Privacy-Preserving Contact Discovery with Applications to End-to-End Encrypted Messaging and Mobile-First Cryptocurrencies

Nicolas Mohnblatt[1]

Supervised by Dr. Philipp Jovanovic

A dissertation submitted in partial fulfilment
of the requirements for the degree of
MSc in Information Security
at
University College London.

August 29, 2020

[1]**Disclaimer:** this report is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

Acknowledgements

Contents

Chapter 1

Introduction

Privacy-oriented services such as end-to-end encrypted messaging are increasingly popular [7]. While they provide strong cryptographic guarantees for the confidentiality of message contents, many still leak or gather user-related data. This is particularly the case during a setup stage known as *contact discovery*. As a result, some of these applications gain access to their users' address book and therefore their mobile social graph [8, 9]. In this project, we are interested in performing *contact discovery* in a privacy-preserving manner while remaining practical for mobile applications with billions of users.

1.1 What is contact discovery?

Contact discovery (alternatively *contact matching*) simply refers to the process by which users of a service are able to find other users to interact with. The applied method is largely determined by the amount of information users choose to make public. In the case of networks such as Facebook or LinkedIn, users are encouraged to publish their legal names and can therefore be found through a simple search. In the cases we study, users are registered using pre-existing human-readable identifiers such as their phone numbers or email addresses. This information is kept private by the service such that only users with prior knowledge of each other's identifier can communicate.

As a user signs up to such a service, she will already hold an *address book* – a register that links people (often referred to as *contacts*) to their identifier. However, phone numbers and email addresses are identifiers generated by other services and there is no guarantee that all her *contacts* are using the new service. Thus in this context, *contact discovery* is more precisely defined as the process by which a user can discover whether or not her *contacts* are using a specific service. Notice that such a process is not only a necessary initialisation step; it must also be regularly refreshed to ensure users keep an up-to-date view of the contacts they can address.

1.2 The privacy challenge

The simplest way to perform contact discovery is arguably to send one's address book to the service operator, allowing them to compute the intersection between the address book and the list of registered users. This is in fact how the popular messaging services WhatsApp and Telegram perform their contact matching [8, 9]. Although efficient, this approach reveals large amounts of private information about users and their contacts, including those that are not register for the service. The service operator is able to construct a social graph of its users and their first connections, allowing it to check for individual connections at will or under government pressure. Such information may discourage whistleblowers from ever speaking up, in fear that their identity may be revealed if they are linked to journalists.

Hash Functions – A naive approach using only cryptographic hash functions will also fail to meet our goal [4, 5]. A user could upload hashes of her contact's identifiers for the service operator to compare against hashes of the registered users' identifiers. While this approach is efficient and yields the desired result, it will still leak the user's address book.

Indeed, although the cryptographic hash function is pre-image resistant, the set of possible pre-images is small enough that hashes can be precomputed into a dictionary and used to find the identifiers that underly the uploaded hashes [5]. Salting these hashes to avoid offline computations renders the system unusable since the service operator would be required to hash the set of registered identifiers using a different salt for each attempt at contact discovery [4].

Advanced approaches and Efficiency – In light of the above, more advanced approaches have been developed to perform privacy-preserving contact discovery. We cover these in greater detail in chapter 2. The issue with such approaches is that they introduce additional complexity through computations, communication requirements, storage requirements or a combination thereof.

In the context of the services we study, contact discovery needs to be performed on mobile devices on a regular basis. These devices are less powerful than modern desktop computers and rely on rechargeable batteries. A computation-intensive process ran regularly on such a device could quickly drain its battery. Furthermore we must allow the process to scale elegantly with the number of registered users, and assume that it can grow to the order of billions.

Efficiency therefore constitutes a priority in the design of such contact discovery schemes. It will also provide a benchmark to evaluate systems against each other, provided that they guarantee a satisfactory level of privacy.

