FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Implementation and Analysis of PrivaTegrity User Discovery: Learning Contact Identifiers with Minimal Information Disclosure

Mário Yaksetig Garcia Ribeiro da Costa

Master in Electrical and Computer Engineering

FEUP Supervisor: Manuel Bernardo Barbosa

UMBC Supervisor: Alan Theodore Sherman

July 28, 2017

Abstract

Presently, individuals are increasingly aware of the privacy aspects that exist in the online world. Consequently, communication systems are working towards providing users with solutions that solve their privacy concerns. To solve this need, companies started deploying systems that provide end-to-end encryption, which allow the content of the communications to be restricted only to users involved in a particular communication channel.

However, a paradox emerges: a truly privacy-preserving communication mechanism should not be able to construct the social graphs of users, since they expose a tremendous amount of personal information associated with each individual. Nevertheless, simultaneously, a communication application needs to be able to map the users registered in the system, so that they are able to find and communicate with each other.

This thesis presents an analysis and implementation of a new Private User Discovery protocol designed to solve the leakage of information present in the social graph of users. Fundamentally, the PrivaTegrity User Discovery protocol provides a mechanism that allows any two users to establish a private channel of communication if both users have the corresponding public contact information for each other. As such, the PrivaTegrity User Discovery protocol assumes two different types of contact identifiers for every user in the system. Users need both a public contact identifier —such as an email address— and a private contact identifier, a cMix ID, which is exclusive to the PrivaTegrity system.

In addition, the PrivaTegrity User Discovery protocol does not upload any cleartext contact information associated with the PrivaTegrity system (i.e., cMix ID). Instead, the protocol combines cryptographic key agreement techniques in a manner that the server, responsible for mapping multiple users to each other, only processes hashed and encrypted information, thus being unable of constructing any type of social graph. However, this Private User Discovery protocol relies on the fact that users are comfortable with sharing a public contact identifier (e.g, email address) with the system and, simultaneously, that every user interaction with the servers goes through a mixnet.

This work features both theoretical and practical contributions in the field of Private User Discovery. In particular, this thesis highlights a detailed analysis of the proposed protocol together with the corresponding cMix mixnet. Moreover, this work includes an evaluation of multiple attack vectors and corresponding implementation solutions to strengthen the security features of the PrivaTegrity User Discovery protocol.

Acknowledgments

Para começar, gostaria de agradecer à minha mãe, Giuliana Rossina Yaksetig Garcia, por estar sempre presente e por me apoiar nos melhores e nos piores momentos. Mãe, obrigado! Sem ti esta dissertação não seria possível.

Em segundo lugar, agradecer ao meu pai, Rui Alexandre Gonçalves Ribeiro da Costa, por me ensinar que na vida não há obstáculos capazes de nos parar. Enquanto uns querem...outros fazem acontecer! Não teria sido recebido a bolsa para estudar nos Estados Unidos sem esta inspiração. Pai, obrigado!

Aos meus irmãos, por encherem a casa de alegria e energia (e também um bocadinho de caos). Este percurso não teria sido o mesmo sem vocês. Inês, Tiago e Vasco, obrigado por tudo!

Agradecer à minha avó, por fazer tudo o que se encontrava ao seu alcance pelos seus netos. Todos sabemos o quão mais complicadas as nossas vidas seriam sem este modelo exemplar na família. Ainda hoje não tenho palavras para te agradecer e descrever tudo o que fazes por nós. Não esquecer também o falecido avô que, apesar de já não estar entre nós, será sempre uma inspiração para mim.

Ao Mário Pedro, por todos os bons momentos que passámos juntos ao longo destes anos. Gostaria também de agradecer à Mónica e ao novo membro da família, Francisco, por introduzirem uma nova —e muito necessária— dinâmica na família.

Ao Jorge, Cláudia e Benedita, por mostrarem que dedicação, empenho e trabalho árduo compensam sempre no final de contas. Obrigado pela vossa presença ao meu lado durante este percurso.

Me gustaría agradecer a toda mi familia peruana, por me ayudar siempre que necesário. A la Tere y Papa Mario, tia Nena, Yaru, Carlos Augusto, Romulo, Esperanza, a mi primo Harold, a la Lelia, a William, al tio Cholo y a todos los demás. Muchas gracias!

Após concluída a família, é óbvia a escolha que se segue nesta lista. Gostaria de agradecer ao Professor Manuel Bernardo Barbosa. Ainda hoje tenho dificuldades em expressar a minha admiração. Tudo começou na primeira aula de Segurança em Sistemas e Redes. No fim da aula, sabia que queria que este Professor fosse o meu orientador. Ainda hoje me sinto afortunado por ter tido a honra de o ter como orientador. Professor, muito obrigado por tudo! Espero continuar a manter contacto consigo no futuro.

It is impossible to have an acknowledgment section without mentioning the great Professor Alan Sherman. The man that made this dream come true. Thank you for accepting me as your student, and I hope I made you proud of my work at UMBC. Moreover, I would also like to thank the crypto giant David Chaum, for letting me be a part of a research project of his. It was a great honor to be included in this work and I will never forget this moment.

In addition, a big shout out to the Cyber Defense Lab (CDL) crew that was super nice and friendly to me. Farid, Kostas & Ennis, thank you for everything, I will miss you guys!

Este parágrafo representa o agradecimento a uma das pessoas que mais contribuiu para que eu chegasse a esta etapa da minha vida. Bernardo Cardoso, Bernie, Vernier, Bernstein, é difícil

descrever a importância que tiveste neste percurso. Resta-me agradecer-te por tudo o que fizeste e por todas as longas horas que perdeste comigo para que eu pudesse passar a múltiplas cadeiras.

Gostaria também de agradecer ao Duarte Fleming e Francisco Torres por, em complemento com o Bernardo, formarem um fantástico grupo de amigos que vai ficar para a vida. É difícil descrever todas as experiências que tivemos. De tal modo que nem consigo imaginar as que ainda estão para vir... Em adição, gostaria de salientar o Miguel Vaz, o José Garcia (Jarro), o Pedro Amaral e a Maria Inês Ladeira por serem amigos de alto nível.

A todos os meus amigos que me acompanharam ao longo da minha vida e ao longo do meu percurso escolar no Colégio Cedros, dando especial destaque a Miguel Neto (e Bessódromo), André Araújo (e irmãos), e Rodrigo Mendes.

Gostaria também de agradecer aos professores que se cruzaram comigo durante este percurso académico, dando especial destaque aos seguintes: Professor João Neves, Professor Ricardo Morla, Professora Ana Aguiar, Professor André Restivo, Professor Armando Sousa, Professor Eurico Carrapatoso, Professor Luís Teixeira, Professor Manuel Ricardo, Professor Rui Araújo e ao Professor Jorge Sobrado.

Gostaria também de agradecer à Alda e Ana Paiva por me ajudarem com todos os detalhes para que eu pudesse vir para os Estados Unidos.

I could not forget to include the group of friends across the world that, even after all these years, is still an important part of my life. I would like to thank Team Comillas for everything and I hope we can hang out together soon!

Last, but far from least, I would like to thank the girl that has been here for me over the past few months. I honestly cannot picture how my US experience would have been without you. The name of such amazing girl is omitted for privacy reasons, yet I am including the hash of her full name. I guess I had to include something about cryptography even in this part of my thesis.

086D508EBC2D882DCCBAE8EB7A3FFBA4E0FE7D0779B496C62D30175F08215ECC

Gostaria de terminar fazendo minhas as palavras de Cristiano Ronaldo, que melhor descrevem o meu actual estado de espírito após concluir esta fase da minha vida: "SIIIIIIIIIM!"

Mário Yaksetig Garcia Ribeiro da Costa

"Audaces Fortuna Juvat".
A Sorte Protege os Audazes

Unknown Author

Contents

List of Figures

Chapter 1

Introduction

Digital and online communications assume an important role in the daily part of human and social interactions. Most currently available online systems, however, do not provide strong privacy protection mechanisms for their users. Moreover, as a consequence of the rapid and increasing expansion of online interactions, entities can collect a substantial amount of potentially private information and personal interests of individuals, just by analyzing their behavior on the Internet.

Networking protocols, such as TCP/IP, do not provide support for masking the identity of communication endpoints on the Internet. While it is possible to use additional protocols, such as TLS, to increase the computational difficulty for eavesdroppers to decipher the data payload of the transmitted packets, they fail to hide the IP addresses of both communicants. Moreover, if a client-server communication is not encrypted, then attackers who perform a man-in-the-middle attack, have full access to the contents of what the client and the server are transmitting and are able to perform active or passive attacks on the transmission, thus creating powerful attack vectors against both endpoints. Therefore, users find themselves in a situation where they are exposing a substantial amount of information regarding their own personal identity.

Aiming to tackle the issue of linking users to actions on the Internet, in 1981, David Chaum [1] proposed the mixnet concept, which allows users to send information over a cascade of trusted nodes, thus providing a privacy-preserving transmission mechanism. Following up on the same principle and providing new performance and privacy upgrades, PrivaTegrity appears.

PrivaTegrity is a mixnet-based system designed to provide new security and anonymity features for online services and communications. Moreover, to improve performance, this new proposed system minimizes the use of real time asymmetric cryptographic operations since the initial mixnet design by Chaum requires heavy algebraic operations in real time.

PrivaTegrity, along with providing new security and anonymity features, includes a new Private User Discovery protocol [2] that combines cryptographic key agreements with the PrivaTegrity messaging system in a novel way. This thesis introduces, analyzes and implements a new Private User Discovery method.

1.1 Problem Statement

Presently, it is easier to access services that allow users to communicate in an encrypted and more private manner. However, an important issue is born from the need of having two or more communicating ends: users need to be able to start a conversation with other users they may wish to contact. This process of having two or more users find each other in a system can assume two different names: User Discovery or Contact Discovery.

To successfully perform such User Discovery, communication systems require users to supply their entire contact list —present on their devices— to the system. Subsequently, a contact manager server, belonging to the system, intersects the uploaded contact list with the database of registered users and proceeds to respond with the group of users present both in the uploaded contact list and in the database of users registered in the system. Even though this is an efficient way of successfully connecting users to contacts on their devices, it has a potential disadvantage: the contact manager server is able to construct the map of every user in the system and how they are related. Such map is known as the social graph of users.

Therefore, an information leaking problem appears. In a strong privacy-preserving system, revealing the social graph of a user is a problem that needs solving. Consequently, a new method that allows users to connect to other users on their contact lists, without revealing their personal social graph to the service, is necessary.

Accordingly, aiming to solve that problem, PrivaTegrity introduces a new Private User Discovery method that, under certain assumptions, provides a new and reliable alternative to currently used protocols.

1.2 Thesis Structure

This thesis is divided into four parts. The first part consists of the definition of the problem and the new contributions. The second part, comprising Chapters 2 and 3, describes the relevant previous work, as well as the background necessary to properly understand the addressed topic. Part three, which includes Chapters 4, 5, and 6, focuses on introducing the PrivaTegrity system along with the Private User Discovery protocol and its security analysis. Part four, Chapters 7 and 8, specify some of the open problems associated with this research along with future work and conclusions. We briefly summarize each chapter:

- Chapter 2 presents the background necessary to provide context and a brief overview of different privacy-preserving systems.

- Chapter 3 details the previous work that has been done in User Discovery as well as some of the advantages and disadvantages of existing Private User Discovery methods. This chapter is fundamental to understand how the proposed method differs from existing Contact Discovery protocols.

- Chapter 4 describes the PrivaTegrity system and its new mixnet concept —known as cMix— as well as the security and privacy features inherent to PrivaTegrity. This chapter aims to introduce the reader to the concept of anonymous communications and a few of the existing systems, that aim to achieve anonymity, along with the differences between such systems and PrivaTegrity.

- Chapter 5 details the analysis and implementation of the PrivaTegrity User Discovery protocol in three different parts: architecture, design, and implementation. These sections cover in detail the new User Discovery protocol specifications and the different use cases.

- Chapter 6 evaluates the security of the proposed protocol. This chapter is divided into two parts: an empirical analysis, and an analysis using automated tools for cryptographic protocols. The chosen tool for such a security analysis is named TAMARIN prover and represents one of many existing options in the field of protocol analysis.

- Chapter 7, discusses some of the Engineering decisions behind the User Discovery protocol and how the proposed protocol deals with different infrastructure failures. Additionally, this chapter introduces different open problems that appeared while implementing the protocol.

- Chapter 8, the Conclusion, summarizes the entire progress as well as the achieved results. Additionally, it gathers a few of the objectives associated with this research that ought to be solved in the future.

