above standard deviation is a measure of stability. Indeed, a lower value of σ, means that the user obtains similar ratios (and then, probably, similar cloaking area for our approach and Casper) as the time varies. After filtering the data by removing outlier users, we selected the *median* user as those with the median σ (if the number of users was even, we selected one of the two users with σ nearest to the median σ).

For completeness, we considered both the above representative users, namely, average and median.

The result of both the approaches is shown in Figures 8.7a, 8.7b, 8.7c, 8.8a, 8.8b, and 8.8c for different values of k and $N = 6000$.

We observe that, as expected, the plots in the Slow scenario are less variable (and then more stable) than the plot in the Middle and Fast Scenario for both Casper and our approach. This is due to the fact that the distribution of the users does not change rapidly in the time and then the cloaking areas obtained are similar. The plots are more stable with higher values of k since they involve, in most cases, the higher LTSs, which return similar cloaking areas due to the exponential growth of the areas in Casper.

For the median user, we observe that they maintain the same area for some intervals and then suddenly change. This is due to the discontinuity arising when the user crosses different zones. This effect is smoothed for the average user.

Finally, to confirm the previous considerations, we show the average relative standard deviation of the size of the cloaking area experimented by the 80 users moving into the maps. The result is shown in Figures 8.9a, 8.9b, and 8.9c for different values of k and $N = 6000$.

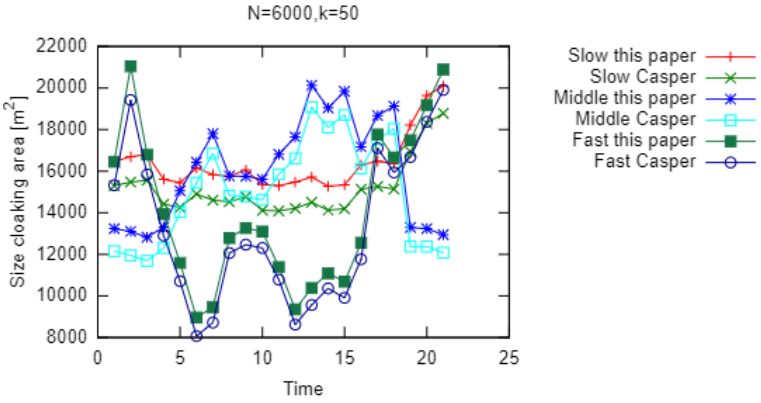
As expected, no appreciable difference exists between our approach and Casper. Our method slightly outperforms Casper (since we have a lower average relative standard deviation) due to our approximation. Indeed, a little modification in the Casper cloaking area does not result in a modification of our cloaking area.

Clearly, the Slow scenario is more stable than the middle scenario, which is more stable than the fast scenario.

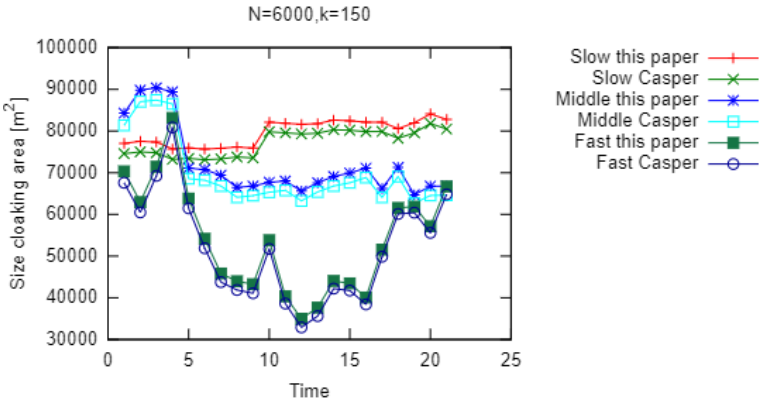
Finally, the scenarios with higher values for k are more stable than the scenarios with lower values for k.

Advantage of the Decentralized Implementation

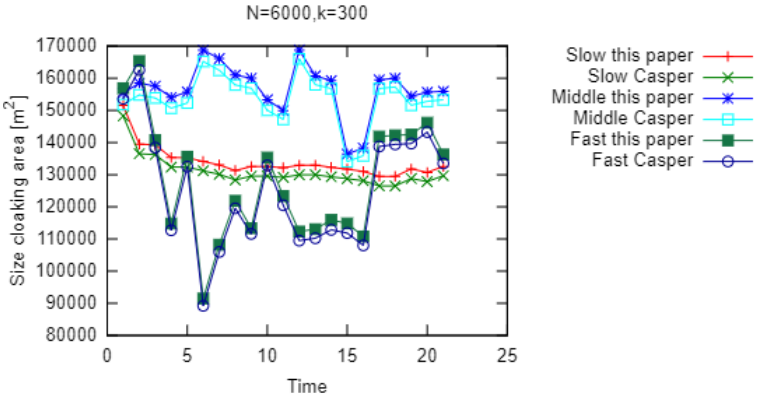
The last considered metric is the network latency required to solve the queries. The aim of this experiment is to provide a first validation of the advantages that our decentralized solution can give when an edge-cloud implementation is adopted. As recently reassessed in [75], in today's Internet, although there can be considerable delay variation over very short time scales, end-to-end latencies can be considered *operationally constant* on long timescales (e.g., the order of a day). Latencies can be considered operationally constant if they remain within bounds that could be considered operationally equivalent. With no the ambition to give a conclusive evaluation of an edge-cloud implementation of our solution (not provided in this paper),



(a) Size of the cloaking area of the average user as the time varies with $k = 50$ and $N = 6000$.

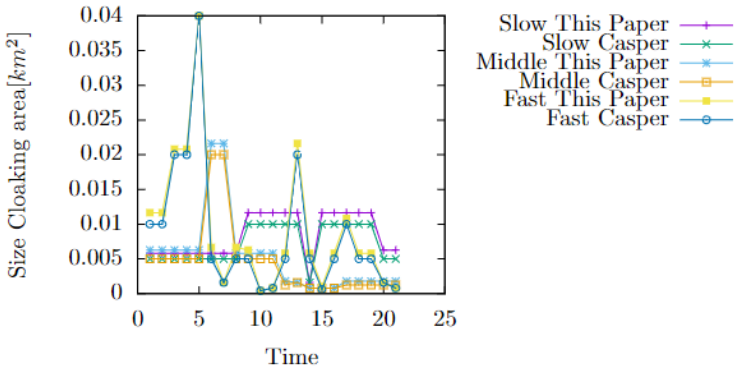


(b) Size of the cloaking area of the average user as the time varies with $k = 150$ and $N = 6000$.

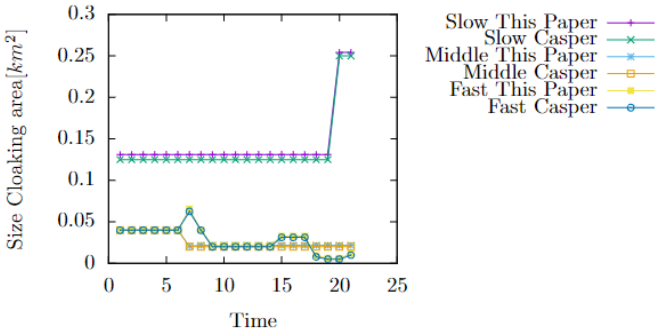


(c) Size of the cloaking area of the average user as the time varies with $k = 300$ and $N = 6000$.

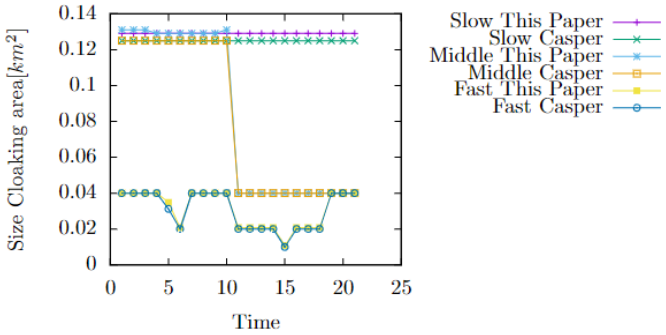
Fig. 8.7: Average User.



(a) Size of the cloaking area of the median user as the time varies with $k = 50$ and $N = 6000$.



(b) Size of the cloaking area of the median user as the time varies with $k = 150$ and $N = 6000$.

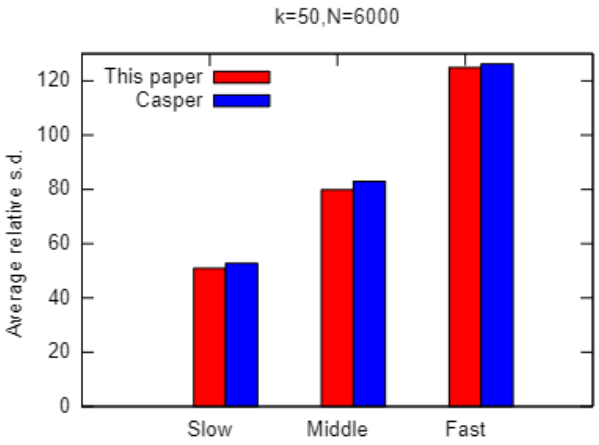


(c) Size of the cloaking area of the median user as the time varies with $k = 300$ and $N = 6000$.

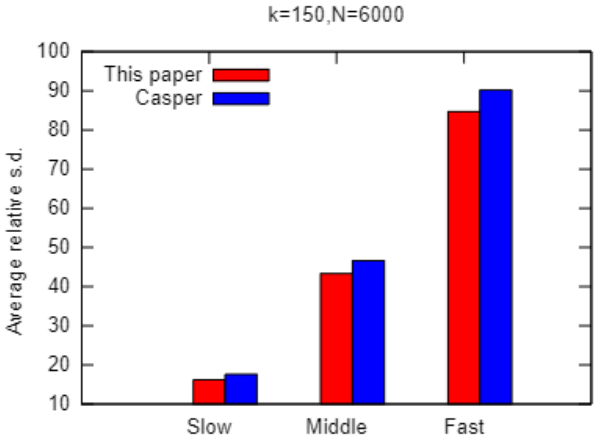
Fig. 8.8: Median User.

but only to have a validation of this implementation direction, we study by simulation the network latency, according to the above results, by setting the bounds to 10-20 ms, 50-80 ms, 100-150 ms, and 200-300 ms for queries resolved by the LTSs of level 0, 1, 2, and 3, respectively. Indeed, we recall, we consider an edge-cloud implementation based on hierarchic tiers like that proposed in [264].

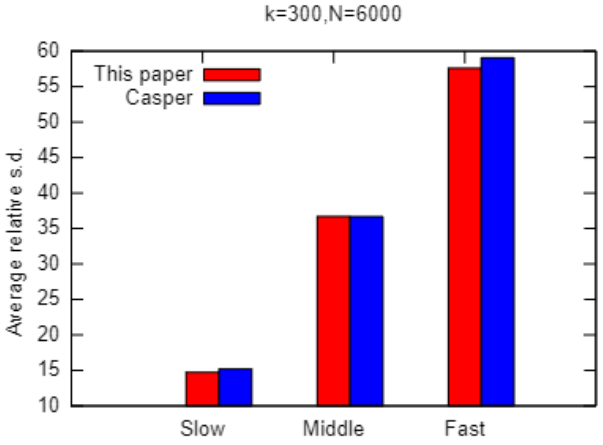
Regarding Casper, we consider a unique LTS of level 3 of competence of the entire zone that solves all the requests. Regarding our approach, if a request cannot be



(a) Average relative standard deviation with $k = 50$ and $N = 6000$.

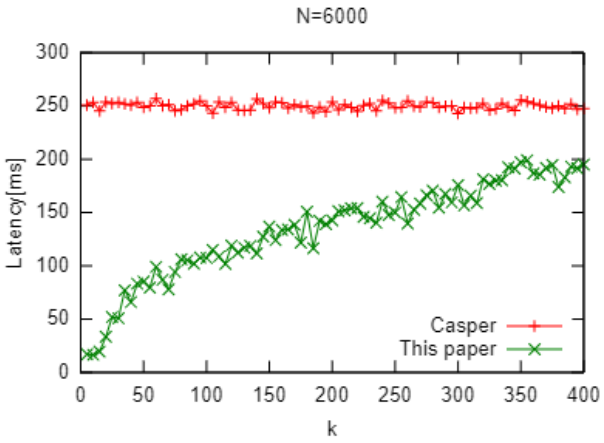


(b) Average relative standard deviation with $k = 150$ and $N = 6000$.

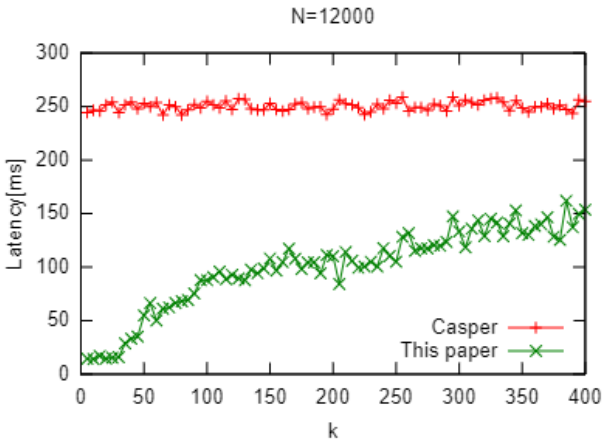


(c) Average relative standard deviation with $k = 300$ and $N = 6000$.

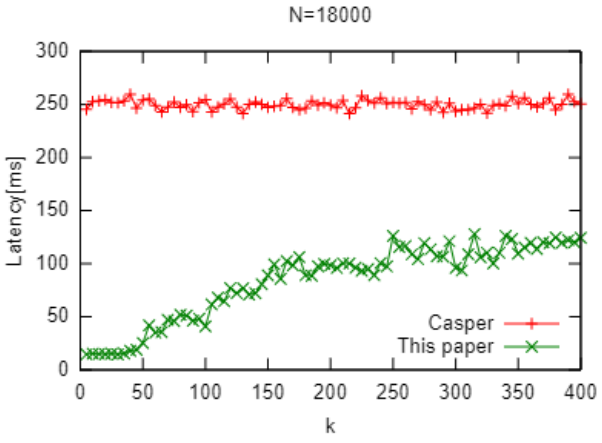
Fig. 8.9: Average relative standard deviation.



(a) Average transmission delay as k varies with $N = 6000$.



(b) Average transmission delay as k varies with $N = 12000$.



(c) Average transmission delay as k varies with $N = 18000$.

Fig. 8.10: Median User.

resolved by an LTS of level i, it is forwarded to an LTS of level $i+1$ and the total latency is the sum of the single latencies associated with the level i and $i+1$ (it is slightly overestimated in favor of the fairness of the experiments. Indeed, the hierarchical edge-cloud architecture might allow lower end-to-end latency when an LTS of level $i+1$ is contacted by an LTS of level i instead of directly the user).

The average latency as k varies is reported in Figures 8.10a, 8.10b, and 8.10c for three values of N (i.e., 6000, 12000, 18000).

Clearly, for Casper, the delay is constant as k and N vary. Regarding our approach, the latency increases with k and decreases with N since higher values of k or lower values of N involve higher-level LTSs to resolve the queries.

We observe that, in the plots, for all the values of k considered, our approach outperforms Casper. This occurs since the LTS of level 3 is never involved to build a cloaking area. Anyway, higher values of k are not required in real-life applications.

8.8 Related Work

In this section, we review the literature related to the subject of our work. Our proposal is related to the wide literature existing in the field of privacy-preserving location-based services (LBS). In this field, different types of protection techniques are available in the literature depending on the asset to be protected, as described in [28, 29, 160, 145, 292]. In detail, it is possible to categorize the LBS services, according to the technique used, into *user's location*, *query content* and *user's identity*.

In the case of user's location, the goal is to protect the location information through a perturbation on location data or by hiding realistic data within dummy information [152, 294, 201]. In [281], the authors emphasize the importance of decentralized approaches, in which secret sharing is used to obtain position obfuscation. The proposed techniques aim to ensure, however, a good quality of the service despite the obfuscation allowing to contrast also inference attacks. A specific focus on inference attacks is given in [254, 161]. A type of perturbation used to obtain obfuscation is enlarging, adopted in many works available in the literature [70, 62, 10, 121, 214].

Query content techniques have the purpose to protect the content of LBS queries. There exist many proposals belonging to this category [222, 283]. Also this approach is prone to inference attacks. Differential-privacy or machine-learning-based techniques [8, 285, 232, 200, 90] exist to increase robustness of obfuscation and perturbation techniques against inference attacks. Other approaches, giving privacy guarantees, are based on PIR (Private Information Retrieval) [101].

The last category of techniques is user's identity, which is the category our paper belongs to. In this case, by making sufficiently anonymous the user, the sensitivity of the query does not threaten their privacy. In the literature, the approach well-consolidated aimed to protect the user's identity is based on *cloaking areas* which are zones that include at least k users and allow us to achieve *location k-anonymity* [113]. Some very recent papers using this approach are [302, 1, 287, 304, 95, 141].

Our work falls within this type of technique, and, in particular, in those reaching this goal by using a Trusted Third Party, called Location-Trusted Service (LTS) [240, 113, 216]. This approach is explained in detail in Section 8.2. To have effective location k-anonymity, it is necessary to satisfy *reciprocity* [106, 63] that is fulfilled if the returned anonymity set includes at least k users whose anonymity set is the same. A considerable effort has been devoted (also recently) in the literature [280, 16, 115] to obtain cloaking areas with robust privacy guarantees.

A different class of techniques aimed to obtain cloaking areas are those based on *density services* [278, 140]. In this case, no trusted party is involved (thus, the LTS does not exist). The users, in collaboration, relying on public services making available the distribution of people in the territory, construct autonomously the cloaking area. Then, through an anonymous communication network, submit the query along with the cloaking area directly to the LBS provider. Filtering the response is also in charge of the users. Observe that, the density-service-based system proposed in [140], is hierarchical too. Anyway, this class of techniques is not comparable with the LTS-based techniques, for evident reasons.

Despite the fact that a number of LTS-based hierarchical approaches exist in the literature [166, 66], this aspect is restricted to the cloaking-area-construction algorithm. In our case, the hierarchy is at the basis of a multi-organization distributed implementation of the LTS system. To the best of our knowledge, the only existing proposal of distributed LTS system is given in [306]. However, there are many differences with our approach.

First, the solution proposed in [306] is not hierarchical and requires the presence of a *secure comparison server* and a *directory server*. Furthermore, the user has to actively participate in the protocol by encrypting some data. Another difference is that even though the authors enable location broker overlapping, each user can register only with a single broker. On the other hand, in our approach, we allow users to register with multiple LTSs (playing the role of location broker). Finally, in [306], no explicit mechanism is provided to protect the anonymity of the communication against a global adversary able to observe the flow of exchanged messages.

Concerning the global adversary threat model, the only existing approach is based on P2P collaboration among users to mix LBS requests and prevent traffic

analysis ([179] is a paper well representing this approach). Other P2P approaches to resolve the single-point-of-failure problem of the LTS have been proposed [104]. However, in our paper, we refer to scenarios in which P2P collaborative approaches can be considered little realistic since users are in general not inclined to open their device to incoming anonymous traffic. Moreover, the mechanism of incentives is not effective. The only non-P2P previous approaches rely on a further Trusted Third Party (TTP) [216, 301] (called *Function Generator* and *Converter*, resp.). Even though these approaches could be in principle combined with our technique, the authors themselves highlight the serious drawback that both the LTS and these additional TTPs are not allowed to collude with malicious users, thus forgetting that any entity can always play the role of user.

9

Anonymous service delivery with accountability guarantees

Anonymous service delivery has attracted the interest of research and industry for many decades. To obtain effective solutions, anonymity should be guaranteed against the service provider itself. However, if full anonymity of users is implemented, no accountability mechanism can be provided. This represents a problem, especially when referring to scenarios in which a user, protected by anonymity, may perform illegally when leveraging the anonymous service. In this chapter, we propose a blockchain-based solution to the trade-off between anonymity and accountability. In particular, our solution relies on three independent parties (one of which is the service provider itself) such that only the collaboration of all three actors allows the disclosure of the real identity of the user. In all the other cases, anonymity is guaranteed. To show the feasibility of the proposal, we develop a prototype with user-friendly interfaces that minimize client-side operations. Our solution is then effective also from the point of view of usability. The results of this solution are published in a research paper [45].