1.3 A peer-to-peer approach

In this report, we present a peer-to-peer approach that makes use of pairing-based cryptography. By doing so, we reduce the service operator's role to a minimum and provide clients with the tools to compute shared secret keys with their contacts. Computations on the client side are of linear order with respect to the size of their address book. Furthermore, clients are only expected to communicate with the service during set-up and are only required to store short cryptographic material.

1.4 Structure

Chapter 2

Related Work

In this chapter we provide an overview of state-of-the-art methods for privacy-preserving contact discovery, as well as academic attempts at solving a similar problem. These methods can be divided according to their underlying approach: the first aims at computing the intersection between a list of registered users and an address book, the second aims at providing users with the necessary cryptographic material needed to authenticate and establish shared secrets between each other

In section 2.1, we cover Signal's approach which is to simply process each user's address book without storing her contacts [6]. To convince users that they are trustworthy, Signal publish their code and allow their servers to be audited remotely. In section 2.2, we investigate cryptographic ways to perform a set intersection between two parties without either party learning the other's data. This is known as a private set intersection (PSI). The subsequent attempts fall under the second approach described above. Thus section 2.3 focuses on public key infrastructure and section 2.4 on identity-based key exchanges.

2.1 Public source code and remote attestation

2.1.1 Signal and Intel SGX

Signal's approach is arguably the simplest: request a user's address book, process it against the list of registered users and clear the servers from any knowledge linked to it [6]. While this process may seem trivial, it creates new challenges in terms of security and user trust. First, Signal must guarantee that no knowledge of the address book remains on the server, be it obtained through regular or side channels. Secondly, Signal needs to earn the trust of it users. Not only do they need to convince users that their process is completely oblivious, they must also provide constant evidence that their servers are running that particular process rather than any other.

They meet both challenges by publishing their server-side code and performing all their processing within “secure enclaves” on their servers.

2.2 Private set intersection (PSI)

2.3 Public key infrastructure (PKI)

2.4 Identity-based key exchange (IBKE)

Chapter 3

Background

Before we introduce our system, we recall some definitions of lesser known cryptographic primitives and assumptions. Our aim is to provide the necessary technical background to then discuss our system's architecture. Alternatively, readers may proceed to chapter 4 and refer back to this section when needed.

3.1 Bilinear pairings

The following definition for a *pairing* is that provided by Boneh and Shoup in *A Graduate Course in Applied Cryptography* [2]. To remain consistent with the source text, group operations are represented multiplicatively.

Definition 3.1 (Pairing [2]) *Let $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ be three cyclic groups of prime order q where $g_0 \in \mathbb{G}_0$ and $g_1 \in \mathbb{G}_1$ are generators. A **pairing** is an efficiently computable function $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$ satisfying the following properties:*

1. *bilinear: for all $u, u' \in \mathbb{G}_0$ and $v, v' \in \mathbb{G}_1$ we have*

$$e(u \cdot u', v) = e(u, v) \cdot e(u', v) \qquad \text{and} \qquad e(u, v \cdot v') = e(u, v) \cdot e(u', v)$$

2. *non-degenerate: $e(g_0, g_1)$ is a generator of $\mathbb{G}_T$*

*When $\mathbb{G}_0 = \mathbb{G}_1$, we say that the pairing is a **symmetric pairing**. When $\mathbb{G}_0 \neq \mathbb{G}_1$, we say that the pairing is an **asymmetric pairing**. We refer to $\mathbb{G}_0$ and $\mathbb{G}_1$ as the **pairing groups**, or source groups, and refer to $\mathbb{G}_T$ as the **target group**.*

From the bilinear property, we can derive the following equality which is central to our scheme:

$$\forall \alpha, \beta \in \mathbb{Z}_q,\ e({g_0}^{\alpha}, {g_1}^{\beta}) = e(g_0, g_1)^{\alpha\beta} = e({g_0}^{\beta}, {g_1}^{\alpha}) \tag{3.1}$$

Hard Problems in Pairing Groups – The existence of pairings has direct consequences on the discrete logarithm, the decisional Diffie-Hellman (DDH) and the computational Diffie-Hellman (CDH) assumptions. We summarise these in Table 3.1 below.