1.3 Contributions

This thesis analyzes and implements a new Private User Discovery mechanism for the PrivaTegrity system. This protocol represents a new approach to the Private User Discovery problem and contains a substantial number of improvements, comparing to alternative available systems.

The PrivaTegrity User Discovery method does not upload cleartext contact information directly associated with the PrivaTegrity system. Instead, the protocol allows any two users to open a private communications channel, as long as the two users already have contact information for each other, from outside of the PrivaTegrity system. Therefore, two different types of identifiers are necessary for each user: a personal identifier, such as an email address, and the PrivaTegrity identifier, also referred to as the cMix ID. Therefore, this protocol can be used in other communication systems, as long as there exist two different types of identifiers: one public identifier and a private identifier associated with the communication system.

To achieve the privacy protection of the PrivaTegrity contact identifier, the protocol combines existing cryptographic key agreement techniques in a specific manner, so that the server, managing the users registered in the system, handles hash information as well as encrypted cMix identifiers that can only be decrypted by PrivaTegrity users.

Subsequently, users can begin communicating with each other over PrivaTegrity after successfully discovering their desired contacts. Additionally, in order to have a better grasp of how

PrivaTegrity works, all the protections and security features inherent to the PrivaTegrity system are presented in Chapter 4.

Chapter 2

Background

This chapter introduces existing cryptographic concepts, as well as the field of Private Information Retrieval, since they represent a fundamental background for the Private User Discovery problem. Moreover, this chapter covers multiple requirements in Private Information Retrieval systems to provide different types of anonymity and privacy features.

2.1 Encrypted Communication with Server

The first approach to the privacy field involves encryption, a fundamental concept in cryptography. Encryption is a mechanism that allows the encoding of a message so that only authorized parties can access such message. This method is commonly found in online systems where the client does not desire to leak specific information present in the communication. By using encryption, it is possible to establish a secure communication channel that allows the user to hide the content of the messages transmitted to and from the server.

Even though this method prevents third parties from eavesdropping and manipulating the contents of both the transmitted queries and respective results, it finishes up revealing the content of the queries to the server. Therefore, the server knows what the client queried for, which means that this solution alone is not enough to create a Private User Discovery method.

2.2 Private Information Retrieval Systems

Private Information Retrieval (PIR) is a technique, usually based on cryptographic principles, that allows a user to retrieve one or multiple database elements without giving the server knowledge regarding the selected records. This querying technique allows a client to look up information present in a database without conceding to the database server, the query or the response associated with the query. Therefore, the database will reply to a request without knowing what the user wanted to find. PIR is commonly considered irrelevant since the provider of information is often trusted to deal with the data of its users in a sensible way.

An example of a scenario where Private Information Retrieval is necessary is the case where Bob wants to register an online domain using a registrar. To do so, he needs to check if such domain already exists. Bob queries the registrar system to find out if the domain he wishes to register is taken. The system, after analyzing the content of the query, acquires valuable information that should belong only to Bob.

It is important to note that PIR only protects the content of the query. The identity of a user, or information associated with it, can still be revealed to the server. Using the previous example as a reference, the system would potentially have access to information associated with the IP address of the client, unless Bob was using some mechanism to hide such information (e.g., VPN service).

2.2.1 Symmetrical Private Information Retrieval

A Symmetrical Private Information Retrieval system, or Strong Private Information Retrieval (SPIR) system, features the additional requirement that the clients can only learn about the elements they are querying for, and nothing else. This requirement captures the typical privacy needs of a database owner. Moreover, it also excludes solutions like the trivial download. Such systems tend to have performance values that are not convenient for a Private User Discovery method.

2.2.2 Hardware-based Private Information Retrieval

The basic idea behind an Hardware-based PIR system lies on having a secure and trusted hardware installed at the server site.

An example of a basic protocol behind an Hardware-based PIR system using a secure co-processor (SC) consists of having the client encrypt a query with the public key of the secure co-processor and sending it to the SC. Subsequently, the secure co-processor decrypts the received query, and retrieves the entire database from the server. Afterwards, the SC keeps the requested record in its internal memory and proceeds to encrypt the element with the user's key. Finally, the server sends the encrypted element back to the client.

In such an Hardware-based PIR system, it is possible to obtain an optimal communication complexity of $O(1)$ record per query by using a secure co-processor. However, such system tends to have strong requirements as the privacy depends on the trust that is deposited on the hardware of the system, which is a rare assumption in real world systems.

Chapter 3

Previous Work

This chapter introduces the reader to the previous work in the field of User Discovery and the respective advantages and disadvantages of several existing systems.

Section one, the Contact List Intersection, describes a mechanism where users must upload the entire contact list present in their devices to a server responsible for managing the contacts of the users registered in the system. Moreover, Section two features a possible User Discovery method based on the use of hash functions. However, such method features multiple constraints that limit the functioning of the system.

Section three, describes the Entire Database download where the user requests a replica of the entire database and proceeds to query it locally, thus creating a scenario where the privacy requirements of the users are achieved.

Section four, describes a protocol used by Cryptocat, an open-source application, that provides an improved level of privacy as it allows users to decide who to look up in the system. Moreover, the Cryptocat protocol only establishes communication channels between two endpoints once both users approve that they are trying to communicate with each other. Nonetheless, this design does not prevent the social graph exposure of the users.

Finally, the last two sections describe and analyze two different User Discovery protocols. The first one is a resulting design by Open Whisper Systems and is based on bloom filters. Afterwards, this chapter introduces the Silent Circle Contact Discovery Protocol, which relies on a variant of submitting hashed contact information to the contact manager server.

3.1 Contact List Intersection

The simplest and most popular User Discovery approach is the Contact List Intersection, where users registered in a system upload to the server all the contacts on their devices. Therefore, the server can intersect the uploaded contact list with the database of registered users. After performing the intersection of the two sets, the server is able to provide the client with a list of the contacts that are conjointly on the uploaded contact list by the client, and registered in the system.

Moreover, since the server indexes the contacts present on the uploaded list, it can later notify the client if and when any of the user's contacts register in the system.

The Contact List Intersection, besides exposing the complete list of contacts users have on their phones, has a main flaw from a privacy perspective: users that uploaded their contact lists will be informed when their acquaintances register in the system. This could lead to undesired leaks of information as some users might feel uncomfortable with notifying a whole subset of people from their contact list. Moreover, users have no control regarding what they can expose to the system, which ignores the possibility that some users are potentially uncomfortable with uploading their entire contact list and prefer to look up for specific individuals instead.

3.2 Hashing the Contact Information

In addition, another possible solution is to use a hash function to hash the contact identifiers before sending them to the server. Assuming the contact manager server has the corresponding hash for every registered user, it can check for any matching hashes present in the system and return that information back to the client. However, this approach has some problems associated with it. If we assume that the contact information is a phone number, then the preimage space is small enough to calculate a map of all possible inputs. Therefore, this method is vulnerable to rainbow tables. Moreover, it is not possible to salt the information, since the identifiers need to match both on the client and server side.

3.3 Entire Database Download

Theoretically speaking, having a server send a copy of the entire database to the users and allowing them to query the content locally, solves the PIR problem. This process of downloading a copy of the entire database is known as a trivial download. Even though this method provides perfect privacy to users, since the server is unaware of the content present in their queries, it introduces great cost to users, as they need to retrieve all records from the database.

The entire download approach introduces a great communication cost to the user in cases where the databases have a considerable size. Consequently, to avoid solutions that introduce a large communication complexity, the non-triviality requirement emerges. Non-triviality implies that the communication cost must be in $O(n)$, where n is the number of bits in the database. However, from the privacy perspective of the user, this solutions is ideal as it does not disclose any information regarding possible queries. Nevertheless, the entire database download should be used in cases where the database size is small and the entire contents of the database can be disclosed, which is not the case for a User Discovery system.

3.4 Cryptocat

Cryptocat [4], developed by Nadim Kobeissi, is an open source application that uses end-to-end encryption to provide secure communications. Cryptocat's User Discovery protocol is straightforward: before a user can start communicating with another user, it must first send or accept a buddy request. By doing this type of verification on both ends, the system confirms that both users wish to communicate with each other.

Even though this is a fairly simple protocol, it verifies that users only establish communication channels with users they intend to communicate. Moreover, no contact list is uploaded nor any users are notified of any type of registration in the system. Nevertheless, the contact manager server is able to construct the social graph of all the users, which could potentially be a problem, depending on the user's desired privacy requirements and on the trust assumptions associated with the contact manager server.

3.5 Open Whisper Systems (OWS)

Open Whisper Systems [5], founded by Moxie Marlinspike in 2013, is the company behind the Signal communications application and the protocol that is currently used by multiple communication systems, such as Facebook Messenger and Whats App. Open Whisper Systems created a Private User Discovery protocol [6], based on Bloom Filters. However, the method provides a considerable communication overhead, which forced the company to stop using the method as their user base reached a tremendous magnitude, thus causing the communication cost to reach impractical levels.

3.5.1 Bloom Filters

A Bloom filter is a data structure designed by Burton Howard Bloom to inform users, in a rapidly and memory-efficient manner, whether a queried element is possibly present in a set or definitely not in the set. In addition, by manipulating the bloom filter design and combining it with encryption concepts, it is possible to create a Private User Discovery Method. However, even though bloom filters allow elements to be added to the set, they also cause an increase in the communication overhead as the set continuously grows, which limits the scalability of the system.

3.5.2 Sharden Bloom Filters

This section, describes the procedure and computational costs of dividing one single Bloom filter into several subsets, where each contains a different Bloom filter for a specific shard of users. The process of partitioning, or sharding, Bloom filters implies a trade-off between privacy and the network overhead.

By using a single Bloom filter, the server learns nothing when a client requests such filter. However, the server has to transmit a substantial amount of information to satisfy the single bloom filter request successfully.

In contrast, having a system with multiple partitions, each containing a bloom filter with a single user, provides no privacy and a low network overhead. Consequently, the server is able to learn the full content of a query whenever a client requests a filter, since each filter contains only one user.

On the other hand, the middle ground trade-off, where only two bloom filter partitions exist, allows users to retrieve a smaller bloom filter. However, the client is forced to leak part of the information he is trying to query, as the server knows which bloom filter is requested and which contacts exist in that specific filter.

Ultimately, finding an optimal trade-off is problematic. To make the total bloom filter download size acceptable, each partition can only contain a small portion of possible identifiers. Therefore, this scenario also provides a not so great privacy trade-off as it can still allow the server to guess possible social graphs, for example by comparing the location of the user requesting the filter with the areas where the other users in the requested filter are located.

3.5.3 Encrypted Bloom Filters

Coupled with the previous bloom filters examples, it is possible to improve the network efficiency by transmitting the list of registered users in a bloom filter tuned for a low false positive rate.

Moreover, to avoid leaking the complete list of registered users, it is possible to build a symmetric Private Information Retrieval system combining bloom filters and encryption, as described below:

1. Server generates an RSA key pair which is kept private.
2. Server includes the RSA signature of each registered user into the bloom filter instead of including every user.
3. Client proceeds to request the bloom filter, which contains the RSA signature of each registered user.
4. Client constructs a "blinded" query, following David Chaum's [7] blind signature scheme and proceeds to transmit the blinded query to the server.
5. Server signs the blinded query and transmits it back to the client.
6. Client unblinds the query to reveal the server's RSA signature of the contact it wishes to query and checks the local bloom filter for that specific value.

3.5.4 Limitations

Overall, the encrypted bloom filter method operates accordingly as long as there is a small-scale user base, which is no longer the case for Open Whisper Systems. Consequently, this protocol is

no longer practical and the company stopped using it. Instead, OWS informs the users that it writes the server such that transmitted contact information is not stored, thus giving them the choice of opting out of the system.

3.6 Silent Circle Contact Discovery Protocol

Phil Zimmerman [8], the creator of Pretty Good Privacy (PGP), developed the Silent Circle Contact Discovery Protocol (SCCDP), which takes a different approach to the problem of performing contact discovery without sending the server information about the contacts on the device of a user.

The protocol, consists of a variant of submitting hashed information, as described below:

1. Client compiles a list of identifiers for all the users it wants to look up (usernames, emails, phone numbers, whatever the user identifiers for the service are).
2. Client hashes each identifier with a pre-agreed-upon appropriate hashing function.
3. The client truncates each hash to its first N characters (selectable by the client, commonly 4+ bits) and submits the truncated hashes to the server.
4. The server replies with a list of slightly less truncated hashes registered in the system and that start with the characters the client sent.
5. The client then compares each less-truncated hash with the original, and, if all the characters match, it can be reasonably sure that the server knows the identifier of the user the client sent.