9.1 Introduction

Anonymous services are privacy-preserving services offered to users without requiring them to disclose their identities.

We can distinguish two types of services. The first type is represented by *one-time* services exploited by users just once. Some examples are electronic auctions [119, 173], anonymous surveys [128], or e-voting [52]. In the second type of services, users keep an anonymous account and exploit the same service more times. Each user is associated with a pseudonymous username to which their account's activity is linked. The action of users includes also possible data generated by the user when leveraging the service. A typical example is represented by anonymous social networks [276, 103], in which a pseudonymous username is associated with some data such as private messages or posts. In this chapter, we refer to the second type of services.

A minimal requirement is to provide anonymity to the users with respect to other users leveraging the same service. However, to be effective, anonymity should be guaranteed also against the service provider itself.

Even though, for the first type of services, full anonymity may be required, when referring to the second type, it is impractical to pursue user anonymity without taking accountability into consideration. Indeed, without the fear of being identified, held responsible, and punishable when they abuse the services, users are likely to misbehave due to selfishness or malice, thereby disrupting system operations and harming everyone else [132]. Therefore, a trade-off between anonymity and accountability exists [92].

In this chapter, we try to solve this trade-off by proposing a blockchain-based protocol including two further parties in addition to the user and the service provider. They are: (1) an identity provider, which knows the real identity of the user, and (2) a *linkage agency*, acting as a third independent party, which allows the re-identification of the users if needed (e.g., when required by a court order).

In particular, our solution allows the identification of the users only when all three parties (i.e., service provider, identity provider, and linkage agency) collaborate. In all the other cases, anonymity is guaranteed. The present solution reaches similar goals of approaches like [47, 44, 236] with the relevant difference that, in our solution, the collaboration of just two third parties is not enough to de-anonymize the user.

From the security point of view, in our threat model, we require only a minimum degree of trust in the identity provider, in the sense that it does not perform active attacks such as impersonation. However, we allow it to passively collude with at most one of the other entities by disclosing the information it knows (i.e., the real identity of the users) while preserving unlinkability. No trust is required in the service provider and in the linkage agency.

At the basis of our solution, we rely on a challenge-response mechanism leveraging a smart contract deployed on the Ethereum blockchain. This smart contract also provides an account recovery mechanism in case the user loses their access credentials. Indeed, traditional recovery mechanisms based on the telephone number or email address cannot be adopted, since they would reveal personal information about users.

To show the applicability of our proposal, we provide a user-friendly implementation including a time-cost analysis. In this implementation, we care about usability by reducing the number of operations performed client-side. This shows that our solution can be easily applied in real-life contexts.

9.2 Background

In this section, we provide some background notions about blockchain technology with a focus on Ethereum.

Blockchain is a peer-to-peer network that keeps track of the occurrence of events. An entity can generate a transaction towards another entity to exchange a value. This transaction is validated by peers participating in the network, thus without requiring any third-trusted party to validate the transaction. This represents the main advantage offered by this technology. Other features are:

- The transactions have to be validated and cannot be modified after their validation.
- Users cannot repudiate a transaction that they had generated.
- Anyone can access and verify the transactions stored on the blockchain.
- The users generating the transactions should remain anonymous.

Regarding the latter point, for example, Ethereum guarantees pseudonymity, in the sense that the identity of the users is not revealed, but their transactions are linkable.

The above properties are generic of the blockchain technology and apply to almost all the existing *public-permissionless* implementations [25, 265], even though there exist blockchains supporting full anonymity [191, 131].

To give a concrete proposal, in the proposed solution, we refer to the Ethereum blockchain.

In Ethereum, there are two types of *accounts*. The first type is represented by the *External Owned Accounts* (EOAs), which are controlled by users through private/public key pairs. Specifically, a user U generates a random string of 32 bytes. It represents the private key PRK_U of U.

Ethereum leverages the Elliptic Curve Digital Signature Algorithm (ECDSA) [147], by selecting the curve `secp256k1`. Then, by applying the ECDSA algorithm to PRK_U, U obtains a public key PBK_U of 64 bytes (it represents the coordinates of a point of the elliptic curve). Finally, U applies the hash function `Keccak256` [27] to PBK_U and takes the last 20 bytes to obtain an *Ethereum address*. This Ethereum address is used to send and receive transactions and identifies (in pseudonymous form) U in the network.

Each transaction generated by U is signed with the private key PRK_U. The signed transaction is broadcast in the blockchain network. Anyone can verify the signature, by retrieving the public key of the signer (from the signature and the transaction) and checking that it is equal to the expected public key. Furthermore, the Ethereum

address of the signer can be obtained from the public key by applying Keccak256 as described above.

The second type of account is called *contract account*, and represents an account controlled by code (i.e., a *smart contract*). A smart contract [220] can be viewed as a collection of data and functions. To deploy an *instance* of a smart contract, an EOA sends an Ethereum transaction containing the compiled code (on the Ethereum VM) of the smart contract without specifying any recipient. Each instance has its own data (also called *state*) and functions, and it is associated with an Ethereum address. To invoke a function of the smart contract, an EOA generates a transaction including the input of the function, intended for the address of an instance of the smart contract. This function may change the state of such an instance. Since the transaction invoking the function is stored on the blockchain, due to the properties of immutability and non-repudiation, anyone can verify the correct state of any instance at any time.

9.3 Actors and notation

Our solution involves four actors:

- The user U: they require access to an anonymous service offered by a service provider SP.
- The service provider SP: it offers the anonymous service to U. U interacts with SP through a pseudonym *username* $user_U$ not linkable to the *real identity* ID_U of U.
- The identity provider IP: it knows and manages the real identity ID_U of U.
- The linkage agency LA: it is a third independent party that, after receiving an order by a court, is able, through the collaboration of IP and SP, to link the real identity ID_U with the username $user_U$.

As discussed in Section 9.1 and shown in Section 9.7, just the collaboration of two out of IP, SP, and LA is not enough to associate ID_U with $user_U$, but all the three parties have to be involved.

During the interaction between U and SP, some information is generated and associated with $user_U$ (since SP does not know the real identity ID_U). We denote by I_U such *account information*. For example, if SP is an anonymous social network, I_U is represented by the private messages exchanged between U and other users, the messages posted by U, the “likes" released by U, and so on. Like almost all web services, I_U is maintained server-side by SP. Observe that the access to I_U is what the user has to recover in case of loss of credentials.

An association among EOAs (see Section 9.2) and actors is established. Specifically, U owns two Ethereum addresses Add_1^U and Add_2^U. SP owns the Ethereum address Add^{SP}. IP owns the Ethereum address Add^{IP}. Finally, LA owns the Ethereum address Add^{LA}.

Observe that Add_1^U and Add_2^U are pseudonyms of U not linkable between them or with ID_U. Add^{SP}, Add^{IP}, Add^{LA} are made publicly available by SP, IP, and LA, respectively.

As we will see in Section 9.4, our proposal is based on a challenge-response mechanism implemented by the smart contract reported in Listing 1. A number of instances of such a smart contract are deployed by SP, IP, and LA.

Specifically, SP deploys two instances C_1^{SP} and C_2^{SP} with Ethereum addresses $Add^{C_1^{SP}}$ and $Add^{C_2^{SP}}$, respectively. C_1^{SP} is used by U during the registration phase with SP. It allows SP to associate the Ethereum address Add_2^U with $user_U$. Clearly, this operation is performed by SP without knowing the real identity ID_U of U. Moreover, $user_U$ is not publicly disclosed. C_2^{SP} is used by U to retrieve I_U, in case U loses the credentials to connect with SP.

IP deploys the instance C^{IP} with Ethereum address $Add^{C^{IP}}$. It is used by IP to associate Add_1^U with the real identity ID_U of U without disclosing it.

Finally, LA deploys the instance C^{LA} with Ethereum address $Add^{C^{LA}}$. C^{LA} is used by LA to link the Ethereum addresses Add_1^U and Add_2^U of U (without knowing the real identity ID_U and the username $user_U$).

To conclude this section, we introduce a notation to model the Ethereum transactions. In our application, the transactions are generated by the actors only towards instances of a smart contract to invoke some functions. No user-to-user transaction is performed and no Ether transfer is needed. Therefore, we model an Ethereum transaction as a tuple $T = \langle sender, destination, function(params) \rangle$.

$sender \in \{Add_1^U, Add_2^U, Add^{SP}, Add^{IP}, Add^{LA}\}$ represents the Ethereum address of the actor generating the transaction. $destination \in \{Add^{C_1^{SP}}, Add^{C_2^{SP}}, AddC^{IP}, AddC^{LA}\}$ represents the destination address of the transaction. It can be an instance of the smart contract. Finally, $function(params)$ represents the invoked function of the smart contract along with the input parameters.

9.4 Challenge-Response mechanism

At the basis of our protocol, there is a challenge-response mechanism implemented by the smart contract reported in Listing 1.

It offers the following three security properties.

Listing 1 Smart Contract implementing a challenge-response mechanism.

```
contract SmartContract {

  address private owner;

  mapping(address => string ) public stateMap;
  mapping(address => bytes32 ) challengeMap;

  modifier isOwner() {
   require(msg.sender==owner, "Caller is not owner");
    _;
  }

  constructor() {
    owner=msg.sender;
  }

 function setChallenge(address user, bytes32 challenge) public isOwner {
    stateMap[user]="To Confirm";
    challengeMap[user]=challenge;
  }

  function solveChallenge(bytes32 solution) public{
    address sender=msg.sender;
    if(compareStrings(stateMap[sender],"To Confirm")){
      bytes32 challenge=challengeMap[sender];
      if(keccak256(abi.encode(solution))==challenge){
           stateMap[sender]="Confirmed";
            }
        }
    }

  function compareStrings(string memory a, string memory b) public view
  returns (bool) {
    return (keccak256(abi.encodePacked((a))) ==
    keccak256(abi.encodePacked((b))));
  }
}
```

- **P1**: IP, SP, LA have to be sure that the users really own the claimed Ethereum addresses.
- **P2**: if the previous check is satisfied and the owner of an address is able to solve a given challenge, then the smart contract sets, in a verifiable way, the state of such an address as `"Confirmed"`.
- **P3**: the smart contract allows us to notarize some information.

Even though, at this stage of the chapter, it is not clear why we need these properties, we show as the smart contract allows us to satisfy them. Further details are provided in Section 9.5, where we exploit the proposed mechanism to implement our protocol.

Our smart contract includes two mappings, called `stateMap` and `challengeMap`, and two functions, called `setChallenge` and `solveChallenge`.

`stateMap` associates an Ethereum address with a string representing the state of such an address. We admit three possible states: `""` (the built-in default state of Ethereum smart contracts), `"To Confirm"`, and `"Confirmed"`. By default, all the addresses are in state `""`.

`challengeMap` associates an Ethereum address with a `bytes32` variable containing the challenge that such an address has to solve.

`setChallenge` can be invoked only by the owner (i.e., the entity who deployed) of the instance of the smart contract (in our application, the owner can be IP, SP, or LA). It receives as input an Ethereum address and a challenge. Then, it associates the challenge with the address in the `challengeMap` and sets the state of the address to `"To Confirm"` in the `stateMap`.

`solveChallenge` can be invoked by anyone and receives the solution to the challenge as input. The solution to the challenge is a 32-byte word (corresponding to the digest of a cryptographic hash function). The technical detail about how this solution is computed is described in Section 9.5. First, it checks that the address originating the transaction invoking this function is in state `"To Confirm"`. This guarantees **P1**, since only the owner of an Ethereum address can generate a transaction from such an address. If the check is positive, then the function verifies that the solution solves the challenge associated with this address. In the positive case, the state of the address is set to `"Confirmed"` in the `stateMap`. This clarifies as the smart contract guarantees **P2**.

The mechanism to verify the solution to the challenge is straightforward. The smart contract simply checks that $challenge = Keccak256(solution)$.

Regarding **P3**, the solution to the challenge itself represents the information to notarize. Observe that, since this information is published in plaintext on the

blockchain, it should not compromise the anonymity of the user. This will be explained in the next section, in which the whole protocol is described.

9.5 The proposed approach

Our solution involves three interactions performed just once (in a registration phase) by U with IP, SP, and LA, respectively. The goal is to obtain accountability only if LA collaborates with SP and IP to discover the real identity of a user performing illegally when using the anonymous service offered by SP. Furthermore, the proposed approach provides a mechanism to recover the account information I_U when U loses the access credentials.

9.5.1 Interaction between U and IP.

Through this interaction, IP associates the Ethereum address Add_1^U with the real identity ID_U of U. At the end of this phase, Add_1^U will result in state `"Confirmed"` and will be used by LA in the next interaction. This phase proceeds as follows.

First, U authenticates with IP and provides it with Add_1^U. Then, IP picks a random R and generates a challenge $ch = Keccak256(Kekkac256(R||ID_U))$. We recall that U is preliminarily registered with IP, which knows their real identity ID_U.

IP generates a transaction $T_1 = \langle Add^{IP}, Add^{C^{IP}}, setChallenge(Add_1^U, ch)\rangle$ intended for the instance C^{IP} of the smart contract. This transaction invokes the function `setChallenge` that associates the challenge ch with the address Add_1^U and sets Add_1^U in state `"To Confirm"`. Finally, IP sends U the random R.

After receiving R, U computes the solution to the challenge as $sol{=}Kekkac256(R||ID_U)$ and generates the transaction $T_2{=}\langle Add_1^U, Add^{C^{IP}}, solveChallenge(sol)\rangle$. This proves to IP that U is the owner of Add_1^U. The function `solveChallenge` sets Add_1^U in state `"Confirmed"`. Observe that the random R acts as a salt for the hash function $Kekkac256$. This way, ID_U cannot be reversed by sol even performing dictionary attacks by testing all the possible real identities. On the other hand, if IP discloses R to LA, the association $ID_U - Add_1^U$ can be verified through the instance C^{IP} of the smart contract.

Finally, IP verifies the state `"Confirmed"` of the address Add_1^U and stores locally the tuple $\langle Add_1^U, ID_U, R\rangle$ to provide LA if needed.

The above steps are summarized in the sequence diagram of Figure 9.1.

The next two interactions are between U and LA and between U and SP, respectively. They have to be performed in an anonymous form without any authentication of U. Clearly, to avoid IP geolocation, anonymous communication protocols like Tor

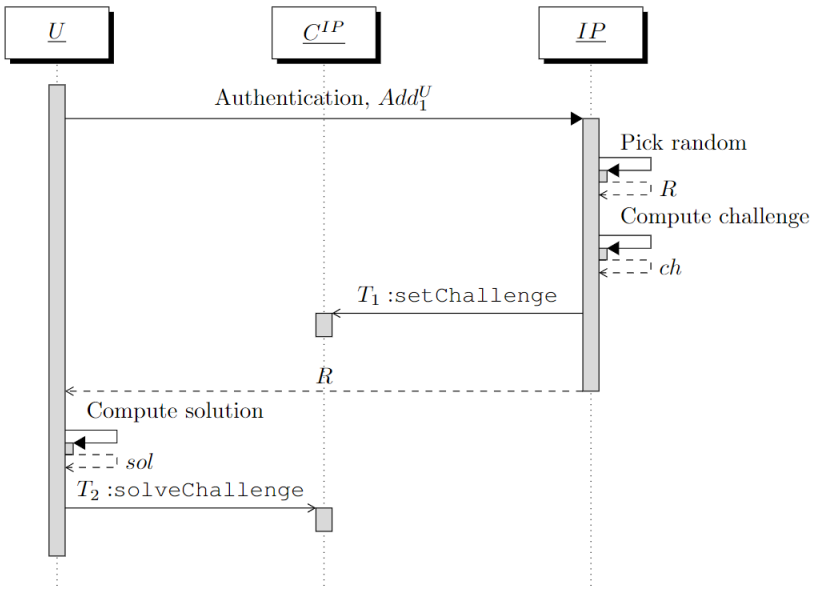


Fig. 9.1: Sequence diagram of the interaction between U and IP.

[258] and VPNs [234] can be adopted. We do not treat this aspect in detail. For further details about the anonymous communication protocols, the reader can refer to parts 1 and 2 of this thesis.

9.5.2 Interaction between *U* and *LA*

Through this interaction, LA links Add_1^U and Add_2^U without knowing the real identity of U.

First, U contacts (anonymously) LA and provides it with Add_1^U. LA needs to verify two conditions. The first is that Add_1^U is an Ethereum address associated with a real identity by IP. However, LA does not need to know such a real identity. This condition can be easily verified by checking that Add_1^U is in state `"Confirmed"` on C^{IP}. The second condition is that Add_1^U really belongs to the user contacting LA. This is done to avoid a user trying to impersonate another user already authenticated with IP.

Even though the challenge-response mechanism used in the previous section allows this task, we should not use it. Indeed, it requires some interactions through the blockchain that would disclose the linkage $Add_1^U - Add_2^U$ to anyone.

Therefore, to verify the second condition, LA generates a random R' and provides U with it. U signs R' with the private key associated with Add_1^U, obtaining a signature σ. Finally, U replies to LA with the pair (σ, Add_2^U).

Starting from σ and R', LA verifies the signature and retrieves the public key associated with Add_1^U. Then, as explained in Section 9.2, LA computes the hash func-

tion Keccak256 on this public key and takes the last 20 bytes to obtain the address $\overline{Add_1^U}$. If $\overline{Add_1^U} = Add_1^U$, then LA is sure that the user contacting it owns Add_1^U.

At this point, LA needs to verify that U owns Add_2^U. The same challenge-response mechanism used with IP is adopted. This way, Add_2^U will move to the state "Confirmed" on the instance C^{LA} and will be used by SP in the next interaction with U.

In detail, LA picks a random $\bar{R}$ and generates a challenge $ch = Keccak256(Kekkac256(\bar{R}||Add_1^U||Add_2^U||\sigma))$.

Then, LA generates a transaction $T_3 = \langle Add^{LA}, Add^{C^{LA}}, setChallenge(Add_2^U, ch)\rangle$. This transaction invokes the function setChallenge that associates the challenge ch with the address Add_2^U and sets Add_2^U in state "To Confirm". The random $\bar{R}$ is provided to U.

U computes the solution of challenge as $sol = Kekkac256(\bar{R}||Add_1^U||Add_2^U||\sigma)$ and generates the transaction T_4=$\langle Add_2^U, Add^{C^{LA}}, solveChallenge(sol)\rangle$. This proves to LA that U is the owner of Add_2^U, which will be set in state "Confirmed" by the function solveChallenge.

Finally, LA verifies the state "Confirmed" of the address Add_2^U and stores locally the tuple $\langle Add_1^U, Add_2^U, \bar{R}, \sigma\rangle$.

The above steps are summarized in the sequence diagram of Figure 9.2.

9.5.3 Interaction between *U* and *SP*

Through the third interaction of our protocol, SP associates Add_2^U with a pseudonym $user_U$ chosen by U. For simplicity, we assume that the access credentials of U are represented by a pair username-password. However, the password can be replaced by the DeviceID of the user's phone as with Whisper [276].

Again, we exploit the challenge-response mechanism of Section 9.4.

U provides SP with $(Add_2^U, user_U, pass_U)$, where $pass_U$ is the password chosen by U. SP verifies Add_2^U is in state "Confirmed" on the instance C^{LA}. Then, it picks a random $\hat{R}$ and generates the challenge $ch = Keccak256(Kekkac256(\hat{R}||user_U))$. Finally, SP generates a transaction T_5=$\langle Add^{SP}, Add^{C_1^{SP}}, setChallenge(Add_2^U, ch)\rangle$. This transaction invokes the function setChallenge that associates the challenge ch with the address Add_2^U and sets Add_2^U in state "To Confirm". The random $\bar{R}$ is provided to U.