	Symmetric Pairing $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$	**Asymmetric Pairing** $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$
Discrete Logarithm	No harder in $\mathbb{G}_0$ than in $\mathbb{G}_T$	No harder in $\mathbb{G}_0$ or $\mathbb{G}_1$ than in $\mathbb{G}_T$
Decisional DH	Easy to solve in $\mathbb{G}_0$, assumed to hold in $\mathbb{G}_T$	Assumed to be hard in $\mathbb{G}_0$, $\mathbb{G}_1$ and $\mathbb{G}_T$
Computational DH	Assumed to be hard in $\mathbb{G}_0$ and $\mathbb{G}_T$	Assumed to be hard in $\mathbb{G}_0$, $\mathbb{G}_1$ and $\mathbb{G}_T$

Table 3.1: Summary table of classic cryptographic problems under pairings

There exist variants of the DDH and CDH assumptions that take into account the pairing operation: the decisional variant is known as the decision Bilinear Diffie-Hellman (DBDH) assumption and the computational variant is known as the co-Computational Diffie-Hellman (co-CDH) assumption. We provide formal definitions for both of the assumptions in Appendix A.

Implementation – Pairings have been implemented in practice on certain pairing-friendly elliptic curves. While the underlying constructions are outside of the scope of this project, we wish to emphasise a few of their features. In asymmetric pairings, the group $\mathbb{G}_0$ is usually built upon a finite field, while groups $\mathbb{G}_1$ and $\mathbb{G}_T$ are built on extensions of that field [2]. This implies that elements in $\mathbb{G}_0$ have a shorter representation than those in $\mathbb{G}_1$ or $\mathbb{G}_T$. Furthermore, operations in $\mathbb{G}_0$ are less computationally intensive. Finally, a pairing operation is much more computationally intensive than exponentiation in any of the three groups [2].

3.2 BLS signatures

One application for pairings is to create deterministic and homomorphic signature schemes such as the one introduced by Boneh, Lynn and Shacham [1] – named BLS after all three of the authors. In this scheme, signatures are elements of one source group and public keys are elements of the other. Although we will make use of both variants, we only present the variant in which signatures are elements of $\mathbb{G}_0$ and public keys are elements of $\mathbb{G}_1$. Once again we write group operations multiplicatively to remain consistent with the source material.

Definition 3.2 (BLS Signatures [1]) *A BLS signature scheme $\mathcal{S}_{BLS}$ is composed of three efficient algorithms* $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}$*. Let H_0 a cryptographic hash function defined as $H_0 : \{0,1\}^* \to \mathbb{G}_0$, and "$\leftarrow_\$$" denote the "choose uniformly at random" operator, we define the three algorithms as:*

> $\mathsf{KeyGen}(\kappa)$: *Choose q, a prime of κ-bits and three cyclic groups $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ of order q such that there exists a pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$ (see definition 3.1). $g_0 \in \mathbb{G}_0$ and $g_1 \in \mathbb{G}_1$ are generators. Choose uniformly at random $x \leftarrow_\$ \mathbb{Z}_q^*$ and set the secret key $\mathsf{sk} \leftarrow x$ and the public key $\mathsf{pk} \leftarrow {g_1}^x$. Output sk to the message signer and pk to the receiver.*
>
> $\mathsf{Sign}(\mathsf{sk}, m)$*: Output the signature $\sigma = H_0(m)^{\mathsf{sk}}$.*
>
> $\mathsf{Verify}(\sigma, m, \mathsf{pk})$*: If $e(\sigma, g_1) = e(H_0(m), \mathsf{pk})$ accept the signature. Otherwise reject.*

Theorem 3.1 (Security of BLS Signatures [2]) *Let $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a pairing, let $\mathcal{M}$ be the message space and let $H : \mathcal{M} \to \mathbb{G}_0$ be a hash function. Then the derived BLS signature scheme is* ***existentially unforgeable under chosen message attacks*** *assuming the co-Computational Diffie-Hellman assumption[1] holds for e, and H is modelled as a random oracle.*

Blind and/or threshold variants of this scheme exist. The former allows to hide the original message from the signer, while the latter allows to hide the complete signature from any individual (non-colluding) signer.