This protocol, represents a variant of the hashing method mentioned in the previous section. However, it experiences the same problem: the preimage space remains as a possible attack vector that can be explored by precomputed tables. Furthermore, it introduces a rate of false positives, which is not ideal for a Contact Discovery service.

In the end, this protocol obtains a smaller communication complexity than the Open Whisper Systems. However, rainbow tables represent a strong attack towards deanonymizing the contact information of users registered in the system.

Chapter 4

PrivaTegrity and cMix

This part of the thesis introduces PrivaTegrity [9]: a new communication system designed in a manner that allows smart devices to communicate anonymously without adding substantial energy consumption or heavy bandwidth usage. Moreover, conjointly with the PrivaTegrity system, this chapter also presents the backbone of PrivaTegrity, cMix [10]: a new mix network that replaces real-time public key operations with precomputations, thus providing significant real-time speedups.

4.1 PrivaTegrity

Both smart devices and online messaging systems have been experiencing a popularity intensification across the world. Nonetheless, existing online messaging services tend to leak metadata information regarding which endpoints are communicating. As such, systems designed to provide additional anonymity and privacy features, such as Tor [11], Vuvuzela [12], Dissent [13], and HORNET [14], experienced a popularity increase. All of these systems, however, have limitations against different types of existing adversaries. Onion-routing, for example, is able to achieve low latency combined with high bandwidth usage and scalability. Nevertheless, the onion routing system is susceptible to several traffic analysis attacks. On the other hand, designs based on DC-nets [15], protect the users against traffic analysis attacks, yet tend to sacrifice bandwidth. In addition, verifiable mixnets uphold strong anonymity with low bandwidth overhead, but suffer from high computation overhead instead. Therefore, systems that provide strong anonymity features represent a challenging implementation problem. Aiming to compete against these systems and to introduce new and stronger security features, PrivaTegrity appears. PrivaTegrity aims to integrate a wide variety of applications within its system, such as chat, photo and video sharing, feed following, online payments, and many others. As such, the PrivaTegrity system offers private online communications as it incorporates new security features for a wide range of anonymous social and online services.

4.2 cMix

To understand the cMix system, it is necessary to introduce multiple concepts in the field of anonymity: onion routing, I2P [16], and mix networks.

Onion routing represents a technique where messages are enclosed in multiple layers of encryption. These encrypted messages are relayed through a sequence of —typically three— nodes, or onion routers, that remove their corresponding encryption layer. As such, the message origin remains anonymous to the system since every node deals only with information regarding both the previous and next hop, thus being unable to reconstruct the full transmission path. Onion routing networks get their security primarily from choosing routes that are difficult for an adversary to observe. Such difficulty implies that the chosen routes tend to be unpredictable.

Besides onion routing, there is the Invisible Internet Project (I2P). At first glance, this system retains a lot of similarities with the onion routing mechanism. However, I2P is designed to provide access to a darknet, which means that it is a service exclusive for a network within the Internet. Moreover, the I2P routing mechanism performs packet based routing, unlike Tor that performs circuit based routing. Finally, I2P establishes two independent simplex tunnels for traffic to traverse the network to and from each host, while Tor creates a single duplex circuit. Furthermore, his section introduces the main idea behind cMix: a mix network. A mixnet typically consists of nodes receiving batches of encrypted messages, re-randomizing them, permuting them and sending such batches to the next node. However, this mixing approach usually forces the nodes to perform public-key operations, which tend to slow down the operation of the system. Unlike onion routing, the security of a mix network system depends on the mixing done by the mix nodes, and may or may not use route unpredictability to enhance security. Moreover, mix nodes are frequently intended to resist an adversary that can observe all the traffic in the system.

cMix is a new mixnet protocol, especially designed for anonymous communications, that provides significant performance and security upgrades, in contrast with existing mixnet systems. Moreover, the cMix protocol does not include any real-time public-key operations nor requires the involvement of the users in public-key operations.

The behavior of cMix can be divided into two phases: an initial precomputation phase and a real-time phase. In the precomputation phase, before any messages are known to the system, cMix nodes establish blinding factors. Subsequently, in the real-time phase, where messages flow through the mix nodes, each node multiplies these factors in. It is important to note that the real-time phase consists only of permutations and multiplications of group elements, thus allowing a significant reduction in the processing cost and consequent speedup.

The main goal of cMix is to ensure unlinkability between messages entering and leaving the system. Nevertheless, as for existing mix networks, the set of users —each identified by a 32 bit cMix ID— communicating in any batch is always known to the system. As shown in the figure below, the system consists of m senders and m' receivers, as well as n mix nodes, which process batches of messages.

Conclusively, cMix represents a new mixnet concept that, according to the authors claims,

outperforms multiple existing mix networks designs. Furthermore, besides introducing the pre-computation phase and the real-time phase, cMix includes safeguards against malicious nodes by adding the concept of node commitments to output values and, introduces multiple prevention mechanisms against message-tagging attacks.

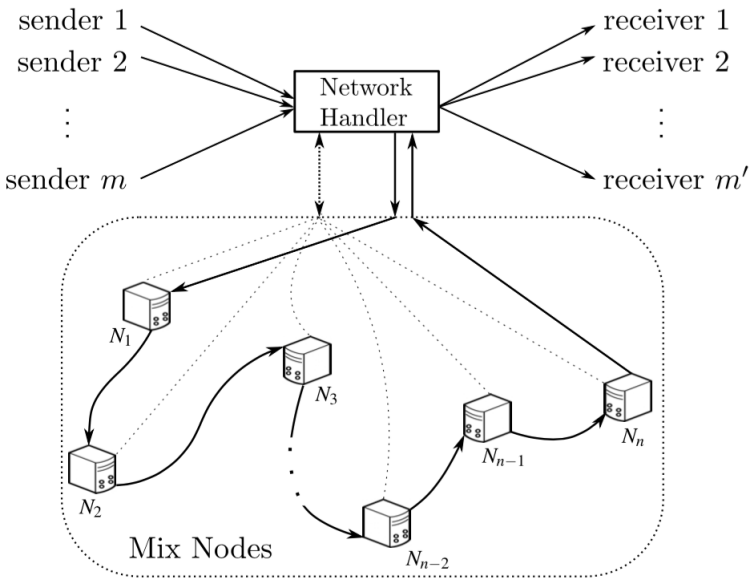

Figure 4.1: The cMix communication model.

4.2.1 Communication Model

cMix assumes m users in the system which, conjointly with a network handler, send messages over a fixed sequence of n mix nodes. Moreover, each node can process β messages at the time, where $\beta \leq m$. Furthermore, the cMix system introduces the concept of a network handler which, as well as introducing efficiency gains, arranges the inputs into batches.

During the precomputation phase, mix nodes fix a permutation of future incoming β messages. Subsequently, the real-time phase is divided into different rounds where each mix node applies a permutation to the message batches. Initially in a round, the first mix node accepts up to β messages, which the handler arranges and sorts by lexicographical order. Any message not accepted into a round is not included and is sent in a subsequent round. Moreover, cMix follows a threshold and timed mixing strategy [17], where the handler starts a new round every t seconds. A round starts only if it has at least β' messages in the buffer, for some parameter $\beta' \leq \beta$, with at least β users using the system at any given time. However, when a smaller number of users is active in the system, this strategy tends to lead to increased latency or even disruption of the service. As such, a possible solution that provides an energy consumption increase, is to inject dummy messages when needed. Therefore, it is possible to ensure enough traffic to have β messages every t seconds. Nonetheless, the specifics of these details depend on the application and are independent to the mixnet design.

4.2.2 Adversarial Model

The cMix system assumes authenticated communication channels among all mix nodes and between the network handler and any mix node. Therefore, an adversary can eavesdrop, forward and delete messages. On the other hand, the adversary is not able to modify, replay or inject new messages in the network, without detection. However, for any communication that does not take place among mix nodes or the network handler, the assumption is that the adversaries can eavesdrop, modify and inject packets.

Furthermore, cMix assumes that at least one node is trustworthy. Moreover, an adversary is able to observe, delay, or stop messages flowing trough nodes. However, such adversary is not able to read the contents of the messages. The assumption is that in case a mix node is compromised, it forwards to the adversary every received message and awaits for instructions regarding the outgoing messages. In addition, cMix accepts one message per user per batch, starting the precomputation once the batch reaches β messages.

The authors assume that the main goal of an adversary in the cMix system is to compromise the anonymity of the communication initiator, who is a user of the cMix system, or to link inputs and outputs of the system. Nonetheless, an adversary might be able to compromise users. However, the system assumes that there are at least two honest users involved in every round. Likewise, mix nodes can be compromised, yet at least one needs to remain honest for the system to be secure. cMix assumes that compromised mix nodes are malicious but cautious: they aim not to get caught violating the protocol. Besides the previous assumptions, it is important that cMix does not consider an adversary who aims to launch denial of service attacks.

4.2.3 Security and Anonymity Analysis

cMix [10] triggers the start of precomputation phase in the forward direction, by sending a message to the entry node, which creates a random permutation and stores it in a database. Moreover, this node forwards a command to the next node to start the precomputation. The system repeats this procedure until the last node finishes and notifies the other nodes and the network handler that the precomputation is complete.

cMix then initiates the precomputation for the opposite direction, which executes in a similar manner, yet includes two differences: First, each node stores the inverse of the precomputation generated earlier. Second, the first node notifies the system when this step is complete.

Consequently, cMix proceeds to inform the entry node that the real-time phase can start. Initiating from the initial node, each mix node retrieves the previously stored precomputation and proceeds to apply it to the current batch, thus triggering the next node to perform the same operation. Once the last node finishes, it informs the network handler and all nodes to start post-processing. Each node confirms to the network handler if they finished the precomputation phase correctly. Once the handler receives confirmation from all nodes, it sends notification to cMix that the real-time phase is complete.

On the other hand, cMix is a system that resists multiple standard mixnet attacks, such as message-tagging attacks, intersection attacks and statistical disclosure attacks, since these attacks take advantage of mix network topologies that permit users to choose routes freely for their messages. However, cMix uses a fixed sequence of mix nodes, which makes it not susceptible to this wide variety of attacks. Moreover, traffic analysis attacks are hard to implement against cMix, since cMix permutes messages in batches using a fixed cascade of nodes, which reduces the ability of linking senders and receivers based on time. Furthermore, cMix introduces dummy traffic to reduce the strength of these attacks.

In addition, cMix ensures sender anonymity [18] [19], which holds if all senders of a single round form an anonymity set where they are indistinguishable from all other potential senders. This anonymity holds for both forward and return messages: cMix ensures that the user who initiated communication will remain anonymous. Moreover, an adversary is only able to break sender anonymity of at least two honest users, if he compromised all the cMix nodes. Furthermore, as the authors claim, an adversary can only learn, from any compromised node, the permutation he applies to the incoming messages and not the messages themselves. In addition, the content of messages sent by users is never forwarded to the adversary and is accessed by nodes using shared memory.

Chapter 5

PrivaTegrity User Discovery

Private User Discovery poses as a difficult challenge for existing communication systems since users need to be able to discover which acquaintances are using a specific communication system. However, some users might not feel comfortable with revealing information to a contact manager server, regarding who they are looking for.

This chapter introduces a new Private User Discovery protocol that aims to solve the leakage of potentially private social graph information to the communication system and eavesdroppers. The PrivaTegrity User Discovery is designed in a manner where, initially, users register in the system by supplying a contact identifier, such as an email address, along with a public key. However, this registration information is sent over the cMix mixnet, which implies that the Contact Manager Server acquires information purely about the email address that joined the system and its corresponding public key, thus masking the origin of the submission. Subsequently, all of the social graph information is hidden from the system as, after the initial registration, the system solely deals with encrypted and hashed information, thus being unable to link information to users.

The goal of the PrivaTegrity User Discovery, proposed in this chapter, is to introduce a new mechanism in which users can establish secure communication channels solely with acquaintances they wish to contact. Moreover, the proposed protocol assumes that users are comfortable with sharing one contact identifier (e.g., an email address) with the system, and that users already possess contact information regarding the user they wish to contact. Consequently, the PrivaTegrity User Discovery protocol provides a method for PrivaTegrity users to find each other in a system without revealing to the server information regarding which users have established secure channels between them.

5.1 Architecture

The PrivaTegrity User Discovery mechanism comprises multiple entities, such as the cMix mixnet, the users of the PrivaTegrity system, and two servers external to the cMix system: A Contact Manager Server and a Hash Manager Server. To initiate the User Discovery Protocol, users connect to the cMix mixnet and proceed to interact with the respective servers.