U computes the solution to the challenge as $sol = Kekkac256(\hat{R}||user_U)$ and generates the transaction $T_6 = \langle Add_2^U, Add^{C_1^{SP}}, solveChallenge(sol)\rangle$. This proves to SP that U is the owner of Add_2^U, which will be set in state "Confirmed" by the function solveChallenge.

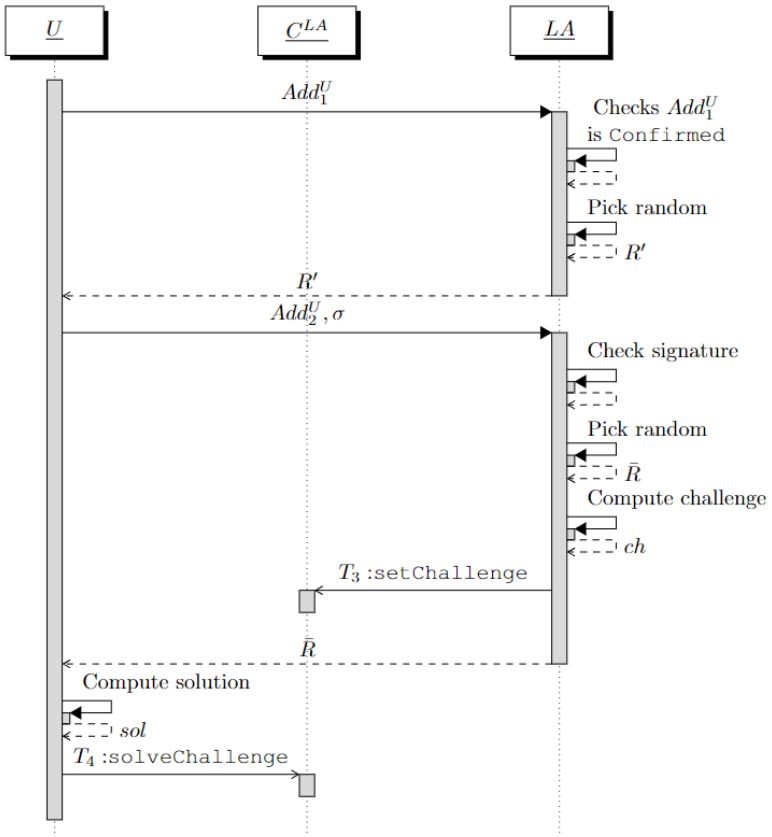


Fig. 9.2: Sequence diagram of the interaction between U and LA.

Finally, SP verifies the state "Confirmed" of the address Add_2^U on the instance C_1^{SP} and stores locally the tuple $\langle Add_2^U, user_U, \hat{R}, pass_U \rangle$. Observe that the password $pass_U$ is not included in the notarized information since it is used by SP only to authenticate U.

The above steps are summarized in the sequence diagram of Figure 9.3.

We conclude this section by observing that, as we will see in Section 9.6.2, to develop a user-friendly application, almost all the client-side operations are performed automatically by some scripts without requiring the interaction of the users. They have just to confirm some operations (generation of the transactions and signature of messages) for security reasons. Furthermore, to simplify the client-side operations, the interactions of U with SP and LA are merged into a single interaction with SP in which the user is redirected to LA.

9.5.4 Account information recovery

Suppose U loses the access credentials with SP and wants to recover the account information I_U. U has just need to prove to SP the ownership of the Ethereum address Add_2^U with which U is registered at SP. This is done, as already discussed in the previous sections, through the challenge-response mechanism involving the in-

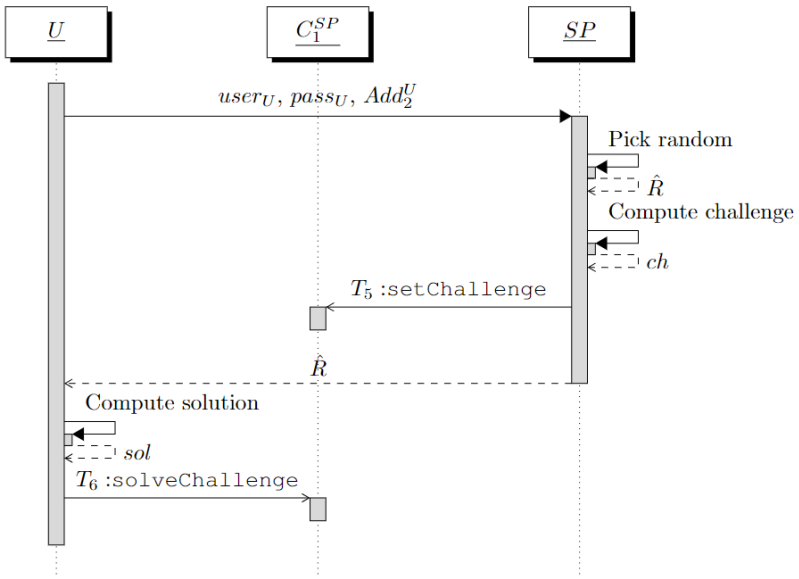


Fig. 9.3: Sequence diagram of the interaction between U and SP.

stance C_2^{SP} of the smart contract. After verifying the possession of Add_2^U, SP enables the standard module to change the password and provides U with I_U.

9.5.5 Law-enforced re-identification of *U*

Suppose a law court receives a notification about an illegal behavior of a user with username $user_U$, when leveraging the anonymous service provided by SP. After verifying whether such behavior violates the law, it emits an order to ask LA to retrieve, in a verifiable way, the real identity ID_U of the user with username $user_U$. First, LA contacts SP to obtain the Ethereum address Add_2^U associated with $user_U$. Then, LA retrieves the Ethereum address Add_1^U linked with Add_2^U (this linkage is maintained by LA itself). Finally, LA contacts IP to obtain the real identity ID_U of U associated with Add_1^U. Observe that, the mapping $user_U - Add_2^U$ is verifiable through the instance C_1^{SP} by disclosing the random $\hat{R}$. Similarly, the mapping $Add_1^U - Add_2^U$ is verifiable through the instance C^{LA} by disclosing the random $\bar{R}$. Finally, the mapping $Add_1^U - ID_U$ is verifiable through the instance C^{IP} by disclosing the random R.

9.6 Implementation and Time-Cost Analysis

In this section, we describe the prototype developed to implement the solution proposed in Section 9.5.

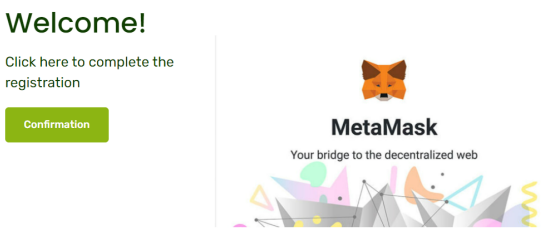


Fig. 9.4: User interface of IP.

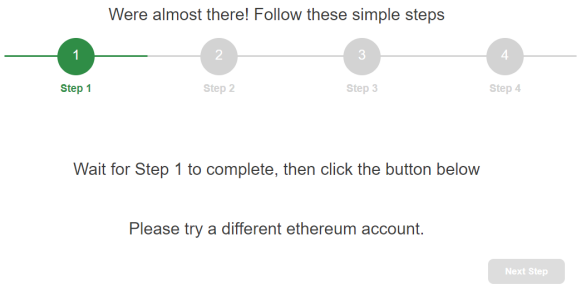


Fig. 9.5: Selection of a IP-validated Ethereum Account.

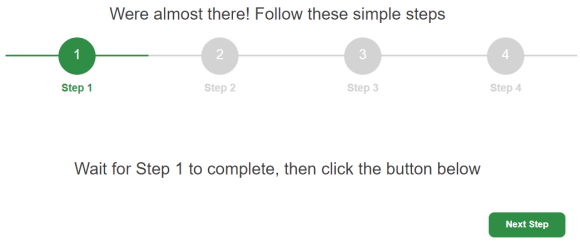


Fig. 9.6: Completion of Step 1.

9.6.1 Adopted technologies

We developed three JAVA web applications to implement IP, SP, and LA. Each application leverages the Apache Struts2 web framework [235]. Struts is an open-source framework that employs a Model, View, Controller (MVC) architecture and that enables you to create maintainable and flexible web applications. The View part of our web applications is constructed using JavaServer Pages (JSP) [163]. To improve the user experience we used Asynchronous JavaScript and XML (AJAX) [100], thus allowing the user's interaction with the application to happen asynchronously.

As explained in the previous sections, our web applications need to interact with the Ethereum blockchain both server-side and client-side.

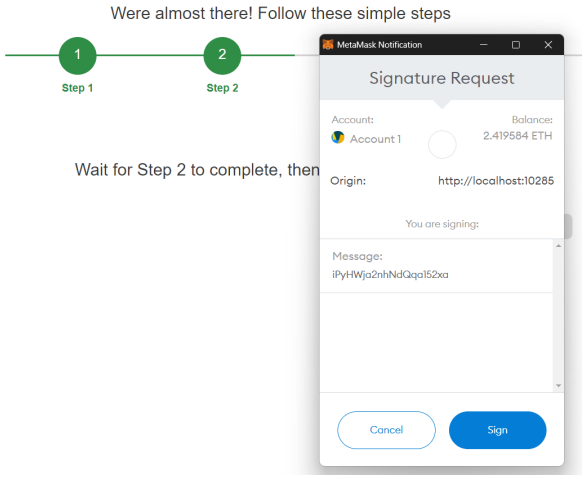


Fig. 9.7: Signature of the *LA* challenge.

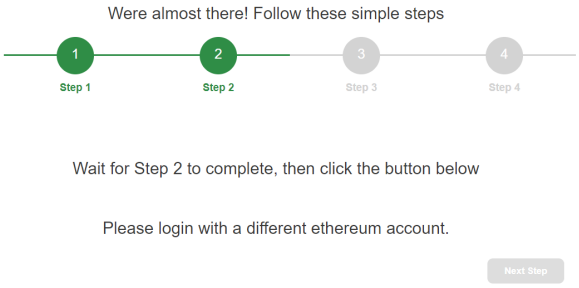


Fig. 9.8: Selection of the Ethereum address Add_2^U.

Server-side, our applications leverage Infura [137]. The Infura API suite allows you to access the Ethereum network through HTTPS and WebSockets. Infura has the main advantage of providing all the necessary tools to develop on Ethereum, without the need to run locally any blockchain node.

Client-side, our applications interface with MetaMask [186] via JavaScript. MetaMask is a very popular application allowing users to write on the blockchain. One of the main benefits of using MetaMask is that users' passwords and keys remain on the user's device since they are not shared with any other parties interacting with it.

9.6.2 Implementation detail and prototype functionalities

Through this section, we describe in detail the implementation of the prototype (including the three web applications) and describe how it works.

The implementation of the *IP* web application (whose user interface is represented in Figure 9.4) follows what has been described in Section 9.5. The selection of the address Add_1^U is executed client-side via JavaScript connecting to MetaMask. It replies with the address that is currently selected in the application. At this point, the

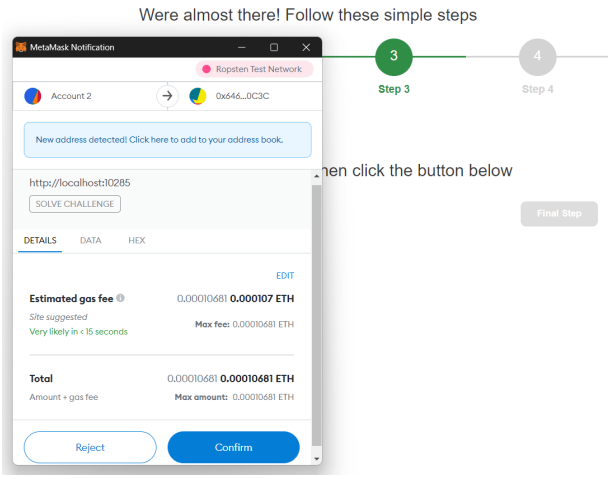


Fig. 9.9: Transaction to solve the *LA* challenge.

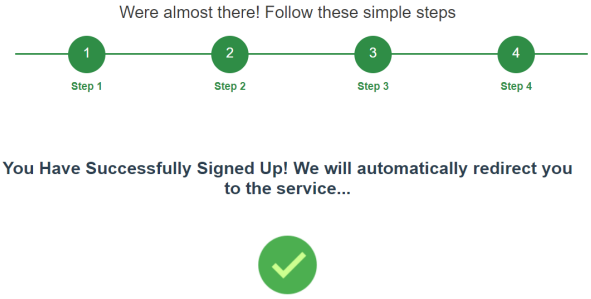


Fig. 9.10: Completion of the interaction with *LA*.

Fig. 9.11: Subscription to *SP*.

challenge-response mechanism starts. *IP* generates the challenge server-side and invokes the smart contract function `setChallenge` via Infura API. After this, the user, client-side, solves the challenge by invoking the smart contract function `solveChallenge` via MetaMask.

Now, we move to the other two applications.

Even though the high-level workflow described in Section 9.5 only require separate interactions between *U* and the two parties *SP* and *LA*, the technical implemen-

tation of the prototype does this transparently for the user, through a redirection from SP to LA of the web traffic.

In detail, when the user asks to subscribe to SP, it will redirect the user to LA for the verification of their Ethereum addresses. The procedure executed with LA is divided into four steps, as represented in Figure 9.5.

The first step is to aid the user in the selection (on Metamask) of the address Add_1^U, so that the user does not have to remember it. The LA client-side application asks MetaMask the current address selected by the user and verifies whether or not it is validated by the IP (i.e., if it is memorized in state `"Confirmed"` in the instance C^{IP}). In the case it is not validated, the LA application returns to the user a message asking them to select a different address (Figure 9.5). The procedure continues until the user selects an address validated by IP. At this point, the user can proceed with the second step (Figure 9.6).

During the second step, the user is asked to sign a random provided by the server (Figure 9.7). Then the signature is sent to the server along with the address Add_1^U, so that the server can verify that the user owns the address Add_1^U.

During the third step, the user is asked to switch to an address different from Add_1^U (Figure 9.8). After doing this, the newly selected address (Add_2^U) is sent to LA. At this point, the challenge-response mechanism starts. Therefore, LA sets a challenge on the instance C^{LA} and provides the client with all the information necessary to solve this challenge (i.e., a random $\bar{R}$). Once received, the user has just to authorize the transaction on MetaMask, in order to invoke the `solveChallenge` function of the smart contract (Figure 9.9).

Once this transaction is confirmed, the user can proceed to the fourth step, which consists of a message confirming that the whole procedure succeeded. At this point, LA will redirect the user to SP forwarding the address Add_2^U (Figure 9.10).

On SP (Figure 9.11), after another challenge-response mechanism, the user can subscribe by entering a username and a password, which will be memorized by SP along with the address Add_2^U.

This address is used also in the case the user forgets their login credentials. Again, the procedure to recover them consists of the application of the challenge-response mechanism. We recall that the user needs to select the right address to invoke the `solveChallenge` function of the smart contract. To aid the user in the selection of the right address, the client-side module of SP implements the same mechanism seen in the first step of the procedure executed with LA. We highlight that this design choice has two main benefits: (i) the user does not have to memorize Add_2^U, (ii) being a procedure performed exclusively client-side, the user's other addresses are not disclosed to the server.

9.6.3 Time-Cost analysis

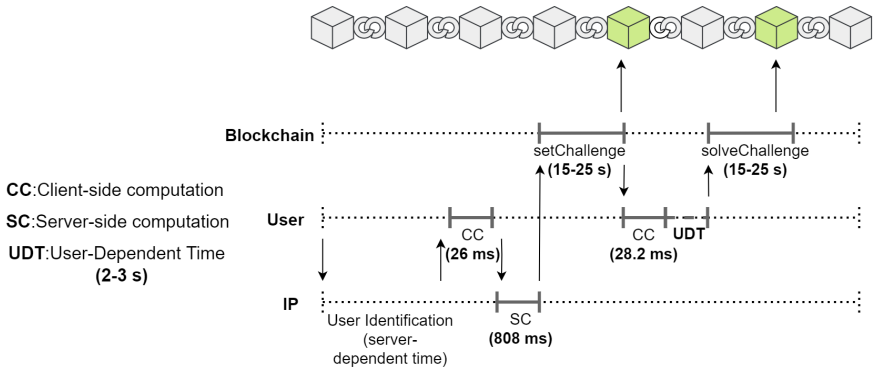


Fig. 9.12: Timeline of the interactions between user, IP, and blockchain.

.

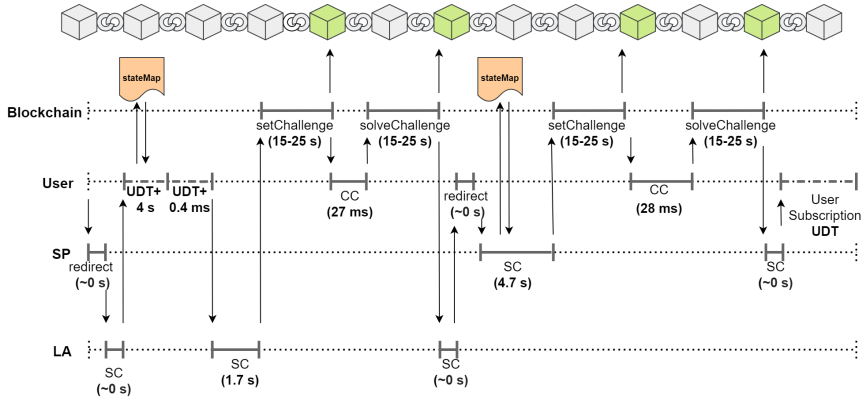


Fig. 9.13: Timeline of the interactions between user, LA, SP, and blockchain.

.

We analyzed our implementation measuring the time to perform the operations required by our solution. The measurements are performed by using a personal computer equipped with 2.8 GHz Intel i7-1165G7 CPU and 16 GB of RAM. The obtained results are reported in Figures 9.12 and 9.13.

In Figure 9.12, the interaction between user, IP, and blockchain is represented. In Figure 9.13, the interaction between user, LA, SP, and blockchain is reported.

We distinguish between the tasks executed client-side (marked by CC) and the tasks executed server-side (marked by SC). The time intervals necessary to perform these two kinds of tasks are represented with a solid grey line. For each of these intervals, the time needed to complete it is reported in the figures.

We also represent the user-dependent time intervals (marked by UDT) with a grey dashed line. Since the duration of these time intervals depends on the time the users need to click the buttons, we assume realistic average times of $2-3s$.

Finally, the measurements of the time intervals between sending a transaction to the blockchain and confirming the transaction itself are reported in the two figures. To perform these measurements, we choose the *Ropsten* test network [233], since it is able to reproduce the network conditions of the live Ethereum mainnet.

Our measurements show that the time intervals pertaining to the blockchain timeline last $15-25s$. From our experiments, these values, compared to the duration of the other measured time intervals, appear to be predominant but still acceptable.

A factor influencing the duration of the time intervals pertaining to the blockchain is the selected gas price. Indeed, a high gas price causes a transaction to be processed faster, at the cost of greater transaction fees. For our experiments, we adopt the default gas price suggested by Metamask.

To summarize the results, we obtain that the entire registration procedure with IP requires less than 1 minute while the registration procedure with SP (including the interaction with LA) requires less than 2 minutes.

By using the default gas price suggested by Metamask, we also measured the costs, in terms of Ethers, resulting from the following operations: smart contract deployment, the execution of the `setChallenge` function, and the execution of the `solveChallenge` function.

In the following, we report the resulting costs in terms of Ethers (ETH) and US dollars ($), in June 2022:

- smart contract deployment: 0.00296997 ETH / 5.34 $;
- `setChallenge` execution: 0.00029962 ETH / 0.54 $;
- `solveChallenge` execution: 0.00013468 ETH / 0.24 $.

We stress that, although the smart contract deployment has a relatively high cost, it is a one-time operation performed by SP, LA, and IP, and therefore its cost is sustainable.

From the above results, we can estimate the overall cost of a user registration, which is about 2.20$. In a real business model, this cost can be charged to the user willing to use the anonymous service. Thereby, our solution does not result in significant costs either for the user or for the service providers.