[1] see Appendix A

3.3 Left/Right constrained pseudorandom functions

Left/right constrained pseudorandom functions were first introduced by Boneh and Waters [3]. These pseudorandom functions (PRFs) are evaluated over a pair of inputs x, y with a random key k – we denote the output value as $F(k,(x,y))$. These functions can then be "constrained" to their left or their right input using *constraining keys*: knowing the left constraining key for a specific value w allows to compute $F(k,(w,y))$ at all points y with no knowledge of k. Similarly, the right constraining key for a value w allows to compute $F(k,(x,w))$ at all points x with no knowledge of k. Left/right PRFs are formally defined in [3] as:

Definition 3.3 (Left/right constrained PRF [3]) *Let $F : \mathcal{K} \times \mathcal{X}^2 \to \mathcal{Y}$ be a PRF. For all $w \in \mathcal{X}$ we wish to support constrained keys $k_{w,\mathrm{LEFT}}$ that enable the evaluation of $F(k,(x,y))$ at all points $(w,y) \in \mathcal{X}^2$, that is, at all points in which the left side is fixed to w. In addition, we want constrained keys $k_{w,\mathrm{RIGHT}}$ that fix the right hand side of (x,y) to w. More precisely, for an element $w \in \mathcal{X}$ define the two predicates $p_w^{(L)}, p_w^{(R)} : \mathcal{X}^2 \to \{0,1\}$ as*

$$p_w^{(L)}(x,y) = 1 \iff x = w \quad \text{and} \quad p_w^{(R)}(x,y) = 1 \iff y = w$$

We say that F supports left/right fixing if it is constrained with respect to the set of predicates

$$P_{LR} = \left\{ p_w^{(L)}, p_w^{(R)} : w \in \mathcal{X} \right\}$$

Security – A left/right constrained PRF is said to be secure if for all probabilistic polynomial time adversaries, the output of the PRF is indistinguishable with a non-negligible advantage from an element chosen uniformly at random in the output domain.

Implementation – Boneh and Waters [3] present a secure left/right constrained PRF construction under the random oracle model by making use of a symmetric pairing. Here we present a variant that makes use of asymmetric pairings. Let $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ be three cyclic groups of prime order q such that there exists a pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$. Let $H_0 : \{0,1\}^* \to \mathbb{G}_0$ and $H_1 : \{0,1\}^* \to \mathbb{G}_1$ be two hash functions modelled as random oracles. For a random key k, we define the left/right constrained PRF F as:

$$F(k,(x,y)) = e(H_0(x), H_1(y))^k \tag{3.2}$$

For $w \in \{0,1\}^*$, the constraining keys for the predicates $p_w^{(L)}$ and $p_w^{(R)}$ are:

$$k_{w,\mathrm{LEFT}} = H_0(w)^k \quad \text{and} \quad k_{w,\mathrm{RIGHT}} = H_1(w)^k \tag{3.3}$$

Using the bilinear property of the pairing, we can check that knowing $k_{w,\mathrm{LEFT}}$ allows to evaluate $F(k,(w,y))$ for all $y \in \{0,1\}^*$:

$$e(k_{w,\mathrm{LEFT}}, H_1(y)) = e(H_0(w)^k, H_1(y)) = e(H_0(w), H_1(y))^w = F(k,(w,y)) \tag{3.4}$$

A similar equality can be written to check that $k_{w,\mathrm{RIGHT}}$ allows to evaluate $F(k,(x,w))$ for all $x \in \{0,1\}^*$ by computing $e\left(H_0(x), k_{w,\mathrm{RIGHT}}\right)$.