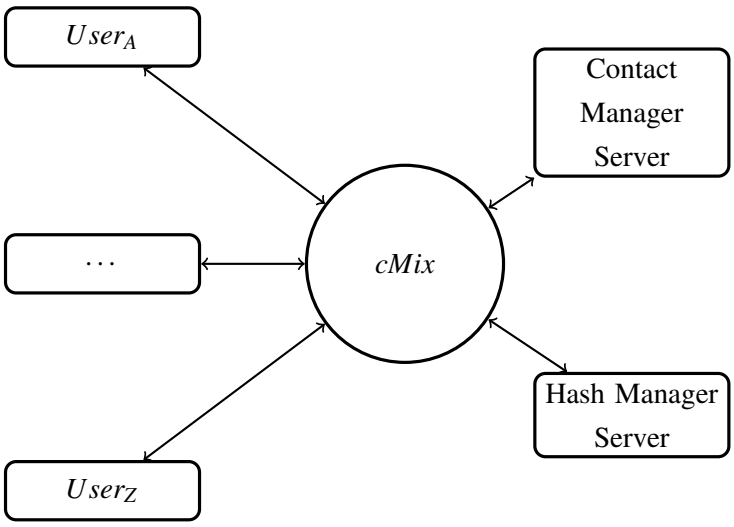

Figure 5.1: Architecture Diagram

As seen in the Fig 5.1, users are able to hide their original endpoints from the User Discovery servers, since all their submissions are performed over cMix. Therefore, from the perspective of the User Discovery servers, the communication involves only the Contact Manager Server or the Hash Manager Server, and the cMix system.

To initialize the PrivaTegrity User Discovery protocol, users establish a communication channel —through cMix— with the Contact Manager server, which is responsible for registering the public contact information (e.g., email address) of users, along with their corresponding public keys. In addition, the Contact Manager server answers requests for the public keys of users in the system. Secondly, after completing the first phase of the User Discovery protocol, which involves the Contact Manager Server, users proceed to contact the Hash Manager Server, which receives uploads consisting solely of encrypted and hashed contact information, thus being unable to create any type of social graph. Later, this chapter explains that the system is resistant to scenarios where both servers are colluding. However, to simplify the explanation and to distinguish clearly the privacy features of the system, the Hash Manager Server and the Contact Manager Server represent two different entities in the system.

5.2 Design

The PrivaTegrity User Discovery protocol is divided into three parts:

- Registration phase, where users register in the Contact Manager Server by uploading their public contact information, such as an email address, together with a corresponding Diffie-Hellman public key.
- Search phase, which consists in retrieving the public key, of the acquaintances a user wishes to contact, from the Contact Manager Server. This search phase introduces security mechanisms to avoid leaking information about users who are not registered in the system.
- Hash Upload phase, where users calculate a session key based on the retrieved public key and their personal private key, and proceed to upload part of the hash of this session key along with their encrypted cMix ID.

Furthermore, it is important to mention that the examples of the PrivaTegrity User Discovery protocol included in the following sections do not include the cMix mixnet. This approach allows the reader to have a clearer understanding of how the protocol works as opposed to including an additional mixnet abstraction in the diagrams. However, in the real world, the cMix mixnet is present between every transmission that involves a PrivaTegrity user and a server.

5.2.1 Client Registration

This section describes the registration process of two users, Alice and Bob, in the PrivaTegrity system. However, it is important to distinguish the terms registering and signing up. While signing up implies that a user joined the PrivaTegrity system and already owns a cMix ID, registering denotes that users want to share contact information —such as an email address— with the system, so that they can establish secure communication channels with other users they wish to contact.

Initially, both users start by generating a Diffie-Hellman key pair. In the case of Bob, he first generates a random private number $b \in_R \{2,\ldots,p-2\}$, where p is a public prime number common to every user in the PrivaTegrity system. Subsequently, Bob generates a corresponding public key B, where $B = g^b \bmod p$.

After generating the key pair, $\{b,B\}$, Bob uploads his email, which is his public contact information, and the corresponding public key B. The Contact Manager server, after receiving the request for a new registration, sends a confirmation email to the submitted email, to verify that the email does in fact belong to the user requesting the ownership. Bob proceeds to confirm the email address by clicking on the link sent to his email. After this validation, the server adds the email and the public key to the database of registered users, thus completing the registration process of Bob.

Later in time, Alice, who is an acquaintance of Bob, registers in the system by following the same procedure. Alice generates a Diffie-Hellman key pair, $\{a,A\}$, where $a \in_R \{2,\ldots,p-2\}$ and $A = g^a \bmod p$, and uploads her email address as well as her public key. The server, as expected,

responds with a confirmation email, containing a link to validate the entry to the database. Alice confirms that she owns the email and her records are added to the database.

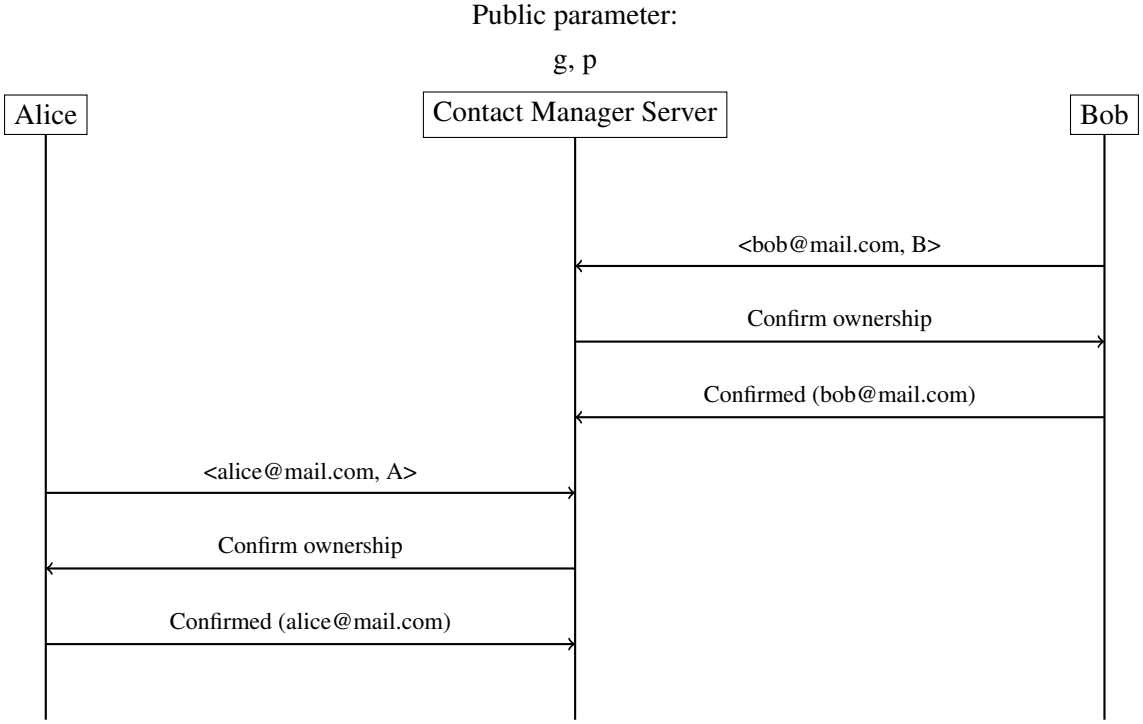

Figure 5.2: Client Registration

After executing this registration procedure, users are added to the PrivaTegrity User Discovery system and have disclosed a contact identifier to the Contact Manager server. However, such server is not able to link the email to a specific endpoint since the registration information is submitted over cMix. Therefore, the Contact Manager server gains information solely regarding a specific email and the corresponding public key originating from the cMix end node.

5.2.2 Alice searches for a user registered in the system

After registering in the system, users desire to start communicating with their acquaintances. Therefore, users must be able to find each other in the system. As such, using the previous registration case as an example, Alice needs to be able to search for Bob. Consequently, Alice sends the Contact Manager Server the email of Bob. The server proceeds to reply with the corresponding public key B, where $B = g^b \mod p$. Alice, using that reply, is able to generate a shared session key, $S_k = B^a \mod p$, with Bob.

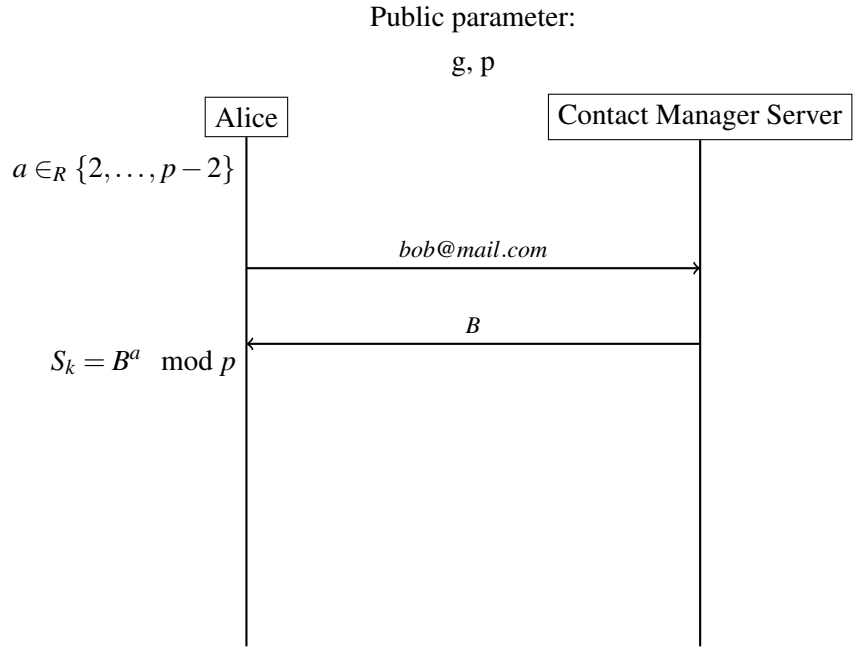

Figure 5.3: Alice looks up a registered user (Bob)

Alice, at this point, owns a session key that can likewise be obtained by Bob, as long as Bob follows the same lookup procedure. After completing this phase, Alice can proceed to the Hash Upload phase, which is the final step of the Private User Discovery protocol.

5.2.3 Alice searches for a user not registered in the system

After registering in the system, Alice desires to look up multiple acquaintances. However, some of the contacts present in the device might not be registered in the system at a certain point. Moreover, in a manner to prevent exhaustive search attacks, where attackers supply multiple email addresses to discover which users are registered in the system, PrivaTegrity includes a mode of operation designed especially in the case where a user is searching for another user that is not yet registered in the system. Therefore, just by querying the Contact Manager Server, from the perspective of the user, it should not be possible to gain information regarding whether a user is or is not registered in the system.

This section illustrates an example where Alice queries the Contact Manager Server by supplying an email corresponding to an acquaintance Charlie, who is not yet registered. The server, since Charlie is not a registered user, creates a fake public key, C', indistinguishable from a valid public key, and replies by sending C' to Alice.

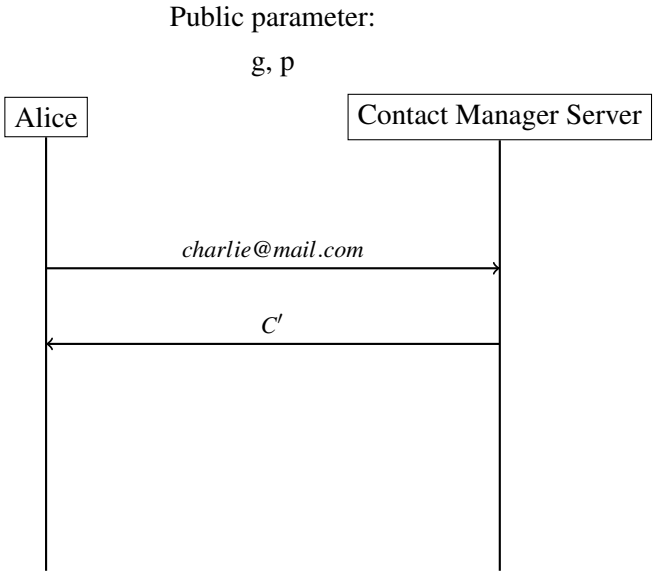

Figure 5.4: Alice looks up a non-registered user (Charlie)

This mechanism of generating a fake public key provides additional privacy features to the system. However, it also introduces multiple constraint that are described in more detail in 5.2.5. Nevertheless, this feature is included in the protocol since the privacy benefits associated with this mode of operation are considered desirable for the PrivaTegrity system.

5.2.4 Hash Upload

The following example illustrates the case where Bob looked up Alice and already possesses the corresponding session key, S_k, and is now ready to initiate the hash upload phase, which is the final step in the PrivaTegrity User Discovery protocol.