9.7 Security analysis

In this section, we discuss the security guarantees offered by our solution.

We analyze the following two *compromises* possibly affecting our protocol.

- **C1:** an adversary, not including the collaboration of IP, SP, and LA simultaneously, is able to discover the link between the username and the real identity of a user.
- **C2:** an adversary is able to forge a fake link, publicly verifiable on the blockchain, between the username and the real identity of a user.

Regarding **C1**, it represents a privacy compromise. As already discussed, our protocol allows the de-anonymization of a username only when IP, SP, and LA collaborate. In the other cases, the link real identity-username should not be disclosed to anyone.

Concerning **C2**, we refer to attacks in which the adversary attempts to associate a username $user_U$ of a user U with an identity $ID_{\bar{U}}$ of another user $\bar{U}$ to falsely accuse the latter of an illegal behavior performed by U.

Regarding the trust required of the involved actors, we make only the following *assumption*.

- **A:** IP is *honest but curious*, in the sense that performs legally the steps of the protocols but attempts to cause the two compromises **C1** and **C2**.

In other words, **A** requires that IP does not swap the real identities of two users. It is a standard assumption of all identity management systems. Indeed, the role of an identity provider is just to certify the real identity of users. However, in our threat model, we allow IP to disclose the identities it knows and show that this does not affect the privacy of the users.

No assumption is made on LA and SP. They can be considered fully malicious with respect to the considered compromises.

Now, we discuss in detail the two compromises and show how our protocol prevents them.

Compromise **C1**. This compromise occurs when the adversary identifies the real identity ID_U of a user U associated with a given username $user_U$.

Since external users just see the transactions originating from some Ethereum addresses on the blockchain, we consider SP, IP, and LA as adversaries. They can access the blockchain too, then they have at least the knowledge of all external users. By excluding the collaboration of all three (by hypothesis), we consider as adversary the collaboration of the pairs (IP, LA), (LA, SP), and (IP, SP) and show as **C1** does not occur.

We start by considering the collaboration of IP and LA. Given the user U, IP knows the real identity ID_U of U and the Ethereum address Add_1^U. Through the collaboration with LA, IP can discover the Ethereum address Add_2^U linked with Add_1^U. However, no information about the username $user_U$ of U is available to IP and LA.

Indeed, the only public information containing $user_U$ (in obfuscated form), is the transaction T_6 generated by U with the address Add_2^U. This transaction contains the solution to the challenge $sol = Kekkac256(\hat{R}||user_U)$. Without $\hat{R}$ (maintained by SP), it is not possible to reverse the hash function Kekkac256 also through dictionary attacks on all possible usernames.

Similar reasoning applies considering the pair (LA, SP). Their collaboration just allows the linkage of Add_1^U, Add_2^U, and $user_U$, but no information is available to LA and SP about ID_U. Indeed, the only public information containing ID_U (in obfuscated form), is the transaction T_2 generated by U with the address Add_1^U. This transaction contains the solution to the challenge $sol = Kekkac256(R||ID_U)$. Without R (maintained by IP), it is not possible to reverse the hash function Kekkac256.

Finally, consider the pair (IP, SP). IP maintains the mapping $Add_1^U - ID_U$, while SP maintains the mapping $Add_2^U - user_U$. However, without the mapping $Add_1^U - Add_2^U$ (maintained by LA), they are not able to link ID_U with $user_U$. The only public information containing this mapping (in obfuscated form), is the transaction T_4 generated by U. It contains the solution to the challenge $sol = Kekkac256(\bar{R}||Add_1^U||Add_2^U||\sigma)$. Again, without $\bar{R}$ and σ (maintained by LA), it is not possible to reverse the hash function Kekkac256.

Compromise **C2**. This compromise occurs whether an adversary forges a valid link (publicly verifiable on the blockchain) between the real identity $ID_{\bar{U}}$ of a user $\bar{U}$ and a username $user_U$ of a different user U.

To forge a valid link, $ID_{\bar{U}}$ has to be present (in obfuscated form) in a transaction intended for the instance C^{IP}. By Assumption **A**, IP performs legally the steps of the protocol and does not notarize the real identity of a user $\bar{U}$ not requiring the anonymous service. Therefore, we consider $\bar{U}$ as a user with Ethereum address $Add_1^{\bar{U}}$ such that the link between $ID_{\bar{U}}$ and $Add_1^{\bar{U}}$ can be verified through the instance C^{IP}.

To perform the compromise **C2**, the attacker has two possibilities. The first possibility is to forge a fake link between $Add_1^{\bar{U}}$ and Add_2^U verifiable through the instance C^{LA} such that Add_2^U is associated with $user_U$ by SP. To do this, the blockchain would have a transaction originated by Add^{LA} including the challenge $ch = Keccak256(Kekkac256(\bar{R}||Add_1^{\bar{U}}||Add_2^U||\bar{\sigma}))$, where $\bar{\sigma}$ is a signature obtained from the private key associated with the address $Add_1^{\bar{U}}$. Moreover, another transaction originated by Add_2^U, including the solution to the challenge, would be present on the blockchain. Even though the attacker coincides with the collaboration of LA and U (the only parties who can generate these two transactions), it is not able to forge the signature $\bar{\sigma}$ without the collaboration of $\bar{U}$. Therefore, this first case cannot occur.

The second possibility is that, a valid link between $Add_1^{\bar{U}}$ and $Add_2^{\bar{U}}$ (both belonging to $\bar{U}$) exists and the attacker attempts to forge a fake link between $Add_2^{\bar{U}}$ and $user_U$. However, similar to the previous case, to accomplish this, the blockchain would have a transaction originated by Add^{SP} including the challenge $ch = Keccak256(Kekkac256(\hat{R}||user_U))$ and a transaction originated by $Add_2^{\bar{U}}$, including the solution to the challenge. Even though the attacker coincides with the collaboration of SP and U, it is able to forge the first transaction but not the second. Therefore, this possibility also cannot occur.

This concludes the security analysis.

9.8 Related work

In the scientific literature there exist several papers dealing with the general issue of balancing anonymity and accountability in different fields.

[49, 12, 262] relies on anonymous credential systems and related schemes. Anonymous credential systems allow users to interact with a service provider in an anonymous yet accountable way [176, 69, 50].

[49] is the first paper proposing an anonymous credential system, in which, to prevent misuse of anonymity, the anonymity property can be revoked for particular transactions.

[12] introduces an approach relying on anonymous credentials that allows access control systems to offer fully anonymous access to resources along with strong accountability guarantees. It is worth noting that the proposed approach relies on a trusted third party to build a mechanism to escrow the identity of the user.

Most of the solutions present in the literature assume a client-server architecture in which only the clients care about their privacy. On the contrary, [267] aims to reach the right balance between privacy and accountability in P2P systems, where both clients and servers are peer users.

In [82], the authors deal with the issue of accountability in anonymous publication and storage services where malicious servers can make documents unrecoverable. To discourage this kind of behavior, the authors propose the creation of a “buddy system", that creates an association between pairs of shares from a given document. Therefore, a server holding a given share is responsible for detecting any anomaly regarding its buddy.

[85] proposes a framework to provide an anonymous mutual authentication protocol in wireless mesh networks. The proposed framework utilizes group signatures, where the private key and the credentials of the users are generated through a secure three-party protocol. User accountability is implemented via user revocation proto-

col, whose execution can be done by two semi-trusted authorities, one of which is the network operator.

[13, 79] deal with the issue of accountability in anonymous communication networks. Indeed, [79] highlights that both anonymity and accountability requirements should be satisfied to gain support for the deployment of large-scale anonymity infrastructures. To do this, [13] presents a mechanism that provides practical repudiation for the proxy nodes by tracing back selected traffic to the predecessor node through a cryptographically verifiable chain.

A proposal including some similarities to our work is presented in [202]. Indeed, the authors proposed a solution that leverages the Bitcoin blockchain as a platform to manage and determine ownership of users' access credentials. The authors design an authentication scheme able to provide anonymity and accountability without relying on any trusted third party.

Anyway, the proposed solution achieves accountability only in the sense that the service provider can blacklist the misbehaving credentials related to a user. Indeed, as highlighted by [266], accountability can still be obtained by eliminating the reliance on a third trusted party for credentials revocation, but this comes with a cost: the revoked user remains anonymous.

Another solution relying on blockchain to support anonymous authentication in VANET is provided in [180].

We conclude this section by discussing some proposals about anonymous social networks [103]. Indeed, these appear as services that might take advantage of adopting our solution.

Anonymous Social Networks completely shift the traditional social networks paradigm. Indeed, while the latter put first the user identity and its social link, anonymous social networks encourage communication between strangers and allow users to express themselves without fear of bullying or retaliation [276]. Among the most famous anonymous social networks, we found Whisper [276] and Yik Yak [32].

Several studies in the literature focus on the anonymity guarantees offered by popular anonymous social networks [57, 32, 276]. However, to the best of our knowledge, no paper concerning the accountability problem is available, although several studies suggest that the lack of accountability may encourage users to engage in illegal behavior, potentially harming other users [132].

10

Anonymous linkage of open data

User profiling activity has captured the attention of modern enterprises. To maximize the benefits, a huge quantity of data is needed. Then, the analysts often rely on open data publicly released and freely accessible. However, privacy threats arise when personal data is treated. Therefore, anonymizing techniques to hide the identity of the users are adopted. In this chapter, we consider the case in which data regarding the same user comes from multiple sources relying on different anonymizing techniques. In this case, on the one hand, the analysts are interested in linking data (regarding such a user) coming from different sources. On the other hand, it is not adequate to disclose such linkage to other parties different from the analysts. Therefore, our aim is to enable the sole analysts to link data by keeping unknown to them the real identity of the users. We propose a solution to the above problem, by instantiating it in the domain of smart cities, in which the sources of the data are represented by city subsystems such as cinemas, theaters, shopping, and so on. To offer a concrete solution, we refer to an existing open-data standard and we implement the protocol through a SAML-based SSO framework adhering to the eIDAS regulation. To the best of our knowledge, our solution is the first proposing the publication of linkable-by-design open data

10.1 Introduction

User profiling [89] is the process of collecting, analyzing, and inferring new information about users. This aspect is particularly critical for modern enterprises [241], since it allows them to capture users' interests and offer customized products and services. Clearly, in order to obtain the most accurate results possible, an enormous amount of data is required. In the literature and in real life, an effective source of such data for the activity of user profiling is represented by open data [130], i.e., data that can be accessed, used, and shared by anyone [194]. However, when dealing with personal information (even non-sensitive), serious privacy issues have to be taken into account. In the literature, several proposals are available with the aim

to anonymize the data published by a source and prevent the linkage with the real identity of the users [240, 178, 170]. In this chapter, we deal with a different problem in which data regarding the same user is provided by several sources. In these cases, the various sources may apply different anonymizing techniques and, as a result, it is not possible to link the different data, albeit in an anonymous form.

Although this can be viewed as a desirable feature from the point of view of privacy, it limits considerably the effectiveness of user profiling activity performed by the analysts, since they cannot link useful information belonging to the same users but coming from different sources. On the other hand, it appears unnecessary and potentially dangerous to disclose such a linkage to other parties, except for analysts.

Therefore, we propose a privacy-preserving solution for the linkage of open data coming from different sources relying on different anonymizing techniques. In particular, we pursue the following goals:

1. The data coming from different sources have not to be linkable for any party but the analysts. This also includes the sources themselves of data.
2. The analysts can link the data coming from different sources, but cannot know the real identity of the user associated with them.

To be concrete, we instance our solution in a smart city scenario [293] in which the sources of data are represented by city *subsystems* such as cinemas, theaters, shopping, and so on. Clearly, they represent the natural sources of useful data for user profiling [192]. As supported by the literature [14, 3], we assume the subsystems publish data in the form of open data.

Observe that, this scenario adheres perfectly to the two above-discussed objectives in terms of interoperability and privacy which represent two fundamental pillars for the development of smart cities.

Another contribution of the proposed solution is that it can be easily integrated into the Single-Sign-On authentication framework [228]. In particular, we refer to the eIDAS regulation relying on the SAML-based SSO authentication and show how it can be extended to support our proposal. Finally, the implementation of the solution, along with a case study, is also provided to witness the feasibility of our proposal.

As highlighted in Section 10.7, all proposals available in the literature aim to tackle the problem of linking data only in the aftermath. In these approaches, the linking process requires a lot of computational effort and can fail with a given percentage of error. On the contrary, our proposal aims to let the subsystems publish open data, which are anonymous and linkable by design only to authorized parties. Thereby, (i) our solution is able to protect individual privacy, (ii) linking data is a

straightforward procedure, i.e., it doesn't require a lot of computation, (iii) our solution guarantees the correctness of the linkage process.

10.2 Background

10.2.1 Open data

Open data is data that can be accessed, used, and shared by anyone. The only constraints are represented by the obligation to acknowledge the source and to share them by using the same type of license under which they had been previously released. In modern contexts, such as smart cities, users interact with several subsystems (e.g., cinemas, theaters, shopping, and so on) resulting in the production of several data. These data, properly pre-processed, can be published as open data so that they can be analyzed by other parties. Clearly, this exposes the users to several privacy concerns since their habits can be disclosed. Therefore, traditional anonymization techniques such as k-anonymity [240], l-diversity [178], and t-closeness [170] can be adopted before data are exposed. More recent and advanced techniques are reported in Section 10.7.

As a best practice, Tim Berners-Lee introduces 5 levels [26] for the definition of the format in which open data should be published. The first level regards data made available on the web in any format (not necessarily machine-readable). Level 5 refers to *Linked Open Data* [18] by requiring that machine-readable data coming from different sources can be linked to perform much more interesting analyses with respect to data coming from a single source. In this chapter, we refer to level 5, since, in smart cities, the citizens use several subsystems by producing different data that may be linked.

10.2.2 eIDAS and SAML 2.0

The eIDAS regulation aims to "provide a common normative basis for secure electronic interactions between citizens, businesses and public administrations and at increasing the security and effectiveness of electronic services and e-business and e-commerce transactions in the European Union" [270]. In this Chapter, we focus on the eIDAS authentication framework for the management and verification of the digital identities of the citizens. This framework is based on the concept of interoperability in such a way that the member states recognize the digital identities of other member states to promote cross-border cooperation.

The most famous standards to implement the eIDAS authentication framework are two: SAML 2.0 [135] and OpenID Connect [238]. To be concrete, in this chapter,

we refer to the former that is largely diffused in government and enterprise especially when the single Sign-On (SSO) approach is adopted. SSO is an authentication method allowing users to authenticate with multiple services by using a single set of credentials.

SAML 2.0 is an XML-based standard for the exchange of secure authentication and authorization messages. There are three main actors:

- The users: they are associated with a digital identity registered at an identity provider. They need to prove such an identity to a service provider to obtain a service.
- Service provider: it provides a service to the users after obtaining guarantees about their digital identity.
- Identity provider: it manages the digital identities of the users and provides the service provider with an assertion certifying a digital identity.

We conclude this section by describing the details of the SAML authentication procedure involving the above-mentioned actors. This procedure performs in several steps reported in Figure 10.1, in which the browser represents a user.

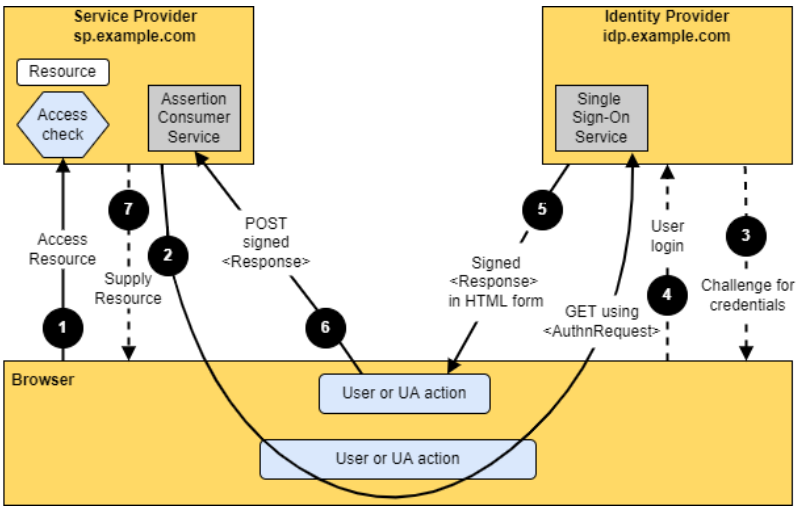


Fig. 10.1: SSO SAML authentication procedure.

1. The user asks the service provider for a resource.
2. Since the user is not authenticated, the service provider generates an `Authentication request` that is forwarded to the identity provider by the user.
3. The Identity provider asks the user for the credentials.
4. The user authenticates with the identity provider.

5. If the authentication is successful, the identity provider generates a `Response` containing an `Assertion` certifying the success of the authentication. This assertion is digitally signed and forwarded (through the user) to the service provider.
6. The service provider checks the digital signature and the validity of the assertion.
7. If the previous check is successful, the service provider supplies the required resource.

Observe that the user can leverage the same credentials to authenticate with a different service provider. Indeed they are provided to the identity provider and not directly to the service provider. This is exactly the goal of SSO.

10.3 Problem formulation and notation

In this section, we introduce the notation to model the access to the services offered by the smart city and the publication of the associated open data. We denote by $\{S_1,\ldots,S_n\}$ a set of subsystems in a smart city. Each of these subsystems offers a certain service to the population. We say that a subsystem represents a service provider in the eIDAS framework. We express the data generated by a service provider S_i as $\langle I^j, D_i^j\rangle$, where I^j represents the real identity of an individual j registered at an identity provider that is known to the service provider S_i, while D_i^j stands for the set of data, belonging to j, collected by S_i. We consider the case where each service provider is willing to publish its data as open data, thus making them publicly available so that they can be freely used for different purposes. Actually, before being published, these data must undergo several transformations (the so-called data anonymization process) in order to make them compliant with privacy regulations. More formally, we will express these transformations with two functions: α_i and δ_i, which are applied, by the subsystem i, to the real identity I^j and its data D_i^j, respectively. In detail, α_i aims to hide sensitive information about a person's identity. Although the data may appear anonymous after hiding just all the sensitive information, this procedure alone does not prevent the risk that individuals can be re-identified. Indeed, the remaining information (i.e., D_i^j) can be used to re-identify individuals by linking or matching the data with other data or by examining unique features found in the released data [257]. Therefore, as discussed in Section 10.2, more advanced privacy-preserving techniques must be applied to the data. In the following, we denote by δ_i the overall transformations applied to D_i^j not only to remove any useless details for the purpose of publication, but also to make the data difficult to de-anonymize. At this point, we assume that each subsystem S_i publishes its own data as the pair: $\langle P_i^j, \bar{D}_i^j\rangle = \langle \alpha_i(I^j), \delta_i(D_i^j)\rangle$. We denote by P_i^j a pseudonym

of the real identity j as published by S_i and by D_i^j the result of the transformation applied on D_i^j.

Observe that, different subsystems, in principle, use different α and δ. Thus, even if the data belonging to the same individual I^i are published by two different subsystems S_j and S_k (in the form of $\langle P_j^i, \bar{D}_j^i \rangle$ and $\langle P_k^i, \bar{D}_k^i \rangle$, respectively), neither the subsystems S_j and S_k, nor any other party, can link these data to I^i. Certainly, this feature is highly desirable, as it aims to protect the privacy of citizens. Indeed, if, on the contrary, the subsystem S_j is able to reverse the function α_k (i.e., to compute $I^i = \alpha_k^{-1}(P_k^i)$), it is able to link $\bar{D}_k^i$ with $\bar{D}_j^i$, and to relate these data to the real identity I_i, which is known to it.

However, as a drawback, this solution prevents, for any party, the linkage of open data belonging to a single individual, even in an anonymous form. Indeed, for example, researchers may need to combine anonymized data from many subsystems related to an individual in order to identify new patterns of relationships that would otherwise remain unknown. Clearly, these analyses cannot be performed with a single data source. Thus, this chapter aims to allow a third authorized party (e.g., a research institute) to find out which of the published data are linked to the same individual without violating that individual's privacy.