Notice that left/right constrained PRFs and BLS signatures are closely related. Indeed they both make use of the same underlying pairing construction. Furthermore, BLS signatures take the same form as a constraining key, namely a group element raised to an unknown power.

Chapter 4

Pairing-Based Contact Discovery

In this chapter we present the architecture for our contact discovery service (section 4.1). We then provide outlines of security proofs (section 4.2), theoretical performance evaluations (section 4.3) and show how our system maps onto real-world applications such as end-to-end encrypted messaging and mobile-first cryptocurrencies (section 4.4).

4.1 Service architecture

The foundational design principle for our system is to provide users with the means to perform contact discovery locally. As we have seen in chapter 2, sending a client the full list of registered users in a probabilistic data structures such as Bloom and Cuckoo filters requires the client to download and store large amounts of data. Instead, we follow an approach similar to the IBKE protocols. Our scheme runs in three phases which we will investigate individually:

1. **Setup:** a one-time step for each user. During the setup phase, a user interacts with the contact discovery service to obtain her unique cryptographic material.

2. **Key derivation:** using this cryptographic material, the user is able to compute shared secret keys with any of her contacts knowing only their discovery identifier (e.g. a phone number or email address).

3. **Discovery:** using their shared secret key, a pair of users can establish a secure meeting point on an online cache, thus allowing for asynchronous contact discovery.

Figure 4.1 shows a diagram of the process described above.

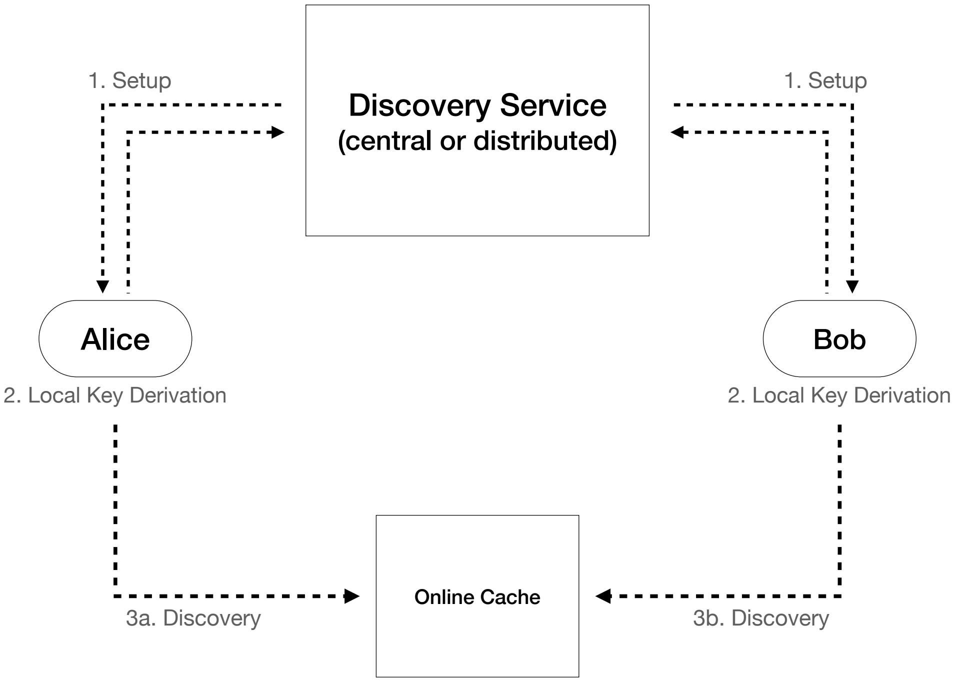

Figure 4.1: Contact discovery between a pair of users Alice and Bob, including setup. Numbers indicate the order of execution