Bob starts by hashing the obtained session key, $Hash(S_k)$, and truncating the resulting hash value, to obtain the first 128 bits. Afterwards, Bob uploads to the server the first 128 bits of the hash of the session key, along with his encrypted—with independent 256 bits of the hash of the session key—cMix ID: $<Hash_{0-127}(S_k), Enc_{Hash_{256-511}(S_k)}(cMix\ ID_{Bob})>$.

Subsequently, Alice performs the same set of steps and uploads the following pair: $<Hash_{0-127}(S_k), Enc_{Hash_{256-511}(S_k)}(cMix\ ID_{Alice})>$. This time, however, the server detects a collision of hashes, since Bob uploaded the same hash previously, and replies with the encrypted cMix ID of Bob. As such, Alice is able to decrypt the cMix ID of Bob and may contact him at any time. However, if Alice does not initiate contact and, later in time, Bob wants to verify if Alice established a communication channel with him, Bob can re-upload the same pair, $<Hash_{0-127}(S_k), Enc_{Hash_{256-511}(S_k)}(cMix\ ID_{Bob})>$, and the server will reply with the encrypted cMix ID of Alice, $Enc_{Hash_{256-511}(S_k)}(cMix\ ID_{Alice})$.

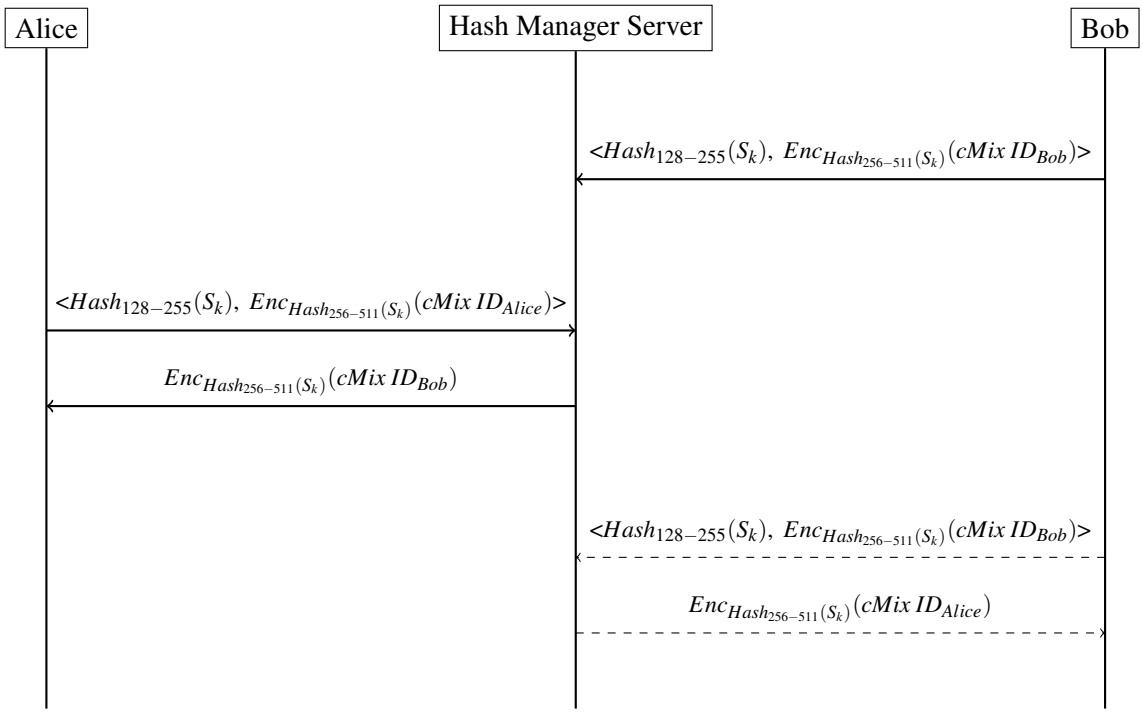

Figure 5.5: Hash Upload

The Hash Upload phase allows users to establish communication channels solely with other users they wish to contact. However, if the other user decides not to initiate an interaction, it forces a more automatic approach to check if a hash collision exists, thus allowing the retrieval of the cMix ID of the user to be contacted.

5.3 Assumptions

The PrivaTegrity User Discovery protocol presents new security features compared to existing User Discovery protocols. However, this Private User Discovery protocol includes specific security goals and assumptions to ensure proper functioning. The initial assumption for the PrivaTegrity User Discovery protocol is that any user in the system can use an anonymous channel for peer-to-peer communications using a pseudonym address, in this case: The cMix mixnet and the corresponding cMix ID. Moreover, such cMix ID is not revealed to the servers involved in the User Discovery process. Otherwise, the Contact Manager Server is able to break the anonymity of the users, since the Contact Manager, besides having access to the email addresses registered in the system, would gain access to the cMix IDs registering such emails. In addition, to use the proposed Private User Discovery protocol properly, users must agree to revealing a personal contact identifier, such as an email address. Finally, regarding the key pairs, the system assumes that Diffie-Hellman is secure in some group, even if a partial hash of a shared session key is revealed.

5.4 Security Goals

The main security goal behind the PrivaTegrity User Discovery protocol is that there are no external parties that have knowledge regarding the set of users who exchanged keys, besides the users. Moreover, the cMix IDs are only revealed when two people wish to establish a communication channel with each other. Therefore, outside entities should be unable to gain access to the cMix ID of users, unless users disclose that information.

In addition, an optional yet included security goal is that users should not able to discover who is registered in the system, just by querying the Contact Manager Server. Therefore, if a user queries the Contact Manager Server for a public key, the response of the server should be indistinguishable in both cases where a user is and is not registered in the system. Nevertheless, the Contact Manager server has access to information regarding the registered email addresses and the corresponding public keys.

5.5 Adversarial Model

Besides the cMix adversarial model, the PrivaTegrity User Discovery protocol assumes an authenticated channel between the cMix system and the Contact Manager Server. Therefore, an adversary is able to eavesdrop, forward and delete messages, yet the adversary is not able to modify, replay, or inject new messages in the communication between the final cMix node and the Contact Manager server without detection. This authenticated channel assumption is fundamental to prevent possible man-in-the-middle attacks against transmissions that involve Diffie-Hellman public keys.

Regarding the User Discovery servers, the system assumes an honest yet curious Contact Manager Server, to ensure key authenticity. The goal of the Contact Manager Server is to compromise

the anonymity of the users and to gain information about the social graph of users. The anonymity of users is compromised if the adversary is able to link an email registered in the system to a cMix identifier. The social graph of users, however, is compromised if the adversary is able to discover the set of users that have established communication channels between each other. To achieve information that compromises the anonymity of the users, the adversary would have to compromise the entire cMix system. Moreover, to gain access to the social circles of users, an adversary needs to gain access to session keys established with each acquaintance of the targeted users. To do so, the adversary needs to compromise the Contact Manager Server and manipulate the responses for each public key request.

The Hash Manager server, on the other hand, does not need to be trusted, except for not performing Denial of Service attacks against the system. Potential malfeasance acts by this Hash Manager Server will be proven ineffective since it only deals with encrypted and hashed information. Moreover, if the channel is not authenticated, the server is able to provide fake replies. However, by providing fake replies, the server is just providing dummy information that will not be properly decrypted nor have any effect on compromising information or users. A potential attack by the Hash Manager server is the case where the server gains access to the encryption key of an encrypted cMix ID present in its database. However, such attack is extremely hard to perpetrate as the key needs to be compromised, and the server would have to perform an exhaustive search in the records, to be able to gain access to a decrypted cMix ID. Moreover, just by knowing the cMix ID of a user, the Hash Manager Server would not be able to link it to the identity of the owner nor to a social graph.

5.6 Implementation

The implementation of the PrivaTegrity User Discovery protocol for this thesis is divided into three Python executables: a Client Application, a Contact Manager Server Application, and a Hash Manager Server Application.

5.6.1 Client Application

The Client Application is responsible for establishing a socket connection with both the Contact Manager and the Hash Manager servers. Moreover, this application generates Diffie-Hellman key pairs, registers users in the system, requests public keys from the Contact Manager Server by supplying an email address, and uploads both hashes and encrypted cMix IDs to the Hash Manager Server. Since this is a cryptographic application, there are multiple requirements necessary to achieve a strong and robust implementation. Factors such as the size of the generated keys and the chosen random number generator, can represent potential attack vectors. As such, respective prevention mechanisms must be implemented.

The Client Application uses a Diffie-Hellman 2048-bit MODP Group with 256-bit Prime Order Subgroup compatible with IETF standards [20], that comply in form and structure with relevant standards from ISO, ANSI, NIST, and the IEEE. Furthermore, this application uses the Python

secrets module, which is recommended for generating cryptographically strong random numbers. Additionally, the private keys generated by the client-side implementation have 256 bits. Finally, when the application requests a public key from the Contact Manager Server, the client verifies the validity of the requested key by performing the calculation of the Legendre symbol of the received public key.

Regarding the transmissions involving the Hash Manager Server, the client application uses the SHA512 hashing function, which provides a 512 bit output. Using SHA512, as opposed to SHA256 for example, takes place since it is necessary to have a significant number of bits as an output in the Hash Upload phase and, the output of 256 bits associated with SHA256, is not sufficient to fulfill the requirements of the application. Finally, the client application uses the AES cipher block chaining mode (CBC) for the encryption and decryption of the cMix IDs.

5.6.2 Contact Manager Server Application

The Contact Manager Server application is responsible for registering users in the database, sending verification emails and, in addition, replying to users when they perform a contact search. Moreover, this application features the generation of fake public keys in the case where a user is not registered in the system.

5.6.2.1 TLS

This section discusses a solution for a potential Man-in-the-Middle attack where a malicious actor, Eve, can position herself in the middle of the endpoints and perform active attacks on the transmission of public keys. By doing so, Eve generates a key pair, e, E, and is able to provide a user requesting a public key, in this case Alice, with a fake public key, $E = g^e \mod p$, and later compromise the anonymity of Alice by linking her email address with her cMix ID.

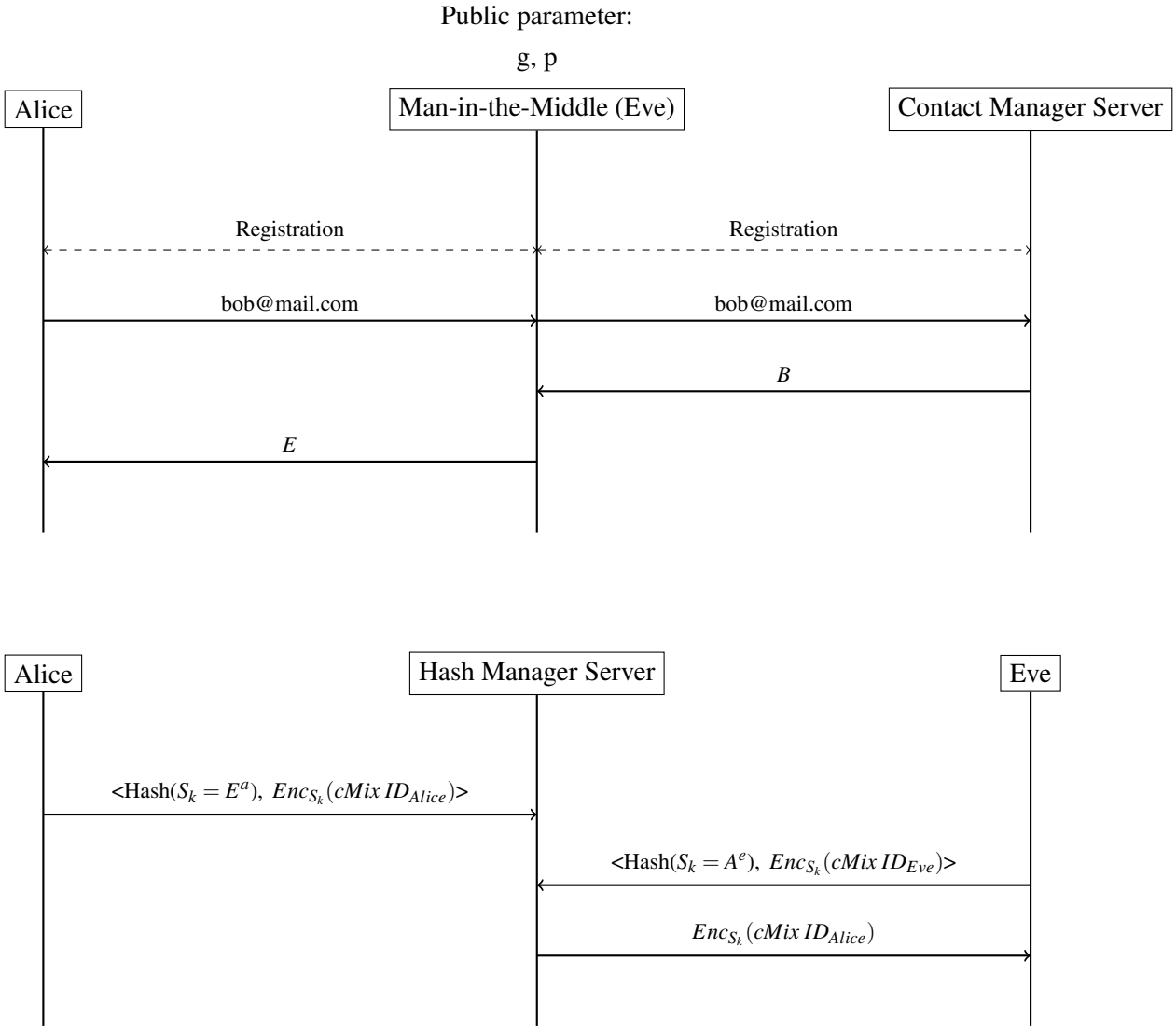

Figure 5.6: Man-in-the-Middle Active Attack

Fig 5.8 shows that Eve, the Man-in-the-Middle, is able to gain access to the cMix identifier of Alice, even though Alice never intended to leak this contact information to anyone but the user she was trying to establish a communication channel with, in this case Bob. Moreover, this active attack on the protocol allows Eve to masquerade as Bob.