10.4 The proposed protocol

In this section, we propose a solution to the problem tackled in Section 10.3.

In detail, we will present our solution from a theoretical point of view, while in Section 10.5 we will show a practical implementation of it.

We will distinguish four different actors in our system:

- Users: denoted by U;
- Service providers: denoted by the set $\{S_1, \ldots, S_n\}$;
- Identity provider: denoted by IP;
- Analysts: denoted by the set $\{A_1, \ldots, A_t\}$;

The first three mentioned actors are the three parties that should interact in a classical SSO approach as described in Section 10.2. In the following, we will refer interchangeably to the actor S_i as service provider or subsystem. In particular, the service provider has the faculty of collecting data from the interaction with the users. Such data, properly anonymized, will be published in some format as open data so that they will be publicly available to any other external party (i.e., parties non-directly involved in the authentication process). To be concrete, we refer to an identity provider adhering to the eIDAS regulation as described in Section 10.2. However, the solution can be easily adapted to any different SSO-based approach.

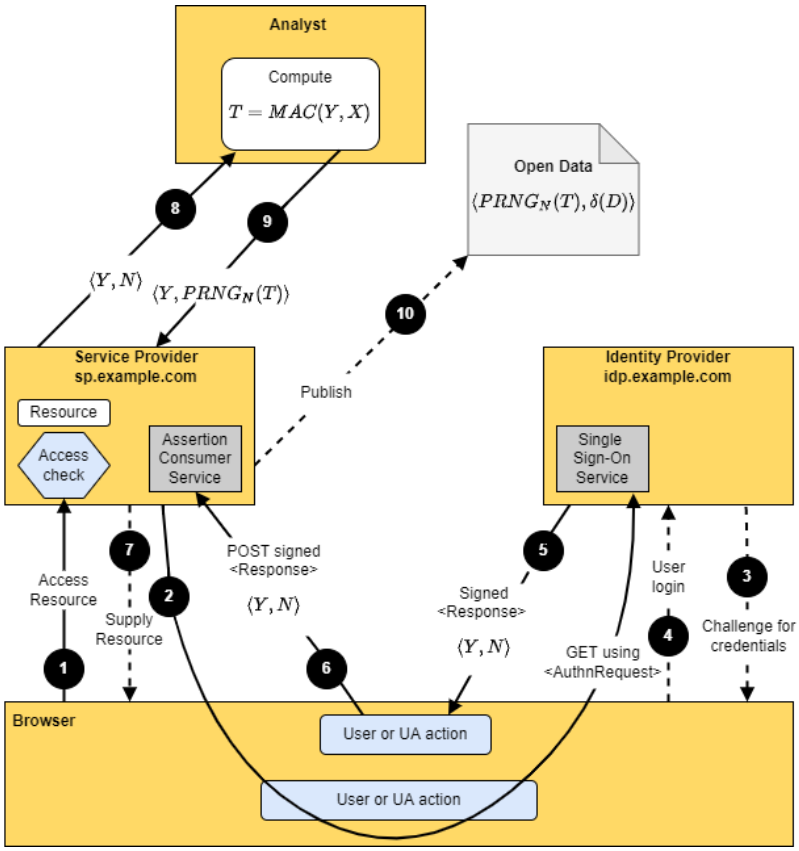


Fig. 10.2: SSO SAML proposed solution

The peculiarity of our problem demands the definition of a fourth actor, the analysts, which should be treated differently from any other external party. Indeed, once the service provider publishes the collected data, while any other external party should not be able to link the data related to the same individual, the analysts should have this capability. Suppose a user U accesses the subsystem S_i, after authenticating through the IP, and interacts with it. From these interactions a new set of data $\langle I^U, D_i^U \rangle$ will be available. As highlighted in Section 10.3, such information will be published in the form of $\langle P_i^U, \bar{D}_i^U \rangle = \langle \alpha_i(I^U), \delta_i(D_i^U) \rangle$. We aim to redefine the function α in such a way that only the analysts will be able to link the published open data referring to the same user U. No change is required for the function δ.

In particular, referring to Figure 10.1, our solution requires a simple modification of the SAML standard. Indeed, in our proposal, we need to include, in the `Assertion` message (step 5 of Figure 10.1) the following information:

- an order number N. This value represents the number of authentications performed (through the identity provider IP) so far by the user U. In other words, N will be incremented by a unity every time a user is successfully authenticated through the IP, regardless of the service provider U is willing to connect to.

- a value $Y = MAC(I^U, Secr^U)$, where MAC represents a secure message authentication code applied to I^U with key $Secr^U$ that is a secret owned by the IP associated with the user U. This secret will prevent an external party from discovering I^U through a dictionary attack performed on Y. Moreover, as Y is the output of a hash function, no collision can be found. Therefore, Y can be uniquely associated with the user U. Finally, observe that two different subsystems will receive the same Y when the same user requires two different services. This is on the basis of the procedure performed by the analysts allowing the data linkage.

We assume that for each subsystem S_k, there is a subset $\mathcal{A}_k \subseteq \{A_1, \ldots, A_t\}$ of the analysts interested in the data collected by S_k. Moreover, all the analysts in $\mathcal{A}_k$ share a secret X_k associated with the subsystem S_k.

Once the subsystem S_i receives the assertion containing $\langle Y, N\rangle$, the following steps are performed:

- S_i sends the pair $\langle Y, N\rangle$ to each analyst in the subset $\mathcal{A}_i$, as defined above.
- S_i chooses randomly an analyst A_x belonging to $\mathcal{A}_i$, to obtain a label to associate with $\bar{D}_i^U$

At this point the chosen analyst A_x proceeds as follows:

- A_x computes $T_i = MAC(Y, X_i)$, where Y is the value described above, and X_i is the secret shared with all the analysts in $\mathcal{A}_i$ and associated with the subsystem S_i as defined above;
- A_x uses T_i as seed of a PRNG, and computes the value $PRNG_N(T_i)$, denoting the N-th number obtained by the PRNG;
- A_x sends S_i the pair $\langle Y, PRNG_N(T_i)\rangle$.

We recall that T_i is related to the identity of the user U (since it depends on Y that in turn depends on I^U). Therefore $PRNG(T_i)$ represents the re-definition of the α function we propose in our solution.

Lastly, the subsystem S_i matches the message $\langle Y, PRNG_N(T_i)\rangle$ to the data D_i^U related to Y and publishes it (as open data) in the form $\langle P_i^U, \bar{D}_i^U\rangle = \langle \alpha_i(I^U), \delta_i(D_i^U)\rangle = \langle PRNG_N(T_i), \delta_i(D_i^U)\rangle$.

Suppose now, U performs Z further authentications with other subsystems after connecting with S_i. We denote by S_j the subsystem with which U performs the $N+Z$-th authentication.

S_j will publish the data D_j^U, related to U, in the form $\langle PRNG_{N+Z}(T_j), \delta(D_j^U)\rangle$. The value $PRNG_{N+Z}(T_j)$ is provided by an analyst $A_{x'}$ belonging to the subset $\mathcal{A}_j \subseteq \{A_1, \ldots, A_t\}$ of the analysts interested in the data of S_j.

At this point, both $\delta(D_i^U)$ and $\delta(D_j^U)$ result to be linkable, albeit in an anonymous form, only for authorized third parties, i.e., the analysts in $\mathcal{A}_j \cap \mathcal{A}_i$, representing the analysts interested in both the data published by S_i and S_j.

Indeed, the analysts in $\mathcal{A}_j \cap \mathcal{A}_i$ owns both the secrets X_i and X_j. Since such analysts receive the values N and $N+Z$ associated with the same Y from the subsystems S_i and S_j respectively, they are able to compute T_i and T_j and then $PRNG_N(T_i)$ and $PRNG_{N+Z}(T_j)$. Therefore, they are able to spot all the open data related to the same user. Moreover, the intrinsic properties of the PRNG assure that no other party can do the same. These points are analyzed more in detail in Section 10.6.

The steps of our solution are summarized in Fig. 10.2, in which we extend the standard SSO approach of Figure 10.1.

10.5 Case study and implementation

Through this section, we provide the implementation of the protocol described in Section 10.4 and show how it works in a simple case study.

Our implementation consists of four modules corresponding to the actors of the protocol, i.e., the user, the identity provider, the service provider, and the analyst. The user module is simply represented by a web browser. The identity provider module is based on Keycloak [65], an open-source JAVA implementation of an identity management system allowing the SSO authentication. To implement the functions described in Section 10.4, we properly modified the `saml-core.jar` library, by adding the components we need, and by intervening, in particular, on the SAML `assertion`. We will provide further details in the sequel of the section. Finally, the service provider and the analyst modules have been implemented from scratch through Servlet and JSP technology [218]. As the format for the open data, we choose JSON-LD [250], a lightweight Linked Data format recommended by W3C. It implements the level 5 format for open data described in Section 10.2.

The case study considered is the following. We have a single user named John Smith, an identity provider IP, two subsystems S_1 and S_2, and two analysts A_1 and A_2. Suppose S_1 is a streaming service (such as Netflix) and S_2 is a DVD rental activity. Furthermore, suppose both A_1 and A_2 are interested in analyzing the data coming from S_1 and S_2, i.e., $\mathcal{A}_1 = \mathcal{A}_2 = \{A_1, A_2\}$.

John authenticates with S_1 through the IP to see the movie `Ghostbuster`. The IP has to compute the values Y and N as described in Section 10.4, and send them to S_1. In our implementation, the IP maintains a folder for each registered user in which it stores the secret $Secr^J$ (where J represents the user John) and the number N counting the number of authentication performed by John. Suppose the secret of

Listing 2 Fragment of code to integrate in the library `saml-core.jar` included in Keycloak.

```
1   byte[] Y= null; Integer N;
2   try {
3       File file = new File("PATH_TO_SECRET_KEY");
4       BufferedReader br = new BufferedReader(new FileReader(file));
5       String key=br.readLine();
6       Mac mac = Mac.getInstance("HmacSHA256");
7       SecretKeySpec secretKeySpec = new
        ↪  SecretKeySpec(key.getBytes(), "HmacSHA256");
8       mac.init(secretKeySpec);
9       Y = mac.doFinal(idp.getNameIDFormatValue().getBytes());
10
11      file = new File("PATH_TO_N");
12      br = new BufferedReader(new FileReader(file));
13      String n=br.readLine();
14      N= Integer.parseInt(n)+1;
15      FileWriter myWriter = new FileWriter("PATH_TO_N");
16      myWriter.write(String.valueOf(N));
17      myWriter.close();
18
19      String pair=encodeHexString(Y)+"-"+String.valueOf(N);
20      nameIDType.setValue(pair);
21  }catch (Exception e) {
22  }
```

John is $Secr^J$=`superSecretPassword` and $N = 0$ (i.e., this is the first authentication of John). Since the computation of Y requires an identifier of John maintained by IP, we used, for simplicity, the Keycloak username of John, say `johnSmith20`. Finally, we implemented the MAC function through *HMAC* [159] based on the cryptographic hash function SHA256.

In Listing 2, we show a fragment of code to compute $\langle Y, N\rangle$ and set them in the SAML `Assertion` for the service provider. This code has to be included in the class `org.keycloak.saml.processing.api.saml.v2.response.SAML2Response` of the `saml-core.jar` library. Observe that the instruction in Line 20 sets the pair $\langle Y, N\rangle$ in field `SubjectID` of SAML `Assertion` in place of the standard username.

With the values set as above, S_1 will receive the pair
$\langle$`d94fe9ff76414b9e742819635f7dccf5fddd03c45e201ab34976f2cd9b4459a7`, 1$\rangle$.

Listing 3 Fragment of code to compute the $PRNG_N(T)$ in the analyst module.

```
SecureRandom sr = null;
try
{
    sr = SecureRandom.getInstance
    ("SHA1PRNG");
}
catch (NoSuchAlgorithmException e)
{
}
sr.setSeed(T);
long PrngN = 0;
for (int i=0;i<N;i++)
PrngN=sr.nextLong();
```

Such a pair is retrieved by a service provider (implemented through Servlet) with the instruction `String pair=request.getUserPrincipal().getName()+"-"+ UUID.randomUUID().toString().replace ("-", "")`.

At this point, S_1 forwards such a pair to A_1 and A_2. Moreover, it selects $A_1 \in \mathcal{A}_1$ to obtain the pseudonymous to associate with John's data.

A_1 computes $T_1 = MAC(Y, X_1)$, where X_1 is a secret shared among A_1 and A_2 and associated with S_1. Suppose $X_1 =$ `AnalystSecret`. Again, we chose $HMAC$ to implement the MAC function, then resulting in $T_1 =$`13715fb857d317962073856cbedbbf417c9d68eb1fe411d6713f260b7ec8af4a`. To obtain the pseudonymous to associate with the data, A_1 needs to compute $PRNG_1(T_1)$. We implemented the PRNG through a cryptographically strong random number generator (CRNG)[208]. In particular, we relied on the Java class `SecureRandom` and chose the algorithm `SHA1PRNG`. The complete code implemented in the analyst module is reported in Listing 3.

The result of this computation is $PRNG_1(T_1) =$`1807256804637968330` that is provided, along with Y, to S_1.

At this point, S_1 can publish the data in JSON-LD format as shown in Listing 4.

Therein, we refer to the Schema.org vocabulary [112], managed by a collaborative community with the aim to create, maintain, and promote schemas for structured data on the Internet. This way, our solution maintains full interoperability between data generated by different service providers. In this example, we have an array containing two objects: a Person identified by the pseudonymous obtained through our solution, and a Movie that includes, among its properties, the property "provider"

Listing 4 Open data published by S_1

```
[
{
    "@context": "https://schema.org/",
    "@type": "Person",
    "identifier": "1807256804637968330"
},
{
    "@context": "https://schema.org/",
    "@type": "Movie",
    "name": "Ghostbusters",
    "productionCompany": {
        "@type": "Organization",
        "name": "Black Rhino"
    },
    "countryOfOrigin": {
        "@type": "Country",
        "name": "USA"
    },
    "genre":"science fiction",
    "provider": {
        "@type": "Organization",
        "legalName": "S1"
    }
}
]
```

standing for the subsystem releasing such a movie. Observe that no useful information identifying John is published.

Now, we continue the description of the case study.

Since John is a fan of the genre `science fiction`, it interacts with the subsystem S_2 to rent the movie `Interstellar`. The procedure is very similar to the interaction of S_1. Then, John authenticates with IP and obtains a new pair $\langle Y, N\rangle$ where Y is the same as before, while N is increased by one. In our case study, S_2 receives the pair $\langle$`d94fe9ff76414b9e742819635f7dccf5fddd03c45e201ab34976f2cd9b4459a7`$,2\rangle$ and forwards it to both A_1 and A_2. Moreover, it selects $A_2 \in \mathcal{A}_2$ to obtain the pseudonymous to associate with John's data.

Listing 5 Open data published by S_2

```
[
{
    "@context": "https://schema.org/",
    "@type": "Person",
    "identifier": "8871020241650837923"
},
{
    "@context": "https://schema.org/",
    "@type": "Movie",
    "name": "Interstellar",
    "productionCompany": {
        "@type": "Organization",
        "name": "Syncopy Films"
    },
    "countryOfOrigin": {
        "@type": "Country",
        "name": "USA"
    },
    "genre":"science fiction",
    "provider": {
        "@type": "Organization",
        "legalName": "S2"
    }
}
]
```

A_2 computes the seed $T_2 = MAC(Y, X_2)$, where X_2 is a secret shared among A_1 and A_2 and associated with S_2. Suppose X_2 =AnalystSecret2. Then, A_2 generates a pseudonymous $PRNG_2(T_2)$ =8871020241650837923 that forwards to S_2. At this point, the data can be published by S_2 as shown in Listing 5.

We conclude this section by observing that, for any entity except for A_1 and A_2, the identifiers 1807256804637968330 and 8871020241650837923 are not linkable. This includes also the subsystems S_1 and S_2. On the other hand, since both A_1 and A_2 receive the same Y and know the same secrets X_1 and X_2, they can link the two data, but they do not know the identity of John.

10.6 Security analysis

Through this section, we provide a security analysis of the proposed solution. We start with two basic assumptions.

A1: The used cryptographic functions are secure.

A2: The anonymizing functions δ used by the subsystems prevent privacy attacks.

A3: The SSO authentication is secure and prevents impersonation attacks.

In our setting, **A1** involves the MAC and $PRNG$ functions. This assumption is easily satisfied if the identity provider and analysts use secure implementations available in the literature for such functions. Some examples, used for our implementation in Section 10.5, are $HMAC$ based on SHA256 for the MAC and the CRNG offered by the class `SecureRandom` implemented with the algorithm `SHA1PRNG`.

Regarding **A2**, again several techniques are available in the literature as discussed in Section 10.7. Finally, **A3** is a standard requirement and it is realistic since it is adopted in several real-life systems.

Consider a user U who has performed the N-th authentication with the subsystem S_i. We recall that the analysts involved in this process are in the subset $\mathcal{A}_i$. Now, consider U performs the $(N+Z)$-th authentication with the subsystem S_j. In this case, the involved analysts are in the subset $\mathcal{A}_j$.

Our system offers the following properties.

P1: The analysts in $\mathcal{A}_i \cap \mathcal{A}_j$ are able to link the data published by S_i and S_j associated with U.

P2: A user $\bar{U}$ cannot make a subsystem publish data linkable with the data associated with U.

P3: No entity but the analysts in $\mathcal{A}_i \cap \mathcal{A}_j$ is able to link the data published by S_i and S_j associated with U.

P4: No analyst can discover the real identity of U.

Property P1

Since the identity provider maintains the same secret $Secr^U$ for the user U, it computes the same $Y = MAC(I^U, Secr^U)$. S_i receives $\langle Y, N\rangle$ and S_j receives $\langle Y, N+Z\rangle$.

$\langle Y, N\rangle$ is sent to all the analysts in $\mathcal{A}_i$ and $\langle Y, N+Z\rangle$ is sent to all the analysts $\mathcal{A}_j$.

The data provided by S_i are associated with a label $PRNG_N(T_i)$, where $T_i = MAC(Y, X_i)$. Similarly, the data provided by S_j are associated with a label $PRNG_{N+Z}(T_j)$, where $T_j = MAC(Y, X_j)$. We recall that X_i is shared by all the analysts in $\mathcal{A}_i$ and X_j is shared by all the analysts in $\mathcal{A}_j$.

At this point, all the analysts in $\mathcal{A}_i \cap \mathcal{A}_j$ receive both $\langle Y, N\rangle$ and $\langle Y, N+Z\rangle$. Moreover, all the analysts in $\mathcal{A}_i \cap \mathcal{A}_j$ owns both X_i and X_j. Therefore, all the analysts in

$\mathcal{A}_i \cap \mathcal{A}_j$ are able to compute both $PRNG_N(T_i)$ and $PRNG_{N+Z}(T_j)$, and then to link the data associated with U.