4.1.1 Actors, assets and notation

We make a brief aside to clarify the actors and assets present in our scheme:

- **Users:** each user A hold a **key pair** $(\mathsf{sk}_A, \mathsf{pk}_A)$ provided by her mobile application. Furthermore, A holds a **discovery identifier** $\mathtt{id}_A$ which could be a phone number, an email address or a username. We represent A's **address book** as a set of discovery identifiers $\mathcal{ID}_A$, which is itself a subset of the set of all possible discovery identifiers $\mathcal{ID}$. Finally, we assume that users exchanged discovery identifiers through out-of-bound communication but are unable to exchange cryptographic material, including their public keys.

- **Discovery Service:** the discovery service S may be a central or distributed entity (in the latter case we denote the set of all servers as $\mathcal{S}$ and the i-th server as S_i). In the centralised version, the discovery service holds a master secret key pair **msk**. In the distributed version, each server holds a share of the master secret key $\mathbf{msk}_i$.

- **Online Cache:** the online cache may be operated by the discovery scheme or by a third party and is assumed to be untrusted. Its role is to manage key-value pairs.

Next we define the cryptographic setting for our scheme:

- q is a prime of κ bits
- $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ are three cyclic groups of order q such that there exists a pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$.
- H_0 and H_1 are two public hash functions modelled as random oracles such that $H_0 : \mathcal{ID} \to \mathbb{G}_0$ and $H_1 : \mathcal{ID} \to \mathbb{G}_1$.
- $F : \mathcal{ID}^2 \times \mathbb{Z}_q \to \mathbb{G}_T$ is a left/right constrained PRF defined as:

$$F(k, (\mathtt{id}_A, \mathtt{id}_B)) = e\left(H_0(\mathtt{id}_A), H_1(\mathtt{id}_B)\right)^k \tag{4.1}$$

- **KDF** is a public, deterministic key derivation function.
- The master secret key **msk** is set to an element $s \in \mathbb{Z}_q$ chosen uniformly at random. In the distributed variant, we assume that the master secret key is shared according to a secure t-out-of-n secret sharing scheme and that no single entity holds the master secret key.

4.1.2 Key derivation

We first introduce the core key derivation step even though it is performed after the setup step. In doing so, we provide the reader with the necessary material to understand the security constraints under which the setup phase operates.

For all users $B \in \mathcal{ID}_A$, user A can compute shared key material with B by evaluating $F(\mathbf{msk}, (\mathtt{id}_A, \mathtt{id}_B))$ and $F(\mathbf{msk}, (\mathtt{id}_B, \mathtt{id}_A))$. From the definition of left/right constrained PRFs, A can do so with the constraining keys $k_{\mathtt{id}_A,\mathrm{LEFT}}$ and $k_{\mathtt{id}_A,\mathrm{RIGHT}}$:

$$F(\mathbf{msk}, (\mathtt{id}_A, \mathtt{id}_B)) = e\left(k_{\mathtt{id}_A,\mathrm{LEFT}}, H_1(\mathtt{id}_B)\right) \tag{4.2}$$

$$F(\mathbf{msk}, (\mathtt{id}_B, \mathtt{id}_A)) = e\left(H_0(\mathtt{id}_B), k_{\mathtt{id}_A,\mathrm{RIGHT}}\right) \tag{4.3}$$

Similarly, B can evaluate F at the same points using the constraining keys $k_{\mathtt{id}_B,\mathrm{LEFT}}$ and $k_{\mathtt{id}_B,\mathrm{RIGHT}}$:

$$F(\mathbf{msk}, (\mathtt{id}_A, \mathtt{id}_B)) = e\left(H_0(\mathtt{id}_A), k_{\mathtt{id}_B,\mathrm{RIGHT}}\right) \tag{4.4}$$

$$F(\mathbf{msk}, (\mathtt{id}_B, \mathtt{id}_A)) = e\left(k_{\mathtt{id}_B,\mathrm{LEFT}}, H_1(\mathtt{id}_A)\right) \tag{4.5}$$