A possible solution for this attack vector is the implementation of the Transport Layer Security (TLS) protocol, thus creating an authentication mechanism for the server. Furthermore, by using message integrity check with a message authentication code (MAC), it is possible to prevent tampering of information during the communication.

5.6.2.2 Generating a Public Key for non-Registered Users

There are multiple solutions to the mode of operation where the Contact Manager Server replies with a fake public key in the case where a user with a specific email is not registered. However, this mechanism of generating fake public keys is necessary to ensure that a potential adversary is not able to distinguish both original and fake public keys just by querying the system.

An initial solution, where the Contact Manager Server is trusted and secure, could include hashing the email address of the user together with a freshly generated salt. Since the private keys used in the system are 256 bits long, by generating the SHA256 output of the email and a salt, the system could create the following key pair for an unregistered user: $< c = SHA256(email + salt)$, $C' = g^{SHA256(email+salt)} \mod p >$. However, such assumptions of a trusted and not compromised server, implies that the server has access to both the public and private key. Therefore, the server, if ever compromised, is able to break the anonymity —by linking the cMix ID and the email address—of multiple users.

In addition, the final and implemented solution, generates a random number —using a cryptographically secure pseudo-random number generator— with the same size as a legitimate public key and proceeds to save it in the database, so that the server always replies with the same value. This solution of generating a random number, introduces a potential problem, attackers might be able to use mathematical expressions, such as the Legendre symbol, to verify if the key is valid or not. However, this solution introduces an efficient mechanism of generating a fake public key and, if compromised, the attacker gains information regarding whether a user is or is not registered in the system, which is not a potential vulnerability in the system, yet it represents an undesirable privacy leak.

Conclusively, the generation of a fake public key represents an open problem in the protocol, since it is a detail that is not fixed in a secure cryptographic manner. If we consider the Kerckhoff's principle, where a system should be secure even if everything about the functioning of the system —except the keys— is known to the adversary, the initial solution of having the Contact Manager Server generate a "temporary" key pair provides robustness about the attack where an attacker aims to find out whether a user is or is not registered in the system. However, if the Contact Manager Server is compromised, then such server is able to break anonymity of users in the system. Moreover, the generation of fake public keys with no corresponding private keys, forces the client application to query the Contact Manager Server multiple times throughout time, as the keys of the users one wishes to contact might change. Furthermore, the use of fake public keys with no corresponding pairs, introduces additional usage of space in the database server, as the Hash Upload represents a collision that will not take place, thus representing unnecessary information.

5.6.2.3 Fixed Response Time

The PrivaTegrity User Discovery protocol produces a fake public key in case a user is not registered in the system. To do so, the system checks the database to verify if the requested email address has

a corresponding public key. Subsequently, since the record will not be present in the database, the server has to generate a fake key corresponding to the email address. By introducing additional steps, the server will be adding delays comparing to the normal procedure where the server would just look up the database and reply with the corresponding public key. Consequently, this allows attackers to perform side-channel attacks, where it is possible to obtain information by analyzing the execution of the application rather than by exploring a weakness in the system.

To prevent side-channel attacks, the system introduces a fixed response time for every case where a user searches for a contact identifier. Introducing a fixed response time introduces a delay in the normal responses. However, this security mechanism represents a protection against an attack vector and, as such, was implemented in the Contact Manager Application.

5.6.3 Hash Manager Server Application

The Hash Manager Application is in charge of processing uploads of information pairs that contain a hash and an encrypted cMix ID. Consequently, to store the information uploaded by the users of the system, this application is connected to a PostgreSQL database and, similarly for the Contact Manager server, uses prepared statements. As such, SQL injections attacks from the client side against the database are limited.

Besides adding records to the database, the Hash Manager server frequently deals with collisions that happen when a user submits an already existing hash. When a collision occurs, the Hash Manager server sends the client the corresponding cMix ID, so that the client can proceed to decrypt it and obtain information about the user whom he wishes to contact.

The Hash Manager Server application represents an executable that has a limited mode of operation as it features solely database accesses and replies. Therefore, this application does not require a substantial amount of security mechanisms against potential attack vectors.

Chapter 6

Security Analysis

Since this thesis describes the analysis and implementation of a new Private User Discovery mechanism proposed for an anonymous communication system, it is important to have a detailed security analysis of the PrivaTegrity UD protocol, in order to obtain information about potential vulnerabilities present in the design of the protocol. Consequently, Section 6.1 describes an analysis of the protocol using empirical research, and Section 6.2 features a security analysis using Tamarin prover, a protocol analysis tool.

6.1 Empirical Analysis

This section provides an empirical analysis and study of the potential attack surface in the PrivaTegrity User Discovery protocol, which comprises four interacting entities: The cMix mixnet, PrivaTegrity users, and the Hash Manager and Contact Manager servers.

Regarding the Contact Manager Server, the proposed protocol assumes an honest yet curious server, which implies that the replies containing the public keys are legitimate. On the other hand, the Hash Manager Server represents an entity that does not require any type of trust, except for not initiating Denial of Service attacks against the system. Moreover, the PrivaTegrity User Discovery protocol considers an adversary that aims to compromise the anonymity of users and to gain access to their corresponding social graph. The anonymity of users is broken when an external party is able to link an email address to a cMix ID. Moreover, the social graph is compromised when an attacker gains access to the list of users that tried to establish contact with each other.

From the perspective of attacks against users, it is possible for an attacker to try to claim the ownership of a contact identifier that belongs to a different user. However, by including the step of adding confirmed email addresses to the database, it is possible to mitigate this attack vector. Nevertheless, an attacker may try to use an email address similar to that of a real user, thus potentially impersonating a different user. However, if the users use the information from the contact list stored in their devices, then such attack is less likely to succeed since the information present in the device of a user tends to be accurate.

In addition, it is possible for a client to receive a message from someone claiming to be a particular contact. However, such message might originate from an attacker impersonating such user. As such, if users use the session key to authenticate the initial message, it is possible to detect this impersonation attack. Moreover, a malicious user can supply a fake cMix ID in the hash upload phase. Such fake submission would originate a deviation from protocol as it potentially establishes a communication channel between an unexpected pair of users. However, if the session key is used to authenticate the initial message, then this attack vector results in a generation of spam in the system, thus not affecting the security properties of the system.

Furthermore, since the system uses an identifier for each user in the system, an attacker may try to enumerate every cMix ID and send malicious messages to multiple users expecting to deceive at least a small percentage of users, as in phishing attacks. To prevent against this type of attack, it is possible to introduce authentication with session keys in the initial message. Moreover, a more extreme solution includes the client application to automatically reject messages that are not authenticated with any of the established session keys, thus avoiding this type of attack.

In addition, the Hash Manager Server may deal with enormous data quantities being stored in its internal storage system, thus causing a Denial of Service against the system. It is possible to prevent this scalability problem by adding two different solutions. First, once a collision occurs and the Hash Manager returns a cMix ID, the server can proceed to remove the entry from the Database, since it is no longer necessary. Second, in order to prevent the system from having excessive amount of information that may or may not be helpful, we can introduce the concept of Time to live in each entry, thus limiting the lifespan of each entry and removing information not used in the system.

Moreover, from the perspective of colluding servers, it is possible to break the anonymity of some users. For example, if Alice, shortly after registering, queries the Contact Manager Server for a non-registered user Bob, then she will get a fake public key corresponding to Bob. Consequently, the Contact Manager Server knows that Alice just registered in the system and that shortly after, a query for Bob exists. The Contact Manager Server, after generating a fake key pair for Bob, can combine this information with the public key of Alice and generate the session key used by Alice for the Hash Upload. By performing this attack, the Contact Manager Server can retrieve the encrypted cMix ID of Alice, thus breaking the anonymity of the user.

On the other hand, from the perspective of attacks against the PrivaTegrity User Discovery servers, if the Contact Manager Server is compromised, then an attacker gains access to the list of the email addresses of users registered in the system, together with the corresponding public keys. However, the attacker is not able to link this information to the corresponding cMix IDs. Even though this leak of information represents an attack against the system, it is yet ineffective since it does not reveal any of the information that an attacker wishes to obtain to break the anonymity of users in the PrivaTegrity system.

6.2 Automated Protocol Verification Using Tamarin Prover

Cryptographic protocols represent a fundamental need for security and privacy in the online world. However, disclosure of structural flaws in existing cryptographic protocols occur recurrently, which implies that multiple of these protocols lack a formal and extensive analysis of their security properties. Consequently, there may be multiple undiscovered vulnerabilities in existing protocols. As such, this section introduces the use of Tamarin prover [21] to analyze the PrivaTegrity User Discovery protocol. Nevertheless, there are multiple alternative tools in the field of automated protocol analysis, which includes tools such as Cryptographic Protocol Shapes Analyzer (CPSA) [22], Maude-NPA [23], Athena [24], ProVerif [25], AVISPA [26], and Scyther [27]. However, it is important to note that each of the aforementioned tools has limitations and performs distinctively [28] according to the constraints and mathematical operations associated with each protocol.

To perform an automated protocol verification, Tamarin receives a security protocol model as an input. Such protocol model specifies the actions —taken by the multiple agents in the system— along with a specification of the adversary and the desired properties of the protocol. After processing the input, Tamarin constructs a proof that features multiple instances of the protocol together with the actions of the adversary in each instance. Therefore, Tamarin displays possible attack graphs in each execution instance of the protocol, which allows the disclosure of multiple existing attacks in a protocol. However, such extensive attack analysis may also result in a never ending execution of the simulation.

6.2.1 Modeling the Protocol

The PrivaTegrity User Discovery protocol includes four different entities in the system: The cMix mixnet, PrivaTegrity users, Contact Manager Server, and Hash Manager Server. As such, it is necessary to model a protocol in the Tamarin prover language that comprises the four entities and their security assumptions, together with a correct modeling of the proposed protocol.

6.2.1.1 Channel properties

Multiple security and privacy features of the PrivaTegrity depend on cMix, a mixnet designed to provide anonymity to its users. However, Tamarin prover does not yet include support for the use of anonymous communication channels such as Tor or cMix [30]. Therefore, modeling anonymous channels using automated tools remains yet an unsolved problem. Therefore, the modeling of the PrivaTegrity User Discovery protocol featured in this thesis does not include the assumption that every user communicates with the Private User Discovery servers through cMix or any type of anonymous channel. However, by combining secure channels and authentication, it is possible to simulate a mode of operation of the PrivaTegrity User Discovery protocol that includes a slightly changed attack surface yet maintains multiple of the original assumptions.

6.2.1.2 User Registration Phase

In the PrivaTegrity User Discovery protocol, users start the registration phase by uploading to the Contact Manager Server a contact identifier and a corresponding public key. Subsequently, the Contact Manager Server replies with a confirmation email, which is later confirmed by the user.

Modeling an exact replica of the registration phase is not possible in Tamarin prover. However, this Contact Manager Server can be modeled as a public key infrastructure (PKI). Moreover, since the Contact Manager Server contains Diffie-Hellman public keys, it is possible to model this server as a Diffie-Hellman Public Key Infrastructure. Using the Tamarin prover initial protocol example that is featured in the user manual —which contains a Public Key Infrastructure— and since Tamarin supports Diffie-Hellman exponentiations, by changing the type of key generation in the provided PKI example, it is possible to model the information that the Contact Manager Server accordingly to the PrivaTegrity User Discovery protocol.