Property P2

This property can be broken only if two cases occur. The first is that a user $\bar{U}$ authenticates with a subsystem S_k in place of U. However, by Assumption **A3**, impersonation attacks are not possible. The other case is that the data actually related to $\bar{U}$ are associated with a label accidentally equal to a label associated with some data related to U. More formally, it means that there exist two pairs $\langle Y, N\rangle$ and $\langle \bar{Y}, \bar{N}\rangle$ such that $PRNG_N(T) = PRNG_{\bar{N}}(\bar{T})$ where $T = MAC(Y, X)$ and $\bar{T} = MAC(\bar{Y}, \bar{X})$. We start by assuming $N \neq \bar{N}$. In this case, by Assumption **A1**, the PRNG is secure, then the probability that $PRNG_N(T) = PRNG_{\bar{N}}(\bar{T})$ is negligible regardless the values of T and $\bar{T}$. Consider now $N = \bar{N}$ (it means that U and $\bar{U}$ performed the same number of authentications). If $T \neq \bar{T}$, again, by **A1**, the probability that $PRNG_N(T) = PRNG_{\bar{N}}(\bar{T})$ is negligible. Otherwise ($T = \bar{T}$ and $N = \bar{N}$), $PRNG_N(T) = PRNG_{\bar{N}}(\bar{T})$ and **P2** is broken. However, it is easy to see that the probability that $T = \bar{T}$ is negligible. Indeed, $T = \bar{T}$ means $MAC(Y, X) = MAC(\bar{Y}, \bar{X})$. If $X \neq \bar{X}$, by Assumption **A1**, the MAC function is secure, then the probability that $T = \bar{T}$ is negligible. Similarly, if $X = \bar{X}$ (i.e., the data are published by the same subsystem), but $Y \neq \bar{Y}$, again by **A1** the probability that $T = \bar{T}$ is negligible. Otherwise ($X = \bar{X}$ and $Y = \bar{Y}$), $T = \bar{T}$ and **P2** is broken. However, it is easy to see that the probability that $Y = \bar{Y}$ is negligible. Since $Y = MAC(I^U, Secr^U)$ and $\bar{Y} = MAC(I^{\bar{U}}, Secr^{\bar{U}})$, $Y = \bar{Y}$ only if the inputs of the MAC functions are the same or if a collision occurs. The first case (same inputs) implies that $\bar{U}$ successfully impersonates U in the authentication process. This is not possible by Assumption **A3**. Finally, the second case (accidental collision), as usual, is impossible by Assumption **A2**.

Property P3

To link the data published by S_i and S_j (associated with U), the attacker should know the pairs $\langle T_i, N\rangle$ and $\langle T_j, N + Z\rangle$. The values N and $N + Z$ can be easily guessed by brute force. On the other hand, T_i is known only by the analysts $\mathcal{A}_i$ and T_j is known only by the analysts $\mathcal{A}_j$. Therefore, the only parties that simultaneously know T_i and T_j are the analysts in $\mathcal{A}_i \cap \mathcal{A}_j$. No other party, even knowing only T_i or only T_j, can link the data published by S_i and S_j.

Property P4

This property can be broken if two cases occur. In the first case, the analyst can invert the MAC function producing Y. In the second case, the analyst can directly de-anonymize the published data. However, the first case cannot occur by Assumption **A1**. Similarly, the second case cannot occur by Assumption **A2**, i.e., the subsystems use only robust anonymizing functions $delta$.

10.7 Related work

In the scientific literature, numerous papers witness the benefits derived by the use of open data [194, 156]. Indeed, open data can improve the efficiency of public services [282, 20], but also produce economic growth in the private sector [308].

In the following, we report different examples of how open data also can play a crucial role in the smart city ecosystem.

Indeed, as highlighted by [196], the growing volume and variety of data produced in the urban ecosystem might be exploited to build knowledge-based solutions for smarter and more sustainable urban development.

As an example, [188] presents the design of a service for providing a personalized path to users with special needs. The designed system leverages data from sensing and crowdsourcing (provided by users) and open data (provided by bus service providers).

[123] discusses how the exploitation of open data can turn a city (in this specific case the city of Helsinki) into a smart city aiming to offer better services to citizens. Indeed, opening up data and enabling startups to use public data at no cost have the main benefits to create new business opportunities in the form of new services and new applications. Another more recent study focusing on urban digitization and smart city development in the context of Nordic society is provided in [295]. Therein, the cities of Helsinki and Espoo are examined.

A different point of view is shown in [177]. Indeed this work investigates what barriers hinder the adoption of open data and, as a countermeasure, proposes practical recommendations to enhance open data development in the context of emerging smart cities. The case study considered in this work focuses on the city of Hong Kong.

Another realistic case study is presented in [14], in which it is explored how the city of Barcelona turned into a leading metropolis in Europe by applying smart city strategies. One of the main components of these successful strategies resides in the exploitation of open data.

Despite all the benefits coming from the exploitation of open data in different scenarios, many privacy issues may arise when it comes to disclosing data related to individual preferences and behavior [139]. As a matter of fact, removing from a given dataset all the obviously identifiable information is not enough to prevent individual re-identification. Indeed, such a method is not effective when the attacker already knows some information about the individuals in question [144].

Traditional solutions to protect individual privacy (thus preventing the above-mentioned attack) are based on the notions of k-anonimity [240], l-diversity [178]

and t-closeness [170]. Unfortunately, this method can still leak information when the attackers already know something about the information contained in the dataset [144].

In recent years, differential privacy [86] has been gaining attention and is considered among the most promising paradigms for privacy-preserving data publication and analysis [290]. Many approaches, trying to reach differential privacy, are based on adding noise to the data before disclosing them [87].

An emerging, but very promising, technique to obtain differential privacy, involves the use of Generative Adversarial Networks (GANs) [288, 303, 286]. Such a technique is used by [99], in which the authors propose a framework for releasing new open data while protecting the users' privacy.

Another challenging issue is represented by the lack of links between related data. The aim of data linkage is to merge all information related to the same entity that can be scattered among different datasets, in order to perform more powerful and efficient analysis.

As highlighted by [64], the linkage process is challenged by the lack of a common unique entity identifier. Therefore [64] proposes a data linkage system called Febrl, which is a platform for researchers to develop, implement and evaluate new data linkage algorithms and techniques. There are two potential limitations in the use of this approach: (i) running large-scale linkage can take days to complete the computation, (ii) there is still a problem of privacy since, even offloading the computation in local computers, data linkage always deals with partially identified data.

[120] provides an overview of challenges in linking administrative data for research. In particular, the authors highlight the need to preserve privacy (for the individual) without negatively impacting performance and linkage quality (accuracy of results). In the work [305], the authors investigate the privacy issues related to linkable data in smart IoT systems. Indeed, individual privacy might be severely threatened by the smart IoT ecosystem, since contents from more sources (related to the same user) are collected. Unfortunately, in [305], no solution is proposed to address this problem.

[129, 35] report the case of the Western Australian Data Linkage System (WADLS), established in 1995, able to maximize efficiency and minimize risk to privacy by centralizing data linkage activities, in addition to supporting related health system management. WADLS has provided linked data to researchers, medical practitioners, and strategic planners of Western Australia for more than 10 years.

Part IV

Conclusions

In this thesis, we proposed several protocols to achieve anonymity features in different applications. Two macro-areas are considered: anonymous communication and anonymous service delivery.

Anonymous communication refers to the protection of the identity of the users sending and/or receiving data over the Internet. In this area, we proposed two different types of protocols. The first type is represented by protocols implemented over the transport layer. In principle, they support any application layer built over them. In this class, among other approaches, we included two extensions of the famous Tor protocol. The second type of protocols for anonymous communication that we considered is implemented on an existing application layer. In this case, the protocols are not general-purpose and work only for the specific application layer considered. This is relevant when it is not possible to set external anonymous communication channels. Furthermore, we can take advantage of the specific characteristics of the application layer, to develop a high-performance protocol. Regarding this latter point, the main metrics in the field of anonymous communication (i.e., latency, cover traffic, and anonymity) are taken into consideration, also through experimental validation, in the design of our protocols.

Concerning the second macro-area, we changed our point of view about anonymity. In particular, we do not focus on anonymous communication between users, but we aim to provide users with services by protecting their identity. In this area, we also investigated new useful features achieved by relaxing the anonymity requirements. Indeed, in some scenarios, full anonymity is not desirable since it fuels cyberbullying, incitement to hate, and so on. In this case, the accountability of the actions performed by the users and their possible re-identification can discourage these phenomena. However, this re-identification should happen under the order of a court and should require the collaboration of more parties (no single party, even trusted, can re-identify the user). Another scenario in which we can relax the anonymity constraints is the anonymous linkage of data. Therein, we present a solution in which different data coming from the same user can be linked only by authorized parties to extract useful statistical information. However, no other party can make this linkage, and authorized parties cannot identify the user to whom the data relate.

A point we want to stress is the security offered by the proposed solutions. All the protocols presented in this thesis include a security analysis performed in terms of adversary capabilities and security properties, that represent a threat model. We investigated different threat models, including one very severe concerning a global adversary able to observe the entire flow of messages exchanged in the network. We show as our solutions achieve protection against such an adversary by paying an acceptable price in terms of performance.

11

Ringraziamenti

Una considerazione personale (assolutamente opinabile) è che nel momento in cui i ringraziamenti includono una lunga lista di persone (anche per pura formalità) si rischia di non far emergere il reale valore dei ringraziamenti fatti a coloro che sono stati davvero fondamentali nella stesura dell' elaborato. Questa premessa è necessaria per evidenziare quanto sentito sia il mio ringraziamento al **Prof. Francesco Buccafurri** ed anche il motivo per cui è l'unica persona inclusa in questi ringraziamenti.

Ringrazio il professore non solo per il contributo dato alla stesura di questo elaborato ma a tutto il mio percorso di dottorato. Fin dal primo momento in cui ho conosciuto il professore al corso di Algoritmi e Strutture Dati ho capito di condividere la stessa visione e approccio scientifico. Ho trovato nettamente conferma di questo nei tre anni del mio percorso di dottorato. Lo ringrazio per tutti gli insegnamenti trasmessi e i consigli che mi ha fornito. Lo ringrazio anche per la fiducia e la disponibilità dimostrata in qualsiasi ora del giorno e della notte (letteramente considerando le varie call alle 23:00).

Per concludere, aggiungo che nei tre anni di questo percorso ho capito di condividere non solo la sua visione scientifica ma anche la sua visione sui valori etici e morali.

Pertanto, non credo di esagerare nell'affermare che attualmente è la persona che stimo maggiormente e alla quale mi ispiro come esempio lavorativo e di vita.

References

1. Osman Abul and Ozan Berk Bitirgen. Anonymous location sharing in urban area mobility. *Knowledge and Information Systems*, pages 1–23, 2021.
2. Jameel Ahamed, Md Zahid, Mohd Omar, and Khaleel Ahmad. Aes and mqtt based security system in the internet of things. *Journal of Discrete Mathematical Sciences and Cryptography*, 22(8):1589–1598, 2019.
3. Bengt Ahlgren, Markus Hidell, and Edith C-H Ngai. Internet of things for smart cities: Interoperability and open data. *IEEE Internet Computing*, 20(6):52–56, 2016.
4. Masoud Akhoondi, Curtis Yu, and Harsha V Madhyastha. Lastor: A low-latency as-aware tor client. In *2012 IEEE Symposium on Security and Privacy*, pages 476–490. IEEE, 2012.
5. Judith Aldridge and David Décary-Hétu. A response to dolliver's "evaluating drug trafficking on the tor network: Silk road 2, the sequel". *International Journal of Drug Policy*, 26(11):1124–1125, 2015.
6. Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. Mcmix: Anonymous messaging via secure multiparty computation. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1217–1234, 2017.
7. Malak Alfosail and Peter Norris. Tor forensics: Proposed workflow for client memory artefacts. *Computers & Security*, 106:102311, 2021.
8. Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914, 2013.
9. Joseph Jose Anthraper and Jaidip Kotak. Security, privacy and forensic concern of mqtt protocol. In *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India*, 2019.
10. Claudio A Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati. An obfuscation-based approach for protecting location privacy. *IEEE Transactions on Dependable and Secure Computing*, 8(1):13–27, 2009.
11. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

12. Michael Backes, Jan Camenisch, and Dieter Sommer. Anonymous yet accountable access control. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pages 40–46, 2005.
13. Michael Backes, Jeremy Clark, Aniket Kate, Milivoj Simeonovski, and Peter Druschel. Backref: Accountability in anonymous communication networks. In *International Conference on Applied Cryptography and Network Security*, pages 380–400. Springer, 2014.
14. Tuba Bakıcı, Esteve Almirall, and Jonathan Wareham. A smart city initiative: the case of barcelona. *Journal of the knowledge economy*, 4(2):135–148, 2013.
15. Ranbir Singh Bali, Fehmi Jaafar, and Pavol Zavarasky. Lightweight authentication for mqtt to improve the security of iot communication. In *Proceedings of the 3rd International Conference on Cryptography Security, and Privacy*, pages 6–12, 2019.
16. Bhuvan Bamba, Ling Liu, Peter Pesti, and Ting Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of the 17th international conference on World Wide Web*, pages 237–246, 2008.
17. Lamiaa Basyoni, Noora Fetais, Aiman Erbad, Amr Mohamed, and Mohsen Guizani. Traffic analysis attacks on tor: A survey. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 183–188. IEEE, 2020.
18. Florian Bauer and Martin Kaltenböck. Linked open data: The essentials. *Edition mono/-monochrom, Vienna*, 710, 2011.
19. Kevin Bauer, Micah Sherr, and Dirk Grunwald. {ExperimenTor}: A testbed for safe and realistic tor experimentation. In *4th Workshop on Cyber Security Experimentation and Test (CSET 11)*, 2011.
20. Grace M Begany and J Ramon Gil-Garcia. Understanding the actual use of open data: Levels of engagement and how they are related. *Telematics and Informatics*, 63:101673, 2021.
21. Amos Beimel and Shlomi Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 16(1), 2003.
22. Iness Ben Guirat, Devashish Gosain, and Claudia Diaz. Mixim: Mixnet design decisions and empirical evaluation. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, pages 33–37, 2021.
23. Krista Bennett and Christian Grothoff. Gap–practical anonymous networking. In *International Workshop on Privacy Enhancing Technologies*, pages 141–160. Springer, 2003.
24. Sascha Berger, Meryem Simsek, Albrecht Fehske, Paolo Zanier, Ingo Viering, and Gerhard Fettweis. Joint downlink and uplink tilt-based self-organization of coverage and capacity under sparse system knowledge. *IEEE Transactions on Vehicular Technology*, 65(4):2259–2273, 2015.
25. Jorge Bernal Bernabe, Jose Luis Canovas, Jose L Hernandez-Ramos, Rafael Torres Moreno, and Antonio Skarmeta. Privacy-preserving solutions for blockchain: Review and challenges. *IEEE Access*, 7:164908–164940, 2019.
26. Tim Berners-Lee. star deployment scheme for open data. *http://5stardata.info/ Accessed on March 2022*, 10(04):2016, 5.

27. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 313–314. Springer, 2013.
28. Claudio Bettini. Privacy protection in location-based services: a survey. In *Handbook of Mobile Data Privacy*, pages 73–96. Springer, 2018.
29. Claudio Bettini, Sergio Mascetti, X Sean Wang, Dario Freni, and Sushil Jajodia. Anonymity and historical-anonymity in location-based services. In *Privacy in location-based applications*, pages 1–30. Springer, 2009.
30. Adhitya Bhawiyuga, Mahendra Data, and Andri Warda. Architectural design of token based authentication of mqtt protocol in constrained iot device. In *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pages 1–4. IEEE, 2017.
31. Lochan Bisne and Manish Parmar. Composite secure mqtt for internet of things using abe and dynamic s-box aes. In *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–5. IEEE, 2017.
32. Erik W Black, Kelsey Mezzina, and Lindsay A Thompson. Anonymous social media–understanding the content and context of yik yak. *Computers in Human Behavior*, 57:17–22, 2016.
33. Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 535–564. Springer, 2018.
34. Thomas Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.
35. Emma L Brook, Diana L Rosman, and C D'Arcy J Holman. Public good through data linkage: measuring research outputs from the western australian data linkage system. *Australian and New Zealand journal of public health*, 32(1):19–23, 2008.
36. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. Wip: An onion-based routing protocol strengthening anonymity. In *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 231–235, 2021.
37. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. Anonymous short communications over social networks. In Joaquin Garcia-Alfaro, Shujun Li, Radha Poovendran, Hervé Debar, and Moti Yung, editors, *Security and Privacy in Communication Networks*, pages 43–63, Cham, 2021. Springer International Publishing.
38. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. Anonymous short communications over social networks. In *International Conference on Security and Privacy in Communication Systems*, pages 43–63. Springer, 2021.
39. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. A distributed location trusted service achieving k-anonymity against the global adversary. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*, pages 133–138. IEEE, 2021.

40. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. Extending routes in tor to achieve recipient anonymity against the global adversary. In *2021 International Conference on Cyberworlds (CW)*, pages 238–245, 2021.
41. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. A privacy-preserving protocol for proximity-based services in social networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.
42. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. A protocol for anonymous short communications in social networks and its application to proximity-based services. *Online Social Networks and Media*, 31:100221, 2022.
43. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, Cecilia Labrini, and Sara Lazzaro. Achieving sender anonymity in tor against the global passive adversary. *Applied Sciences*, 12(1):137, 2022.
44. Francesco Buccafurri, Vincenzo De Angelis, Gianluca Lax, Lorenzo Musarella, and Antonia Russo. An attribute-based privacy-preserving ethereum solution for service delivery with accountability requirements. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–6, 2019.
45. Francesco Buccafurri, Vincenzo De Angelis, and Sara Lazzaro . A blockchain-based framework to enhance anonymous services with accountability guarantees. *Future Internet*, 14(8), 2022.
46. Francesco Buccafurri, Vincenzo De Angelis, and Roberto Nardone. Securing mqtt by blockchain-based otp authentication. *Sensors*, 20(7):2002, 2020.
47. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Accountability-preserving anonymous delivery of cloud services. In *International Conference on Trust and Privacy in Digital Business*, pages 124–135. Springer, 2015.
48. Marco Calabretta, Riccardo Pecori, Massimo Vecchio, and Luca Veltri. Mqtt-auth: A token-based solution to endow mqtt with authentication and authorization capabilities. *Journal of Communications Software and Systems*, 14(4):320–331, 2018.
49. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International conference on the theory and applications of cryptographic techniques*, pages 93–118. Springer, 2001.
50. Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 21–30, 2002.
51. Frank Cangialosi, Dave Levin, and Neil Spring. Ting: Measuring and exploiting latencies between all tor nodes. In *Proceedings of the 2015 Internet Measurement Conference*, pages 289–302, 2015.
52. Renee Carnley and Sikha Bagui. A public infrastructure for a trusted wireless world. *Future Internet*, 14(7):200, 2022.
53. Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts. In *International Conference on Pairing-Based Cryptography*, pages 347–366. Springer, 2010.

54. Sergio Castillo-Pérez and Joaquin Garcia-Alfaro. Onion routing circuit construction via latency graphs. *Computers & Security*, 37:197–214, 2013.
55. Marco Centenaro and Lorenzo Vangelista. A study on m2m traffic and its impact on cellular networks. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 154–159. IEEE, 2015.
56. Sanjit Chatterjee and Palash Sarkar. *Identity-based encryption*. Springer Science & Business Media, 2011.
57. Vasileios Chatzistefanou and Konstantinos Limniotis. On the (non-) anonymity of anonymous social networks. In *International Conference on e-Democracy*, pages 153–168. Springer, 2017.
58. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.
59. David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
60. Chen Chen, Daniele E Asoni, David Barrera, George Danezis, and Adrain Perrig. Hornet: High-speed onion routing at the network layer. In *Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1441–1454, 2015.
61. Chen Chen, Daniele E Asoni, Adrian Perrig, David Barrera, George Danezis, and Carmela Troncoso. Taranet: Traffic-analysis resistant anonymity at the network layer. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 137–152. IEEE, 2018.
62. Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. Preserving user location privacy in mobile data management infrastructures. In *International Workshop on Privacy Enhancing Technologies*, pages 393–412. Springer, 2006.
63. Chi-Yin Chow and Mohamed F Mokbel. Enabling private continuous queries for revealed user locations. In *International Symposium on Spatial and Temporal Databases*, pages 258–275. Springer, 2007.
64. Peter Christen, Tim Churches, and Markus Hegland. Febrl–a parallel open source data linkage system. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 638–647. Springer, 2004.
65. Marcus A Christie, Anuj Bhandar, Supun Nakandala, Suresh Marru, Eroma Abeysinghe, Sudhakar Pamidighantam, and Marlon E Pierce. Using keycloak for gateway authentication and authorization, 2017.
66. Ningning Cui, Xiaochun Yang, and Bin Wang. A novel spatial cloaking scheme using hierarchical hilbert curve for location-based services. In *International Conference on Web-Age Information Management*, pages 15–27. Springer, 2016.
67. Mauro AA da Cruz, Joel JPC Rodrigues, Pascal Lorenz, Valery V Korotaev, and Victor Hugo C de Albuquerque. In. iot—a new middleware for internet of things. *IEEE Internet of Things Journal*, 8(10):7902–7911, 2020.
68. Tore Dalenius. Finding a needle in a haystack or identifying anonymous census records. *Journal of official statistics*, 2(3):329, 1986.