Using this key material, A and B can establish a symmetric secret key using a standardised key derivation function:

$$k_{AB} = k_{BA} = \mathbf{KDF}\left(F(\mathbf{msk}, (\mathtt{id}_A, \mathtt{id}_B)), F(\mathbf{msk}, (\mathtt{id}_B, \mathtt{id}_A))\right) \tag{4.6}$$

A note on security – The constraining keys $k_{\mathtt{id}_A,\mathrm{LEFT}}$ and $k_{\mathtt{id}_A,\mathrm{RIGHT}}$ allow to compute every symmetric key that A may establish with her contacts. As such, those **constraining keys must remain private** to A. The consequences of a leak range from impersonation to a total leak of A's address book and are further detailed in section 4.2.

4.1.3 Setup

The setup stage serves to provide user A with the constraining keys $k_{\mathtt{id}_A,\mathrm{LEFT}}$ and $k_{\mathtt{id}_A,\mathrm{RIGHT}}$. Consequently, the setup is a security-critical task. As we have shown in Equation 3.3, under our construction of F the constraining keys can be expressed as:

$$k_{\mathtt{id}_A,\mathrm{LEFT}} = H_0(\mathtt{id}_A)^{\mathbf{msk}} \quad \text{and} \quad k_{\mathtt{id}_A,\mathrm{RIGHT}} = H_1(\mathtt{id}_A)^{\mathbf{msk}} \tag{4.7}$$

These constraining keys can be seen as BLS signatures by the discovery service on A's discovery identifier. Notice that the service needs to produce signatures under both variants of the BLS scheme: one with signatures in $\mathbb{G}_0$ and one with signatures in $\mathbb{G}_1$.

4.1.4 Discovery

4.2 Security

4.2.1 Outlines of security proofs

4.2.2 Consequences of a breach

4.2.3 Authentication: an open problem

4.3 Theoretical performance evaluation

4.4 Applications

4.4.1 End-to-end encrypted messaging

4.4.2 Mobile-first cryptocurrencies

Chapter 5

Proof-of-Concept Implementation

5.1 Local server emulation

5.2 Local key derivation

5.3 Online meeting point via IPFS

Chapter 6

Conclusion

Appendix A

Bilinear variants of the CDH and DDH problems

A.1 The co-computational Diffie-Hellman (co-CDH) Problem and Assumption

The co-Computational Diffie-Hellman (co-CDH) assumption is a variant of the Computational Diffie-Hellman assumption that applies for asymmetric pairings. Let us recall the definition for the co-Computational Diffie-Hellman assumption given in [2], using a multiplicative notation for the group operation as in the source text..

Attack Game A.1 (co-CDH [2]) *Let $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ be three cyclic groups of prime order q such that there exists a pairing $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$. For a given adversary $\mathcal{A}$, the attack game runs as follows:*

- *The challenger picks at random $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$ and computes*

$$u_0 \leftarrow {g_0}^{\alpha}, \qquad u_1 \leftarrow {g_1}^{\alpha}, \quad v_0 \leftarrow {g_0}^{\beta}, \quad z_0 \leftarrow {g_0}^{\alpha\beta}$$

- *The adversary $\mathcal{A}$ receives the tuple (u_0, u_1, v_0) and outputs $\hat{z}_0 \in \mathbb{G}_0$*

We define the advantage of $\mathcal{A}$ in solving the co-CDH problem for e as:

$$\text{coCDHadv}[\mathcal{A}, e] := \Pr(\hat{z}_0 = z_0) \tag{A.1}$$

Notice that for symmetric pairings, $\mathbb{G}_0 = \mathbb{G}_1$ therefore $g_0 = g_1$, $u_0 = u_1$ and attack game A.1 is identical to the Computational Diffie-Hellman attack game.