Furthermore, to simulate the use of TLS between the user and the Contact Manager server, it is possible to model instances where messages are sent over a secure channel, which implies that an adversary in Tamarin prover can neither modify nor learn messages that are sent over such secure channel. Nevertheless, an adversary can store messages for replay at a later point in time

On the other hand, it is not possible to model the email confirmation step. However, by introducing authentication in the initial registration submission, Tamarin assumes that the submitted information belongs in fact to the respective user, thus simulating an approximate variant of the email confirmation step.

6.2.1.3 User Lookup Phase

The user lookup phase represents consists of uploading a contact identifier to the Contact Manager Server. Afterwards, the Contact Manager Server replies with a public key. However, as mentioned in Chapter 5, this public key might be a fake key.

By having multiple modes of operation for the user lookup phase, the Contact Manager Server introduces branching in the system. It is yet unclear if Tamarin prover supports branching. However, for an initial simulation, the Tamarin prover model proposed in this thesis considers that the queried user is registered in the system and that the server is able to reply with the corresponding public key. As such, and by changing the example in the Tamarin prover manual that features a PKI, it is possible to replicate this phase.

6.2.1.4 Hash Upload Phase

The hash upload phase features an interaction between a user and the Hash Manager Server, which stores a pair of information that contains a truncated hash and a cMix ID encrypted with different bits from the same hash.

Consequently, the mode of operation of the Hash Manager Server represents a complicated modeling problem since Tamarin prover does not support the truncation of variables. Therefore, the protocol needs to be modeled in a manner where the user uploads a slightly variant of the

expected pair. A possible solution features the upload of the hash of the session key and the encryption of the cMix ID using the session key, instead of using independent bits of the hash of the session key. Otherwise, the protocol models a scenario where the user uploads a hash of a session key and a cMix encrypted with the same hash, which creates a scenario where the user reveals the encryption key, and consequently the cMix ID, to the Hash Manager Server.

Furthermore, according to the PrivaTegrity User Discovery, the connection between the user and the Hash Manager Server is not secure. Therefore, to model this phase, it is possible to use the default channel rules in Tamarin prover, which consider the standard Dolev-Yao model, where the adversary sees all communication, is able to block, alter and redirect traffic at will.

6.2.1.5 Attack Vectors

This research work does not achieve the complete execution of the Tamarin prover simulation of the proposed protocol, therefore there are no attack vectors revealed by Tamarin Prover. Nevertheless, this thesis comprises a detailed description regarding the multiple obstacles associated with modeling the PrivaTegrity User Discovery protocol and provides a new insight on the difficulties associated with protocol analysis using automated tools.

Chapter 7

Discussion, Open Problems, and Future Work

This thesis analyzes and implements a new Private User Discovery protocol designed, by David Chaum, for a new communication system called PrivaTegrity. However, multiple questions appear throughout this research period. This section aims to cover problems that are yet unsolved as well as future objectives.

7.1 Node Failure

The PrivaTegrity User Discovery protocol assumes that the Contact Manager Server and the Hash Manager Server are highly reliable entities in the system. Therefore, occurrences where one or both servers fail, represent an extremely rare event. However, the case where one or multiple nodes stop working is considered, as it leads to the cessation of the User Discovery system.

A possible solution is to establish multiple replicated servers to create a redundant connection and, therefore, in case of failure, the client application can connect to a different server. However, this approach introduces additional implementation costs.

Similarly, node failure might also occur in the cMix mixnet. However, cMix is prepared to handle scenarios where one or multiple nodes fail, as it incorporates a detection mechanism that readjusts the mode of operation accordingly to node failures.

7.2 Alternative Uses

The PrivaTegrity User Discovery protocol allows users to share encrypted information —cMix IDs— with other users as long as the involved parties possess contact information about each other. Therefore, using the same principle, users are able to share other types of information instead of a cMix IDs. As such, it is possible to apply this process to an asynchronous communication system, where a user communicate with another user by submitting variants of the partial hash of the shared key together with an encrypted message. Later, the other user uploads the same partial hash and

retrieves the corresponding message. This mode of operation represents a communication system where the Hash Manager Server stores all the transmitted information yet gains no information regarding which users are communicating and the contents of the transmission as such information is encrypted.

In addition, multiple applications might be able to use or modify the proposed protocol to perform a different functionality. Such option is possible and may represent an interesting approach to multiple privacy issues.

7.3 Open Problems

Private User Discovery represents a fundamental privacy issue present in existing anonymous communication systems. Therefore, there are defined problems that are yet unsolved. As such, this section features multiple open problems present in the PrivaTegrity User Discovery protocol.

7.3.1 Generating Fake Public Keys

The PrivaTegrity User Discovery protocol features a mechanism where users can query a Contact Manager Server to obtain the public key of a user registered in the PrivaTegrity system. Moreover, if such queried user is not registered in the user, the Contact Manager Server generates a fake public key to prevent users from gaining information regarding whether a user is or is not registered just by querying the system. However, the mechanism to generate fake keys represents an unsolved problem as it might be possible to infer whether a key is or is not legitimate by performing specific mathematical calculations, such as the Legendre symbol. As such, a mechanism which does not involve strong trust assumptions from the Contact Manager Server that is able to generate indistinguishable public keys, is necessary to solve this problem.

7.3.2 Malicious Users

The PrivaTegrity User Discovery protocol assumes multiple interactions between users and the Contact Manager and Hash Manager servers. Frequently, interaction between these parties tend to achieve the expected mode of operation. However, multiple cases where malicious users can upload fake information to the server and affect the privacy of a subset of users might exist. As such, this represents an open problem, since users might be able to perform a new set of powerful attacks, designed to break the anonymity of other users.

7.3.3 Blocking Users

Occasionally, users in communication systems deal with unexpected messages from undesired senders. In the real world, communication systems tend to allow users to block other users. Such a feature is not incorporated in the PrivaTegrity system, yet addressing this issue is slightly peculiar as it includes privacy concerns, such as whether a blocked user should or should not know if another user blocked him. Moreover, decisions regarding where such block should occur are

yet unclear. If the system allows users to block other users then, the cMix system can block the message before entering the system or, before leaving the final mix node. However, if the cMix system is excluded from the blocking process, then this issue can be handled by the client application. However, by handling the blocking process in the client application, unnecessary information is transmitted across the cMix system.

7.3.4 Registration with Pseudonym

Multiple users might feel uncomfortable with registering their personal email in the PrivaTegrity system. As such, users can create an alias, or pseudonym, email and register that email instead. However, registering with a different email will generate a problem in the system. The user registering a pseudonym email needs to share that contact information with the users he may wish to contact in the first place.

7.3.5 Optimizing the User Discovery Process

Even though the PrivaTegrity User Discovery represents an advance in Private User Discovery, multiple performance enhancements might exist. Diverse implementations may feature similar principles associated with the PrivaTegrity User Discovery. However, by incorporating a different mode of operation it might be possible to introduce performance speedups.

7.4 Future Work

This section features multiple existing goals for this research project on Private User Discovery.

7.4.1 Kali Linux Tamarin Prover Installation Manual

During this thesis, multiple issues appeared while installing Tamarin prover. Consequently, after contacting the development team of Tamarin prover, both the thesis author and the development team agreed to cooperate. Therefore, one future goal is to add a clearer and detailed description of the installation process for the Kali Linux operating system to the official Tamarin prover manual.

7.4.2 Integrating PrivaTegrity User Discovery

The final result of this thesis includes a wide variety of implemented features of the PrivaTegrity User Discovery protocol. Therefore, together with the team that developed the cMix prototype, the implementations presented in this thesis can be included in the testing PrivaTegrity implementation.

7.4.3 Analysis Using Tamarin Prover

Since the analysis of the PrivaTegrity User Discovery protocol is not yet concluded, one of the proposed goals for this research process is to finish the analysis with Tamarin prover, the automated

tool for protocol analysis. By doing so, it might be possible to discover new attack vectors existing in the proposed protocol.

Chapter 8

Conclusion

This thesis represents four months of research in Private User Discovery. As a result, this work starts by featuring the contrast between existing User Discovery mechanisms. Afterwards, this thesis includes a detailed analysis of a new Private User Discovery protocol for the PrivaTegrity system, along with multiple details and considerations to achieve a robust implementation with strong security properties. This chapter outlines the main contributions of this research.

The PrivaTegrity User Discovery protocol aims to solve the central question in Private User Discovery: *How can users of a communication system establish a communication channel with other users without revealing their personal social graph to the server of the communication system?* After concluding this thesis, the answer to this question is that it is in fact possible to perform a type of User Discovery where the social graph of the users is kept private. However, every solution requires multiple constraints for such protocols to achieve the desired mode of operation.

In the PrivaTegrity User Discovery protocol, users must be able to use an anonymous communication channel for peer-to-peer communications, cMix, that uses a pseudonym address —cMix ID— which is revealed purely to acquaintances users wish to contact. By using a pseudonym address, there is no leak of information directly associated with the personal identity of the user. Moreover, the PrivaTegrity User Discovery protocol assumes that users already possess contact information relative to the users they may wish to contact and that users are comfortable with sharing one contact identifier with the server responsible for managing the users of the PrivaTegrity system. However, such contact identifier, such as an email address, is not linkable to a pseudonym address since every interaction with servers from the communication system involve the cMix mixnet. Therefore, if such assumptions are true, by using the PrivaTegrity User Discovery protocol, users are able to discover other acquaintances privately and establish an encrypted communication channel.

In addition, there are multiple contributions present in this research work. The main contribution, however, is that this thesis features the first implementation of the PrivaTegrity User Discovery protocol. Moreover, since this research work represents the first approach to this PrivaTegrity User Discovery protocol, it is possible that there are multiple enhancements yet to be added. Nevertheless, this thesis includes diverse refinements to the initial design that reduce the

potential attack surface of the protocol, which represents a gratifying first step.

On the other hand, this thesis introduces different applications that feature the principles behind the PrivaTegrity User Discovery protocol. By using the key agreements techniques that represent the foundation of the proposed protocol, it is possible to achieve a wide range of diverse applications that remain with strong privacy requirements. Therefore, this research may introduce advantageous contributions in the privacy field.

In conclusion, this research represents a progress in Private User. Moreover, even though the proposed Private User Discovery protocol is designed especially for the PrivaTegrity system, it provides a protocol that is able to perform conjointly with diverse anonymity systems. Similar Private User Discovery applications involving anonymous systems such as Tor should work accordingly, as long as the multiple necessary constraints exist in such system.

References

[1] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[2] David Chaum. Privategrity User Discovery, 2016.

[3] Andrew C. Yao. Protocols for secure computations. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 00:160–164, 1982.

[4] Nadim Kobeissi. Cryptocat. https://crypto.cat, 2011.

[5] Moxie Marlinspike. Open Whisper Systems. https://whispersystems.org, 2013.

[6] Moxie Marlinspike. The Difficulty of Private Contact Discovery. https://whispersystems.org/blog/contact-discovery, 2014.

[7] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum, 1982.

[8] Stavros Korokithakis. A privacy-preserving contact discovery server. https://github.com/SilentCircle/contact-discovery, 2015.

[9] David Chaum. The Next Social Media We Want and Need! https://www.wired.com/2016/01/the-next-social-media-we-want-and-need/, 2016.

[10] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri de Ruiter, and Alan T. Sherman. cMix: Anonymization by high-performance scalable mixing. Cryptology ePrint Archive, Report 2016/008, 2016. http://eprint.iacr.org/2016/008.

[11] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[12] 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*, SOSP '15, pages 137–152, New York, NY, USA, 2015. ACM.

[13] Henry Corrigan-Gibbs and Bryan Ford. Dissent: Accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 340–350, New York, NY, USA, 2010. ACM.

[14] Chen Chen, Daniele Asoni, David Barrera, George Danezis, and Adrian Perrig. HORNET: High-speed onion routing at the network layer. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, 2015.

[15] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptol.*, 1(1):65–75, March 1988.

[16] The Invisible Internet Project. https://geti2p.net/en/, 2003.

[17] Andrei Serjantov, Roger Dingledine, and Paul F. Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien A. P. Petitcolas, editor, *Information Hiding*, volume 2578 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2002.

[18] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, 2009.

[19] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A Framework For Analyzing Anonymous Communication Protocols. In *Proceedings of the of the 26th IEEE Computer Security Foundations Symposium (CSF)*, pages 163–178. IEEE, 2013.