69. Ivan Bjerre Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *Conference on the Theory and Application of Cryptography*, pages 328–335. Springer, 1988.
70. Maria Luisa Damiani, Elisa Bertino, Claudio Silvestri, et al. The probe framework for the personalized cloaking of private locations. *Trans. Data Priv.*, 3(2):123–148, 2010.
71. George Danezis and Claudia Diaz. A survey of anonymous communication channels. *Technical Report MSR-TR-2008-35 Microsoft Research*, 2008.
72. George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *2009 30th IEEE Symposium on Security and Privacy*, pages 269–282. IEEE, 2009.
73. George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *International Workshop on Information Hiding*, pages 293–308. Springer, 2004.
74. Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency-choose two. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 108–126. IEEE, 2018.
75. Lily Davisson, Joakim Jakovleski, Nhiem Ngo, Chau Pham, and Joel Sommers. Reassessing the constancy of end-to-end internet latency. *traffic*, 41(44):46, 2021.
76. Sumit Kumar Debnath, Pantelimon Stănică, Nibedita Kundu, and Tanmay Choudhury. Secure and efficient multiparty private set intersection cardinality. *Advances in Mathematics of Communications*, 15(2):365, 2021.
77. Plinio Santini Dester, Francisco Helder C dos S Filho, and Paulo Cardieri. Performance analysis of uplink traffic for machine type communication in wireless sensor networks. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2018.
78. Massimo Di Pierro. What is the blockchain? *Computing in Science & Engineering*, 19(5):92–95, 2017.
79. Claudia Diaz and Bart Preneel. Accountable anonymous communication. In *Security, Privacy, and Trust in Modern Data Management*, pages 239–253. Springer, 2007.
80. Tim Dierks and Christopher Allen. Rfc2246: The tls protocol version 1.0, 1999.
81. Dan Dinculeană and Xiaochun Cheng. Vulnerabilities and limitations of mqtt protocol used between iot devices. *Applied Sciences*, 9(5):848, 2019.
82. Roger Dingledine, Michael J Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *Designing Privacy Enhancing Technologies*, pages 67–95. Springer, 2001.
83. Changyu Dong and Grigorios Loukides. Approximating private set union/intersection cardinality with logarithmic complexity. *IEEE Transactions on Information Forensics and Security*, 12(11):2792–2806, 2017.
84. Jos Dumortier. Regulation (eu) no 910/2014 on electronic identification and trust services for electronic transactions in the internal market (eidas regulation), 2017.
85. Ahmet Onur Durahim and Erkay Savaş. A-make: An efficient, anonymous and accountable authentication framework for wmns. In *2010 Fifth International Conference on Internet Monitoring and Protection*, pages 54–59. IEEE, 2010.

86. Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.
87. Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 486–503. Springer, 2006.
88. Matthew Edman and Bülent Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Computing Surveys (CSUR)*, 42(1):1–35, 2009.
89. Christopher Ifeanyi Eke, Azah Anir Norman, Liyana Shuib, and Henry Friday Nweke. A survey of user profiling: State-of-the-art, challenges, and solutions. *IEEE Access*, 7:144907–144924, 2019.
90. Ehab ElSalamouny and Sébastien Gambs. Differential privacy models for location-based services. *Transactions on Data Privacy*, 9(1):15–48, 2016.
91. Esra Erdin, Chris Zachor, and Mehmet Hadi Gunes. How to find hidden users: A survey of attacks on anonymity networks. *IEEE Communications Surveys & Tutorials*, 17(4):2296–2316, 2015.
92. Csilla Farkas, Gábor Ziegler, Attila Meretei, and András Lörincz. Anonymity and accountability in self-organizing electronic communities. In *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 81–90, 2002.
93. Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Anonymity analysis of onion routing in the universally composable framework. In *Proc. of the 2012 Workshop on Provable Privacy*, 2012.
94. Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Probabilistic analysis of onion routing in a black-box model. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):1–28, 2012.
95. Jingyu Feng, Yin Wang, Jialin Wang, and Fang Ren. Blockchain-based data management and edge-assisted trusted cloaking area construction for location privacy protection in vehicular networks. *IEEE Internet of Things Journal*, 8(4):2087–2101, 2020.
96. Ian Fette and Alexey Melnikov. The websocket protocol, 2011.
97. Marten Fischer, Daniel Kümper, and Ralf Tönjes. Towards improving the privacy in the mqtt protocol. In *2019 Global IoT Summit (GIoTS)*, pages 1–6. IEEE, 2019.
98. Michael J Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206, 2002.
99. Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 151–164. Springer, 2019.
100. Jesse James Garrett et al. Ajax: A new approach to web applications, 2005.
101. William Gasarch. A survey on private information retrieval. *Bulletin of the EATCS*, 82(72-107):113, 2004.

102. Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2007.
103. Natalie Gerhart and Mehrdad Koohikamali. Social network migration and anonymity expectations: What anonymous social network apps offer. *Computers in Human Behavior*, 95:101–113, 2019.
104. Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Mobihide: a mobilea peer-to-peer system for anonymous location-based queries. In *International Symposium on Spatial and Temporal Databases*, pages 221–238. Springer, 2007.
105. Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, and Nikos Mamoulis. Fast data anonymization with low information loss. In *Proceedings of the 33rd international conference on Very large data bases*, pages 758–769, 2007.
106. Gabriel Ghinita, Keliang Zhao, Dimitris Papadias, and Panos Kalnis. A reciprocal framework for spatial k-anonymity. *Information Systems*, 35(3):299–314, 2010.
107. Yossi Gilad and Amir Herzberg. Spying in the dark: Tcp and tor traffic analysis. In *International symposium on privacy enhancing technologies symposium*, pages 100–119. Springer, 2012.
108. David M Goldschlag, Michael G Reed, and Paul F Syverson. Hiding routing information. In *International workshop on information hiding*, pages 137–150. Springer, 1996.
109. Kaj J Grahn, Thomas Forss, and Göran Pulkkis. Anonymous communication on the internet. In *Proceedings of Informing Science & IT Education Conference (InSITE)*, pages 103–120, 2014.
110. Alex Grech, Ira Sood, and Lluís Ariño. Blockchain, self-sovereign identity and digital credentials: promise versus praxis in education. *Frontiers in Blockchain*, 4:616779, 2021.
111. Andre Greubel, Steffen Pohl, and Samuel Kounev. Quantifying measurement quality and load distribution in tor. In *Annual Computer Security Applications Conference*, pages 129–140, 2020.
112. W3C Schema.org Community Group. schema.org project. `https://github.com/schemaorg/schemaorg`, 2022.
113. Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42, 2003.
114. Clement Guitton. A review of the available content on tor hidden services: The case against further development. *Computers in Human Behavior*, 29(6):2805–2815, 2013.
115. Ajay K Gupta and Udai Shanker. Omcpr: Optimal mobility aware cache data prefetching and replacement policy using spatial k-anonymity for lbs. *Wireless Personal Communications*, pages 1–25, 2020.
116. Vatsal Gupta, Sonam Khera, and Neelam Turk. Mqtt protocol employing iot based home safety system with abe encryption. *Multimedia Tools and Applications*, 80(2):2931–2949, 2021.
117. Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 7(4):12–18, 2008.

118. Feng Hao, Peter YA Ryan, and Piotr Zieliński. Anonymous voting by two-round public discussion. *IET Information Security*, 4(2):62–67, 2010.
119. Michael Harkavy, J Doug Tygar, and Hiroaki Kikuchi. Electronic auctions with private bids. In *USENIX Workshop on Electronic Commerce*, 1998.
120. Katie Harron, Chris Dibben, James Boyd, Anders Hjern, Mahmoud Azimaee, Mauricio L Barreto, and Harvey Goldstein. Challenges in administrative data linkage for research. *Big data & society*, 4(2):2053951717745678, 2017.
121. Tanzima Hashem, Lars Kulik, and Rui Zhang. Privacy preserving group nearest neighbor queries. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 489–500, 2010.
122. Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.
123. Hendrik Hielkema and Patrizia Hongisto. Developing the helsinki smart city: The role of competitions for open data applications. *Journal of the Knowledge Economy*, 4(2):190–204, 2013.
124. Andreas Hirt, Michael Jacobson, and Carey Williamson. Taxis: scalable strong anonymous communication. In *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, pages 1–10. IEEE, 2008.
125. HiveMQ. Hive mq community edition. `https://github.com/hivemq/hivemq-community-edition/wiki`, 2022.
126. HiveMQ. Hivemq mqtt client. `https://hivemq.github.io/hivemq-mqtt-client/`, 2022.
127. Nguyen Phong Hoang, Panagiotis Kintis, Manos Antonakakis, and Michalis Polychronakis. An empirical study of the i2p anonymity network and its censorship resistance. In *Proceedings of the Internet Measurement Conference 2018*, pages 379–392, 2018.
128. Susan Hohenberger, Steven Myers, Rafael Pass, et al. Anonize: A large-scale anonymous survey system. In *2014 IEEE Symposium on Security and Privacy*, pages 375–389. IEEE, 2014.
129. C D'Arcy J Holman, John A Bass, Diana L Rosman, Merran B Smith, James B Semmens, Emma J Glasson, Emma L Brook, Brooke Trutwein, Ian L Rouse, Charles R Watson, et al. A decade of data linkage in western australia: strategic design, applications and benefits of the wa data linkage system. *Australian Health Review*, 32(4):766–777, 2008.
130. Frank Hopfgartner and Joemon M Jose. Semantic user profiling techniques for personalised multimedia recommendation. *Multimedia systems*, 16(4):255–274, 2010.
131. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *GitHub: San Francisco, CA, USA*, page 1, 2016.
132. Homa Hosseinmardi, Richard Han, Qin Lv, Shivakant Mishra, and Amir Ghasemianlangroodi. Analyzing negative user behavior in a semi-anonymous social network. *CoRR abs*, 1404, 2014.
133. Axelle Hue, Gaurav Sharma, and Jean-Michel Dricot. Privacy-enhanced mqtt protocol for massive iot. *Electronics*, 11(1):70, 2021.

134. Axelle Hue, Gaurav Sharma, and Jean-Michel Dricot. Privacy-enhanced mqtt protocol for massive iot. *Electronics*, 11(1), 2022.
135. John Hughes and Eve Maler. Security assertion markup language (saml) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, 13, 2005.
136. Alfonso Iacovazzi and Yuval Elovici. Network flow watermarking: A survey. *IEEE Communications Surveys & Tutorials*, 19(1):512–530, 2016.
137. Infura Inc. The World's Most Powerful Blockchain Development Suite.
138. Shweta Iyer, GV Bansod, Praveen Naidu, and Shefali Garg. Implementation and evaluation of lightweight ciphers in mqtt environment. In *2018 International conference on electrical, electronics, communication, computer, and optimization techniques (ICEECCOT)*, pages 276–281. IEEE, 2018.
139. Tanja Jaatinen. The relationship between open data initiatives, privacy, and government transparency: a love triangle? *International Data Privacy Law*, 6(1):28, 2016.
140. Hiba Jadallah and Zaher Al Aghbari. Spatial cloaking for location-based queries in the cloud. *J. of Ambient Intelligence and Humanized Computing*, 10(9):3339–3347, 2019.
141. Priti Jagwani and Saroj Kaushik. Soft computing for scalability in context aware location based services. *Soft Computing*, 11(4), 2020.
142. Rob Jansen, Kevin S Bauer, Nicholas Hopper, and Roger Dingledine. Methodically modeling the tor network. In *CSET*, 2012.
143. Husam Al Jawaheri, Mashael Al Sabah, Yazan Boshmaf, and Aiman Erbad. Deanonymizing tor hidden service users through bitcoin transactions analysis. *Computers & Security*, 89:101684, 2020.
144. Zhanglong Ji, Zachary C Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.
145. Hongbo Jiang, Jie Li, Ping Zhao, Fanzi Zeng, Zhu Xiao, and Arun Iyengar. Location privacy-preserving mechanisms in location-based services: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(1):1–36, 2021.
146. Aaron Johnson, Rob Jansen, Nicholas Hopper, Aaron Segal, and Paul Syverson. Peerflow: Secure load balancing in tor. *Proc. Priv. Enhancing Technol.*, 2017(2):74–94, 2017.
147. Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.
148. Mouna Kacimi, Stefano Ortolani, and Bruno Crispo. Anonymous opinion exchange over untrusted social networks. In *Proceedings of the second ACM EuroSys workshop on social network systems*, pages 26–32, 2009.
149. Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE transactions on knowledge and data engineering*, 19(12):1719–1733, 2007.
150. Ishan Karunanayake, Nadeem Ahmed, Robert Malaney, Rafiqul Islam, and Sanjay Jha. Anonymity with tor: A survey on tor attacks. *arXiv preprint arXiv:2009.13018*, 2020.
151. Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, 2019.

152. Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. Protection of location privacy using dummies for location-based services. In *21st International Conf. on Data Engineering Workshops (ICDEW'05)*, pages 1248–1248. IEEE, 2005.
153. Geonwoo Kim, Seongju Kang, Jiwoo Park, and Kwangsue Chung. An mqtt-based context-aware autonomous system in onem2m architecture. *IEEE Internet of Things Journal*, 6(5):8519–8528, 2019.
154. Jong Wook Kim, Kennedy Edemacu, and Beakcheol Jang. Privacy-preserving mechanisms for location privacy in mobile crowdsensing: A survey. *Journal of Network and Computer Applications*, page 103315, 2022.
155. Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Annual International Cryptology Conference*, pages 241–257. Springer, 2005.
156. Rob Kitchin. *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage, 2014.
157. Chelsea H Komlo, Nick Mathewson, and Ian Goldberg. Walking onions: Scaling anonymity networks while protecting users. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1003–1020, 2020.
158. Panayiotis Kotzanikolaou, George Chatzisofroniou, and Mike Burmester. Broadcast anonymous routing (bar): scalable real-time anonymous communication. *International Journal of Information Security*, 16(3):313–326, 2017.
159. Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication, 1997.
160. John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.
161. Li Kuang, Yin Wang, Xiaosen Zheng, Lan Huang, and Yu Sheng. Using location semantics to realize personalized road network location privacy protection. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):1, 2020.
162. Christiane Kuhn, Martin Beck, and Thorsten Strufe. Breaking and (partially) fixing provably secure onion routing. In *2020 IEEE Symp. on Security and Privacy (SP)*, pages 168–185. IEEE, 2020.
163. Budi Kurniawan. Java for the web with servlets, jsp, and ejb, 2002.
164. Jung-Hyok Kwon, Hwi-Ho Lee, Yongseok Lim, and Eui-Jik Kim. Dominant channel occupancy for wi-fi backscatter uplink in industrial internet of things. *Applied Sciences*, 6(12):427, 2016.
165. Stevens Le Blond, David Choffnes, Wenxuan Zhou, Peter Druschel, Hitesh Ballani, and Paul Francis. Towards efficient traffic-analysis resistant anonymity networks. *ACM SIGCOMM Computer Communication Review*, 43(4):303–314, 2013.
166. Jaeheung Lee, Seokhyun Kim, Yookun Cho, Yoojin Chung, and Yongsu Park. A hierarchical clustering-based spatial cloaking algorithm for location-based services. *Journal of Internet Technology*, 13(4):645–654, 2012.
167. Brian N Levine, Michael K Reiter, Chenxi Wang, and Matthew Wright. Timing attacks in low-latency mix systems. In *International Conference on Financial Cryptography*, pages 251–265. Springer, 2004.

168. Bingdong Li, Esra Erdin, Mehmet Hadi Gunes, George Bebis, and Todd Shipley. An overview of anonymity technology usage. *Computer Communications*, 36(12):1269–1283, 2013.

169. Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2006.

170. Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2007.

171. Bin Liang, Mark A Gregory, and Shuo Li. Multi-access edge computing fundamentals, services, enablers and challenges: A complete survey. *Journal of Network and Computer Applications*, page 103308, 2021.

172. Teh-Lu Liao, Hong-Ru Lin, Pei-Yen Wan, and Jun-Juh Yan. Improved attribute-based encryption using chaos synchronization and its application to mqtt security. *Applied Sciences*, 9(20):4454, 2019.

173. Bingyu Liu, Shangyu Xie, Yuanzhou Yang, Rujia Wang, and Yuan Hong. Privacy preserving divisible double auction with a hybridized tee-blockchain system. *Cybersecurity*, 4(1):1–14, 2021.

174. Bo Liu, Wanlei Zhou, Tianqing Zhu, Longxiang Gao, and Yong Xiang. Location privacy and its applications: A systematic study. *IEEE access*, 6:17606–17624, 2018.

175. Edoardo Longo, Alessandro EC Redondi, Matteo Cesana, and Pietro Manzoni. Border: a benchmarking framework for distributed mqtt brokers. *IEEE Internet of Things Journal*, 2022.

176. Anna Lysyanskaya, Ronald L Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *International Workshop on Selected Areas in Cryptography*, pages 184–199. Springer, 1999.

177. Ruiqu Ma and Patrick TI Lam. Investigating the barriers faced by stakeholders in open data development: A study on hong kong as a "smart city". *Cities*, 92:36–46, 2019.

178. Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

179. Emmanouil Magkos, Panayiotis Kotzanikolaou, Spyros Sioutas, and Konstantinos Oikonomou. A distributed privacy-preserving scheme for location-based queries. In *2010 IEEE International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE, 2010.

180. Azees Maria, Arun Sekar Rajasekaran, Fadi Al-Turjman, Chadi Altrjman, and Leonardo Mostarda. Baiv: An efficient blockchain-based anonymous authentication and integrity preservation scheme for secure communication in vanets. *Electronics*, 11(3):488, 2022.

181. Sergio Mascetti, Claudio Bettini, Dario Freni, X Sean Wang, and Sushil Jajodia. Privacy-aware proximity based services. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 31–40. IEEE, 2009.

182. Suja P Mathews and Raju R Gondkar. Protocol recommendation for message encryption in mqtt. In *2019 International Conference on Data Science and Communication (IconDSC)*, pages 1–5. IEEE, 2019.
183. Gianluigi Me and Liberato Pesticcio. Tor black markets: economics, characterization and investigation technique. In *Cyber Criminology*, pages 119–140. Springer, 2018.
184. Diego Mendez Mena, Ioannis Papapanagiotou, and Baijian Yang. Internet of things: Survey on security. *Information Security Journal: A Global Perspective*, 27(3):162–182, 2018.
185. Fredy Mendoza-Cardenas, Rai Stiv Leon-Aguilar, and Jose Luis Quiroz-Arroyo. Cp-abe encryption over mqtt for an iot system with raspberry pi. In *2022 56th Annual Conference on Information Sciences and Systems (CISS)*, pages 236–239. IEEE, 2022.
186. MetaMask. A crypto wallet & gateway to blockchain apps.
187. Michael Michaelides, Cigdem Sengul, and Paul Patras. An experimental evaluation of mqtt authentication and authorization in iot. In *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization*, pages 69–76, 2022.
188. Silvia Mirri, Catia Prandi, Paola Salomoni, Franco Callegati, and Aldo Campi. On combining crowdsourcing, sensing and open data for an accessible smart city. In *2014 Eighth international conference on next generation mobile apps, services and technologies*, pages 294–299. IEEE, 2014.
189. Mohamed F Mokbel, Chi-Yin Chow, and Walid G Aref. The new casper: Query processing for location services without compromising privacy. In *Proc. of the 32nd international conf. on Very large data bases*, pages 763–774, 2006.
190. Antonio Montieri, Domenico Ciuonzo, Giuseppe Aceto, and Antonio Pescape. Anonymity services tor, i2p, jondonym: classifying in the dark (web). *IEEE Transactions on Dependable and Secure Computing*, 17(3):662–675, 2018.
191. Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. An empirical analysis of traceability in the monero blockchain. *arXiv preprint arXiv:1704.04299*, 2017.
192. Vaia Moustaka, Athena Vakali, and Leonidas G Anthopoulos. A systematic review for smart city data analytics. *ACM Computing Surveys (cSuR)*, 51(5):1–41, 2018.
193. Steven J Murdoch and George Danezis. Low-cost traffic analysis of tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 183–195. IEEE, 2005.
194. Peter Murray-Rust. Open data in science. *Nature Precedings*, pages 1–1, 2008.
195. Suntherasvaran Murthy, Asmidar Abu Bakar, Fiza Abdul Rahim, and Ramona Ramli. A comparative study of data anonymization techniques. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 306–309. IEEE, 2019.
196. Fátima Trindade Neves, Miguel de Castro Neto, and Manuela Aparicio. The impacts of open data initiatives on smart cities: A framework for evaluation and monitoring. *Cities*, 106:102860, 2020.