Definition A.1 (co-CDH Assumption [2]) *We say that the co-CDH assumption holds for the pairing e if for all efficient adversaries $\mathcal{A}$ the quantity* $\mathrm{coCDHadv}[\mathcal{A}, e]$ *is negligible.*

A.2 The decision bilinear Diffie-Hellman (DBDH) Problem and Assumption

The decisional variant is relatively straight-forward having already defined the co-CDH assumption. The attack setting is closely related, however the adversary is expected to distinguish an element from random (rather than required to computed it). Once again, the definition is adapted from [2] and uses a multiplicative notation for group operations.

Attack Game A.2 (Decision bilinear Diffie-Hellman [2]) *Let $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a pairing where $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$ are cyclic groups of prime order q with generators $g_0 \in \mathbb{G}_0$ and $g_1 \in \mathbb{G}_1$. For a given adversary $\mathcal{A}$, we define the following experiment:*

- *The challenger picks at random $\alpha, \beta, \gamma, \delta \leftarrow_{\$} \mathbb{Z}_q$, computes*

$$u_0 \leftarrow {g_0}^{\alpha}, \quad u_1 \leftarrow {g_1}^{\alpha}, \quad v_0 \leftarrow {g_0}^{\beta}, \quad w_1 \leftarrow {g_1}^{\gamma}, \quad z^{(0)} \leftarrow {g_0}^{\alpha\beta\gamma}, \quad z^{(1)} \leftarrow {g_0}^{\delta}$$

  and flips a bit $b \leftarrow_{\$} \{0, 1\}$. Using the result of the bit flip, the challenger sends $(u_0, u_1, v_0, w_1, z^{(b)})$ to $\mathcal{A}$.

- *$\mathcal{A}$ receives $(u_0, u_1, v_0, w_1, z^{(b)})$ and outputs a bit $\hat{b} \in \{0, 1\}$*

We define the advantage of $\mathcal{A}$ in solving the DBDH problem for e as:

$$\mathrm{DBDHavd}[\mathcal{A}, e] := \frac{1}{2} - \Pr(\hat{b} = b) \tag{A.2}$$

Definition A.2 (Decision BDH assumption [2]) *We say that the decision bilinear Diffie-Hellman assumption holds for the pairing e if for all efficient adversaries $\mathcal{A}$ the quantity* $\mathrm{DBDHavd}[\mathcal{A}, e]$ *is negligible.*

Appendix B

Code

Bibliography

[1] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of Cryptography 17(4), pp 297–319 (2004)

[2] Boneh, D., Shoup, V.: A Graduate Course in Applied Cryptography (v0.5). Published online at https://toc.cryptobook.us (January 2020)

[3] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. Cryptology ePrint Archive, Report 2013/352 (2013), https://eprint.iacr.org/2013/352

[4] Kales, D., Rechberger, C., Schneider, T., Senker, M., Weinert, C.: Mobile private contact discovery at scale. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 1447–1464. USENIX Association, Santa Clara, CA (Aug 2019), https://www.usenix.org/conference/usenixsecurity19/presentation/kales

[5] Marlinspike, M.: The difficulty of private contact discovery. https://signal.org/blog/contact-discovery/ (January 2014)

[6] Marlinspike, M.: Technology preview: Private contact discovery for Signal. https://signal.org/blog/private-contact-discovery/ (September 2017)

[7] Reuters: Whatsapp users cross 2 billion, second only to facebook. Online https://www.reuters.com/article/us-whatsapp-users-idUSKBN20626L (February 2020), retrieved on 28th August 2020

[8] Telegram: Telegram privacy policy. https://telegram.org/privacy, retrieved on 18th August 2020

[9] WhatsApp Inc.: Terms of Service. https://www.whatsapp.com/legal#terms-of-service, retrieved on 17th August 2020