[20] Matt Lepinski and Stephen Kent. Additional Diffie-Hellman Groups for Use with IETF Standards. RFC 5114, 2008.

[21] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *Proceedings of the 25th International Conference on Computer Aided Verification*, CAV'13, pages 696–701. Springer-Verlag, 2013.

[22] CPSA: Symbolic cryptographic protocol analyzer. https://hackage.haskell.org/package/cpsa.

[23] Maude Tools: Maude-NPA. http://maude.cs.illinois.edu/w/index.php?title=Maude_Tools:_Maude-NPA.

[24] Dawn Xiaodong Song, Sergey Berezin, and Adrian Perrig. Athena: A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1/2):47–74, 2001.

[25] ProVerif: Cryptographic protocol verifier in the formal model. http://prosecco.gforge.inria.fr/personal/bblanche/proverif/.

[26] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In *Proceedings of the 17th International Conference on Computer Aided Verification*, CAV'05, pages 281–285, Berlin, Heidelberg, 2005. Springer-Verlag.

[27] The Scyther Tool. https://www.cs.ox.ac.uk/people/cas.cremers/scyther/.

[28] Cas J. F. Cremers, Pascal Lafourcade, and Philippe Nadeau. Comparing state spaces in automatic protocol analysis. In *Formal to Practical Security*, volume 5458/2009 of *Lecture Notes in Computer Science*, pages 70–94. Springer Berlin / Heidelberg, 2009.

[29] Tamarin Prover. https://tamarin-prover.github.io/manual/book, 2003.

[30] David Basin, Cas Cremers, and Catherine Meadows. *Model Checking Security Protocols*, chapter 24. Springer. To appear.

[31] Jérémy Jean. TikZ for Cryptographers. http://www.iacr.org/authors/tikz, 2016.

Appendix A

Acronyms and Abbreviations

AES	Advanced Encryption Standard
ANSI	American National Standards Institute
CBC	Cryptographic Protocol
CPSA	Cryptographic Protocol Shapes Analyzer
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISO	International Organization for Standardization
MAC	Message Authentication Code
MODP	Modular Exponential
NIST	National Institute of Standards and Technology
OWS	Open Whisper Systems
PGP	Pretty Good Privacy
PIR	Private Information Retrieval
PKI	Public Key Infrastructure
SCCDP	Silent Circle Contact Discovery Protocol
SHA	Secure Hash Algorithm
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UD	User Discovery

Appendix B

Tamarin Prover

B.1 Installation

Even though the Tamarin-Prover Manual [29] includes an installation section, it lacks a clear and detailed explanation of the complete installation process. Since in the first install attempt in this thesis, multiple installation issues not reported in the manual occurred, this section aims to include details regarding the installation process in a (Kali) Linux machine, so that future users can experience an easier installation.

Initially, before starting the installation process, the user should execute the following commands in the terminal:

- sudo apt-get update
- sudo apt-get upgrade

The first command updates the list of available packages and the corresponding versions. Moreover, the second command installs a more recent version of the packages installed in the machine. After concluding this initial process, users should install, from the Linux Package Manager, both Maude —the high performance logical framework— and The Haskell Tool Stack. Moreover, after installing stack, it is necessary to upgrade the installed —yet possibly outdated— version. To do so, the user should execute the following command:

- stack upgrade

After installing multiples dependencies from the Package Manager, it is possible to execute the following commands to install the remaining dependencies for Tamarin prover from the Linux terminal:

- sudo apt-get install graphviz
- sudo apt-get install libz-dev
- sudo apt-get install ocaml

- git clone https://github.com/tamarin-prover/tamarin-prover
- make default

The previous commands install multiple dependencies together with the Tamarin prover tool. After executing the make default command, the system is supposed to compile over 120 dependencies from scratch. As such, this process is supposed to last over 30 minutes. Once the installation is done, the user should verify whether the tool is properly installed in the computer. Consequently, the user can execute the following commands:

- cd ~/.local/
- xdg-open bin

The last execution from the previous set opens the folder 'bin', which contains multiple files from the Tamarin prover installation. After opening the folder, the user can confirm if the Tamarin prover executable is in the bin folder, along with the FirstExample.spthy file. If so, the user can proceed to execute the Tamarin prover tool for the first time.
To execute Tamarin prover, the terminal should be in the directory which contains the executable files. Therefore, users need to execute the following two commands, which change the directory and execute the test file that verifies if Tamarin prover is properly installed:

- cd ~/.local/bin
- ./tamarin-prover test

After executing the last set of commands, the terminal should display the following message, confirming that the test execution does not contain any errors.

```
$ ./tamarin-prover test
Self-testing the tamarin-prover installation.

*** Testing the availability of the required tools ***
maude tool: 'maude'
checking version: 2.7. OK.
checking installation: OK.

GraphViz tool: 'dot'
checking version: dot - graphviz version 2.38.0 (20140413.2041). OK.

*** Testing the unification infrastructure ***
Cases: 55  Tried: 55  Errors: 0  Failures: 0

*** TEST SUMMARY ***
All tests successful.
The tamarin-prover should work as intended.

:-) happy proving (-:
```

At this point, the installation process is concluded and the terminal should display a message notifying the user that all tests were successful and that Tamarin prover should work as expected. Moreover, since the tool is installed, the user may execute the example file —FirstExample.spthy— provided by the Tamarin team to verify if the tool was successfully installed and if it works accordingly.
To execute Tamarin prover with a graphical user interface, the user should run the following command:

- ./tamarin-prover interactive FirstExample.spthy

The previous command starts an interactive version of Tamarin prover —that is accessible via any browser in 127.0.0.1:3001— which should display the following welcome screen:

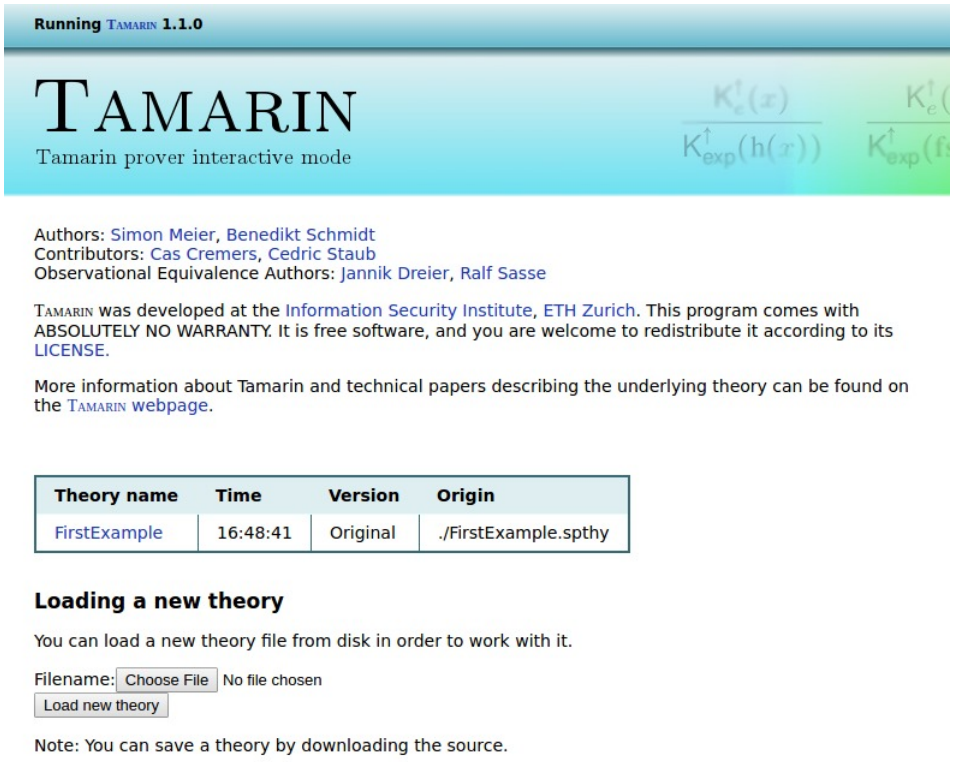

Figure B.1: Tamarin Prover Welcome Screen

The table, located in the middle of the previous figure, displays the loaded theory, or file, which contains the formal definition of an example protocol. At this point, the user finished installing the tool and can start using the Tamarin prover tool for protocol analysis.

B.2 Source Code

```
/*
User Discovery Protocol Analysis with Tamarin Prover
====================================================

Author:    Mario Yaksetig Costa
Date:        May 2017

*/

theory UserDiscovery
begin

builtins: diffie-hellman, hashing, signing, asymmetric-encryption, symmetric-
    encryption

/*
### PROTOCOL EXECUTION STARTS HERE ###
*/


/*
REGISTRATION PHASE
*/


// Client Generates DH Key Pair and uploads public identifier ($A) and Public
    Key ('g'^~a)
rule Client_1:

let

m = <A, 'g'^~a>

in

[ Fr(~a), !Pk($A, 'g'^~a), m ]
--[ Send(A,m) ]->
[ Client_1($S, $A, $B, ~a, 'g'^~a), Out(m) ]


// Contact Manager Server registers user in the System
rule CMServer_1:

[ !Pk($B, 'g'^~b), In(m) ]
--[ Recv(S,m), Authentic(A,m) ]->
[ !Pk($A, 'g'^~a), !Pk($B, 'g'^~b) ]
```

```
46
/*
48 LOOKUP PHASE
*/
50

52 // Alice uploads the contact identifer of Bob, to retrieve his corresponding
    Public Key.
rule Client_2:
54
let
56
m = B
58
in
60
[ Client_1(S, A, B, a, 'g'^a) ]
62 --[ Send(A,m) ]->
[ Client_1(S, A, B, a, 'g'^a), Out(m) ]
64

66 // Contact Manager Server Receives Request for Bob's Public Key
rule CMServer_2:
68
[ !Pk(A, 'g'^a), !Pk(B, 'g'^b), In(m) ]
70 --[ Recv(S,m) ]->
[ !Pk(A, 'g'^a), !Pk(B, 'g'^b) ]
72

74 // Contact Manager Server replies with Bob's Public Key
rule CMServer_3:
76
let
78
m = 'g'^b
80
in
82
[ !Pk(A, 'g'^a), !Pk(B, 'g'^b) ]
84 --[ Send(S, m) ]->
[ !Pk(A, 'g'^a), !Pk(B, 'g'^b), Out(m) ]
86

88 // Alice receives Bob's Public Key
rule Client_3:
90
[ Client_1(S, A, B, a, 'g'^a), In(m)  ]
92 --[ Recv(A, m) ]->
[ Client_3(S, A, B, a, 'g'^a, 'g'^b) ]
```

```
94
96 /*
   HASH UPLOAD PHASE
98 */

100
   // Alice calculates the Diffie-Hellman Session Key with Bob's public key and
       her Private Key
102 rule Client_4:

104 let

106 sk = 'g'^b^a

108 in

110 [ Client_3(S, A, B, a, 'g'^a, 'g'^b) ]
   -->
112 [ Client_4(S, A, B, a, 'g'^a, 'g'^b, sk, h(sk)) ]

114
   // Alice uploads the pair of the hash of the session key and her encrypted cMix
        ID
116 rule Client_5:

118 let

120 m = <h(sk), enc(cMix_A, sk)>

122 in

124 [ Client_4(S, A, B, a, 'g'^a, 'g'^b, sk, h(sk)) ]
   --[ Send(A, m) ]->
126 [ Client_5(S, A, B, a, 'g'^a, 'g'^b, sk, h(sk), enc(cMix_A, sk)), Out(m) ]

128

130 // Hash Manager Server registers a hash entry in the DB
   rule HMServer_1:
132
   [ h(sk), enc(cMix_B, sk), In(m)
134 ]  --[ Recv(S,m) ]->
   [ HashServer(h(sk), enc(cMix_A, sk), enc(cMix_B, sk)), Out(enc(cMix_B, sk)) ]
136

138 // Alice receives the encrypted cMix of Bob, because there was a collision.
   rule Client_6:
140
```

```
  [ Client_5(S, A, B, a, 'g'^a, 'g'^b, sk, h(sk), enc(cMix_A, sk))
142 , In(m)
  ]
144 --[ Recv(A, m) ]->
  [ Client_6(S, A, B, a, 'g'^a, 'g'^b, sk, h(sk), dec(cMix_B, sk)) ]
146


148 /*
  LEMMAS START HERE
150 */

152 lemma executable:
  exists-trace
154 "Ex A S m #i #j. Send(A,m)@i & Recv(S,m) @j"

156 lemma message_authentication:
  "All a m #j. Authentic(a,m) @j ==> Ex #i. Send(a,m) @i &i<j"
158

  end
```