197. Navid Nikaein, Mahesh K Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Christian Bonnet. Openairinterface: A flexible platform for 5g research. *ACM SIGCOMM Computer Communication Review*, 44(5):33–38, 2014.
198. Aimaschana Niruntasukrat, Chavee Issariyapat, Panita Pongpaibool, Koonlachat Meesublak, Pramrudee Aiumsupucgul, and Anun Panya. Authorization mechanism for mqtt-based internet of things. In *2016 IEEE International Conference on Communications Workshops (ICC)*, pages 290–295. IEEE, 2016.
199. Helen Nissenbaum. The meaning of anonymity in an information age. *The Information Society*, 15(2):141–144, 1999.
200. BEN NIU, Yahong Chen, Zhibo Wang, Boyang Wang, Hui Li, et al. Eclipse: Preserving differential location privacy against long-term observation attacks. *IEEE Transactions on Mobile Computing*, 2020.
201. Ben Niu, Qinghua Li, Xiaoyan Zhu, Guohong Cao, and Hui Li. Achieving k-anonymity in privacy-aware location-based services. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 754–762. IEEE, 2014.
202. Yukun Niu, Lingbo Wei, Chi Zhang, Jianqing Liu, and Yuguang Fang. An anonymous and accountable authentication scheme for wi-fi hotspot access with the bitcoin blockchain. In *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 1–6. IEEE, 2017.
203. Mohammad Reza Nosouhi, Shui Yu, Keshav Sood, and Marthie 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)*, pages 350–357. IEEE, 2019.
204. Atul Oak and RD Daruwala. Assessment of message queue telemetry and transport (mqtt) protocol with symmetric encryption. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 5–8. IEEE, 2018.
205. Digital Ocean. Digital ocean cloud platform. `https://docs.digitalocean.com/`, 2022.
206. Jessica Oueis and Emilio Calvanese Strinati. Uplink traffic in future mobile networks: Pulling the alarm. In *International Conference on Cognitive Radio Oriented Wireless Networks*, pages 583–593. Springer, 2016.
207. Gareth Owen and Nick Savage. Empirical analysis of tor hidden services. *IET Information Security*, 10(3):113–118, 2016.
208. Fatih Özkaynak. Cryptographically secure random number generator with chaotic additional input. *Nonlinear Dynamics*, 78(3):2015–2020, 2014.
209. Gavin O'Gorman and Stephen Blott. Large scale simulation of tor. In *Annual Asian Computing Science Conference*, pages 48–54. Springer, 2007.
210. Jacob Palme and Mikael Berglund. Anonymity on the internet. *Retrieved August*, 15:2009, 2002.
211. Francesco Palmieri. A distributed flow correlation attack to anonymizing overlay networks based on wavelet multi-resolution analysis. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2271–2284, 2019.

212. Andriy Panchenko, Fabian Lanze, and Thomas Engel. Improving performance and anonymity in the tor network. In *2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC)*, pages 1–10. IEEE, 2012.
213. Andriy Panchenko, Lexi Pimenidis, and Johannes Renner. Performance analysis of anonymous communication channels provided by tor. In *2008 Third International Conference on Availability, Reliability and Security*, pages 221–228. IEEE, 2008.
214. Alireza Partovi, Wei Zheng, Taeho Jung, and Hai Lin. Ensuring privacy in location-based services: A model-based approach. *arXiv preprint arXiv:2002.10055*, 2020.
215. Chintan Patel and Nishant Doshi. A novel mqtt security framework in generic iot model. *Procedia Computer Science*, 171:1399–1408, 2020.
216. Tao Peng, Qin Liu, and Guojun Wang. Enhanced location privacy preserving scheme in location-based services. *IEEE Systems Journal*, 11(1):219–230, 2014.
217. Giovanni Perrone, Massimo Vecchio, Riccardo Pecori, Raffaele Giaffreda, et al. The day after mirai: A survey on mqtt security solutions after the largest cyber-attack carried out through an army of iot devices. In *IoTBDS*, pages 246–253, 2017.
218. Bruce W Perry. *Java servlet & JSP cookbook*. "O'Reilly Media, Inc.", 2004.
219. Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. *Technical report*, 2010.
220. Giuseppe Antonio Pierro, Roberto Tonelli, and Michele Marchesi. An organized repository of ethereum smart contracts' source codes and metrics. *Future internet*, 12(11):197, 2020.
221. Tobias Christian Piller, David Maria Merz, and Abdelmajid Khelil. Mqtt-4est: Rule-based web editor for semantic-aware topic naming in mqtt. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–8. IEEE, 2022.
222. Aniket Pingley, Nan Zhang, Xinwen Fu, Hyeong-Ah Choi, Suresh Subramaniam, and Wei Zhao. Protection of query privacy for continuous location based services. In *2011 Proceedings IEEE INFOCOM*, pages 1710–1718. IEEE, 2011.
223. Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1199–1216, 2017.
224. Thomas Prantl, Lukas Iffländer, Stefan Herrnleben, Simon Engel, Samuel Kounev, and Christian Krupitzer. Performance impact analysis of securing mqtt using tls. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pages 241–248, 2021.
225. Stefan Profanter, Ayhun Tekat, Kirill Dorofeev, Markus Rickert, and Alois Knoll. Opc ua versus ros, dds, and mqtt: performance evaluation of industry 4.0 protocols. In *2019 IEEE International Conference on Industrial Technology (ICIT)*, pages 955–962. IEEE, 2019.
226. The Tor Project. Tor metrics, 2009–2018.
227. Yanina Protskaya and Luca Veltri. Broker bridging mechanism for providing anonymity in mqtt. In *2019 10th International Conference on Networks of the Future (NoF)*, pages 110–113. IEEE, 2019.

228. V Radha and D Hitha Reddy. A survey on single sign-on techniques. *Procedia Technology*, 4:134–139, 2012.
229. David Recordon and Drummond Reed. Openid 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, 2006.
230. Michael K Reiter and Aviel D Rubin. Crowds: Anonymity for web transactions. *ACM transactions on information and system security (TISSEC)*, 1(1):66–92, 1998.
231. Florentin Rochet and Olivier Pereira. Dropping on the edge: Flexibility and traffic confirmation in onion routing protocols. *Proc. Priv. Enhancing Technol.*, 2018(2):27–46, 2018.
232. Marco Romanelli, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Optimal obfuscation mechanisms via machine learning. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 153–168. IEEE Computer Society, 2020.
233. Ropsten. Ropsten Testnet Explorer, 2006.
234. Michael Rossberg and Guenter Schaefer. A survey on automatic configuration of virtual private networks. *Computer networks*, 55(8):1684–1699, 2011.
235. Ian Roughley. Starting struts 2, 2007.
236. Antonia Russo, Gianluca Lax, Baptiste Dromard, and Menad Mezred. A system to access online services with minimal personal information disclosure. *Information Systems Frontiers*, pages 1–13, 2021.
237. Ousmane Sadio, Ibrahima Ngom, and Claude Lishou. Lightweight security scheme for mqtt/mqtt-sn protocol. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 119–123. IEEE, 2019.
238. Natsuhiko Sakimura, John Bradley, Mike Jones, Breno De Medeiros, and Chuck Mortimore. Openid connect core 1.0. *The OpenID Foundation*, page S3, 2014.
239. Juha Salo. Recent attacks on tor. *Aalto University*, 2010.
240. Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression, 1998.
241. Klaus-Dieter Schewe, Roland Kaschek, Claire Matthews, and Catherine Wallace. Modelling web-based banking systems: Story boarding and user profiling. In *International Conference on Conceptual Modeling*, pages 427–439. Springer, 2002.
242. M Zubair Shafiq, Lusheng Ji, Alex X Liu, Jeffrey Pang, and Jia Wang. Large-scale measurement and characterization of cellular machine-to-machine traffic. *IEEE/ACM transactions on Networking*, 21(6):1960–1973, 2013.
243. Tianxiang Shen, Jianyu Jiang, Yunpeng Jiang, Xusheng Chen, Ji Qi, Shixiong Zhao, Fengwei Zhang, Xiapu Luo, and Heming Cui. Daenet: making strong anonymity scale in a fully decentralized network. *IEEE Transactions on Dependable and Secure Computing*, 2021.
244. SeongHan Shin, Kazukuni Kobara, Chia-Chuan Chuang, and Weicheng Huang. A security framework for mqtt. In *2016 IEEE Conference on Communications and Network Security (CNS)*, pages 432–436. IEEE, 2016.

245. Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. A survey on routing in anonymous communication protocols. *ACM Computing Surveys (CSUR)*, 51(3):1–39, 2018.

246. Vitaly Shmatikov and Ming-Hsiu Wang. Measuring relationship anonymity in mix networks. In *Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 59–62, 2006.

247. Robin Snader and Nikita Borisov. A tune-up for tor: Improving security and performance in the tor network. In *ndss*, volume 8, page 127, 2008.

248. Robin Snader and Nikita Borisov. Improving security and performance in the tor network through tunable path selection. *IEEE Transactions on Dependable and Secure Computing*, 8(5):728–741, 2010.

249. Ridha Soua, Maria Rita Palattella, Andre Stemper, and Thomas Engel. Mqtt-mfa: a message filter aggregator to support massive iot traffic over satellite. *IEEE Internet of Things Journal*, 2021.

250. Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindström. Json-ld 1.0. *W3C recommendation*, 16:41, 2014.

251. OASIS Standard. Mqtt version 3.1. 1. *URL http://docs. oasis-open. org/mqtt/mqtt/v3*, 1:29, 2014.

252. Andy Stanford-Clark and Hong Linh Truong. Mqtt for sensor networks (mqtt-sn) protocol specification. *International business machines (IBM) Corporation version*, 1(2):1–28, 2013.

253. Wei-Tsung Su, Wei-Cheng Chen, and Chao-Chun Chen. An extensible and transparent thing-to-thing security enhancement for mqtt protocol in iot environment. In *2019 Global IoT Summit (GIoTS)*, pages 1–4. IEEE, 2019.

254. Gang Sun, Shuai Cai, Hongfang Yu, Sabita Maharjan, Victor Chang, Xiaojiang Du, and Mohsen Guizani. Location privacy preservation for mobile users in location-based services. *IEEE Access*, 7:87425–87438, 2019.

255. Gang Sun, Yuxia Xie, Dan Liao, Hongfang Yu, and Victor Chang. User-defined privacy location-sharing system in mobile online social networks. *Journal of Network and Computer Applications*, 86:34–45, 2017.

256. Yi Sun, Qian Liu, Xingyuan Chen, and Xuehui Du. An adaptive authenticated data structure with privacy-preserving for big data stream in cloud. *IEEE Transactions on Information Forensics and Security*, 15:3295–3310, 2020.

257. Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

258. Paul Syverson, Roger Dingledine, and Nick Mathewson. Tor: The second generation onion router. In *Usenix Security*, pages 303–320, 2004.

259. Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*, pages 96–114. Springer, 2001.

260. Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On multi-access edge computing: A survey of the emerging 5g network

edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

261. Qingfeng Tan, Xuebin Wang, Wei Shi, Jian Tang, and Zhihong Tian. An anonymity vulnerability in tor. *IEEE/ACM Transactions on Networking*, 2022.
262. Isamu Teranishi and Kazue Sako. K-times anonymous authentication with a constant proving cost. In *International Workshop on Public Key Cryptography*, pages 525–542. Springer, 2006.
263. Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, and Colin Keng-Yan Tan. Performance evaluation of mqtt and coap via a common middleware. In *2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*, pages 1–6. IEEE, 2014.
264. Liang Tong, Yong Li, and Wei Gao. A hierarchical edge cloud architecture for mobile computing. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.
265. Horst Treiblmaier. What is coming across the horizon and how can we handle it? bitcoin scenarios as a starting point for rigorous and relevant research. *Future Internet*, 14(6):162, 2022.
266. Patrick P Tsang, Man Ho Au, Apu Kapadia, and Sean W Smith. Perea: Towards practical ttp-free revocation in anonymous authentication. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 333–344, 2008.
267. Patrick P Tsang and Sean W Smith. Ppaa: Peer-to-peer anonymous authentication. In *International Conference on Applied Cryptography and Network Security*, pages 55–74. Springer, 2008.
268. Florian Tschorsch and Björn Scheuermann. Mind the gap: Towards a backpressure-based transport protocol for the tor network. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 597–610, 2016.
269. Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 423–440, New York, NY, USA, 2017. Association for Computing Machinery.
270. European Union. Regulation EU No 910/2014 of the European Parliament and of the Council, 23 July 2014. `http://eur-lex.europa.eu/legal-content/EN/TXT\\/HTML/?uri=CELEX\%3A32014R0910\&from=EN`.
271. European Union. European blockchain services infrastructure (ebsi). `https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/EBSI/`, Accessed online 2022.
272. Daniel Uroz and Ricardo J Rodríguez. Characterization and evaluation of iot protocols for data exfiltration. *IEEE Internet of Things Journal*, 2022.
273. Jelle Van Den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

274. Luis Von Ahn, Andrew Bortz, and Nicholas J Hopper. K-anonymous message transmission. In *Proc. of the 10th ACM conf. on Computer and Communications Security*, pages 122–130, 2003.

275. Kathleen A Wallace. Anonymity. *Ethics and Information technology*, 1(1):21–31, 1999.

276. Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y Zhao. Whispers in the dark: analysis of an anonymous social network. In *Proceedings of the 2014 conference on internet measurement conference*, pages 137–150, 2014.

277. Jian Wang. Casper-cloaking implementation in java. `https://github.com/iwangjian/Casper-Cloaking`, 2018.

278. Song Wang and X Sean Wang. In-device spatial cloaking for mobile user privacy assisted by the cloud. In *2010 Eleventh international conference on mobile data management*, pages 381–386. IEEE, 2010.

279. Wei Wang, Mehul Motani, and Vikram Srinivasan. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 323–332, 2008.

280. Wenxi Wang, Weilong Zhang, Zhang Jin, Keyan Sun, Runlin Zou, Chenrong Huang, and Yuan Tian. A novel location privacy protection scheme with generative adversarial network. In Yuan Tian, Tinghuai Ma, and Muhammad Khurram Khan, editors, *Big Data and Security*, pages 17–27, Singapore, 2020. Springer Singapore.

281. Marius Wernke, Frank Dürr, and Kurt Rothermel. Pshare: Ensuring location privacy in non-trusted systems through multi-secret sharing. *Pervasive and Mobile Computing*, 9(3):339–352, 2013.

282. Bev Wilson and Cong Cong. Beyond the supply side: Use and impact of municipal open data in the us. *Telematics and Informatics*, 58:101526, 2021.

283. Zongda Wu, Guiling Li, Shigen Shen, Xinze Lian, Enhong Chen, and Guandong Xu. Constructing dummy query sequences to protect location privacy and query privacy in location-based services. *World Wide Web*, pages 1–25, 2020.

284. Yusheng Xia, Rongmao Chen, Jinshu Su, and Hongcheng Zou. Balancing anonymity and resilience in anonymous communication networks. *Computers & Security*, page 102106, 2020.

285. Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1298–1309, 2015.

286. Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

287. Ling Xing, Xiaofan Jia, Jianping Gao, and Honghai Wu. A location privacy protection algorithm based on double k-anonymity in the social internet of vehicles. *IEEE Communications Letters*, 2021.

288. Chugui Xu, Ju Ren, Deyu Zhang, Yaoxue Zhang, Zhan Qin, and Kui Ren. Ganobfuscator: Mitigating information leakage under gan via differential privacy. *IEEE Transactions on Information Forensics and Security*, 14(9):2358–2371, 2019.

289. Hong Yang and Erik G Larsson. Can massive mimo support uplink intensive applications? In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2019.

290. Yin Yang, Zhenjie Zhang, Gerome Miklau, Marianne Winslett, and Xiaokui Xiao. Differential privacy in data publication and analysis. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 601–606, 2012.

291. Yawning. obfs4-spec.

292. Ayong Ye, Qiuling Chen, Li Xu, and Wei Wu. The flexible and privacy-preserving proximity detection in mobile social network. *Future Generation Computer Systems*, 79:271–283, 2018.

293. ChuanTao Yin, Zhang Xiong, Hui Chen, JingYuan Wang, Daven Cooper, and Bertrand David. A literature survey on smart cities. *Science China Information Sciences*, 58(10):1–18, 2015.

294. Man Lung Yiu, Christian S Jensen, Xuegang Huang, and Hua Lu. Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *2008 IEEE 24th International Conference on Data Engineering*, pages 366–375. IEEE, 2008.

295. Johanna Ylipulli and Aale Luusua. Smart cities with a nordic twist? public sector digitalization in finnish data-rich cities. *Telematics and Informatics*, 55:101457, 2020. https://www.sciencedirect.com/science/article/pii\\/S0736585320301167.

296. Adam L Young and Moti Yung. The drunk motorcyclist protocol for anonymous communication. In *2014 IEEE Conference on Communications and Network Security*, pages 157–165. IEEE, 2014.

297. Ruiyun Yu, Zhihong Bai, Leyou Yang, Pengfei Wang, Ann Oguti, and Yonghe Liu. A location cloaking algorithm based on combinatorial optimization for location-based services in 5g networks. *IEEE Access*, 4:6515–6527, 01 2016.

298. Sameh Zakhary and Abderrahim Benslimane. On location-privacy in opportunistic mobile networks, a survey. *Journal of Network and Computer Applications*, 103:157–170, 2018.

299. Bassam Zantout, Ramzi Haraty, et al. I2p data communication system. In *Proceedings of ICN*, pages 401–409. Citeseer, 2011.

300. Lucy Zhang. Building facebook messenger. https://www.facebook.com/notes/10158791547142200/. Accessed: 2022-05-04.

301. Shaobo Zhang, Kim-Kwang Raymond Choo, Qin Liu, and Guojun Wang. Enhancing privacy through uniform grid and caching in location-based services. *Future Generation Computer Systems*, 86:881–892, 2018.

302. Shaobo Zhang, Xiong Li, Zhiyuan Tan, Tao Peng, and Guojun Wang. A caching and spatial k-anonymity driven privacy enhancement scheme in continuous location-based services. *Future Generation Computer Systems*, 94:40–50, 2019.

303. Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model (technical report). *arXiv preprint arXiv:1801.01594*, 2018.

304. Yong-Bing Zhang, Qiu-Yu Zhang, Yan Yan, Yi-Long Jiang, and Mo-Yi Zhang. A k-anonymous location privacy protection method of polygon based on density distribution. *International Journal of Network Security*, 23(1):57–66, 2021.

305. Xu Zheng, Zhipeng Cai, and Yingshu Li. Data linkage in smart internet of things systems: a consideration from a privacy perspective. *IEEE Communications Magazine*, 56(9):55–61, 2018.

306. Ge Zhong and Urs Hengartner. A distributed k-anonymity protocol for location privacy. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–10. IEEE, 2009.

307. Peng Zhou, Xiapu Luo, and Rocky K.C. Chang. Inference attacks against trust-based onion routing: Trust degree to the rescue. *Computers & Security*, 39:431–446, 2013.

308. Anneke Zuiderwijk, Marijn Janssen, Kostas Poulis, and Geerten van de Kaa. Open data for competitive advantage: insights from open data use by companies. In *Proceedings of the 16th Annual International Conference on Digital Government Research*, pages 79–88, 2015.