Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Metadata Reduction for the Signal Protocol

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)

eingereicht von:	Fabian Kaczmarczyck
geboren am:	6.2.1992
geboren in:	Berlin
Gutachter/innen:	Prof. Dr. Jens-Peter Redlich
	Prof. Dr. Björn Scheuermann

eingereicht am: .... verteidigt am: ....

The messenger Signal encrypts conversations, but the protocol leaks the social graph. This thesis proposes two modifications to hide all links between users. The first is a solution to the open problem of contact discovery. The current implementation needs its clients to upload their address books to a central server to determine who can privately talk with each other and users need to trust the server not to store this information. The second part hides the communication partners of conversations, using a pseudonymous authentication. No further user interaction is required by any of the new methods to maintain usability.

Contents

1 Introduction

Since Edward Snowden's disclosures in 2013, people's awareness of privacy has risen and the technology sector was influenced significantly [20]. Shortly after in 2014, the Signal app was released [32], aiming at providing easy-to-use encryption. Its development was lead by the insight that a good usability is a necessary part of security, giving less opportunity to operate the software incorrectly. The result is a messenger using state-of-the-art encryption that is enabled by default for each conversation.

In the aftermath of Snowden's leaks, the power of mass surveillance was discussed in a presidential debate at Johns Hopkins University, that was eventually directed towards metadata. It provides information about, among others, the participants, time and length of a conversation, in contrast to its content. Nevertheless, the importance of metadata is stated as telling "everything about somebody's life. If you have enough metadata, you don't really need content." [6]

1.1 Motivation

The goal of this thesis is a critical analysis of possible metadata leaks in the Signal protocol. Special attention is paid on means to gather information on the connection between users. A structure containing this data is called the social graph. Consisting of the same elements as its mathematical equivalent, it contains the users as nodes and edges as indicators for their connectedness. In the context of a messenger app, this means an address book entry or a conversation.

Open Whisper Systems, the organization behind Signal, is concerned with the privacy of its users' address books. Protecting this kind of metadata is described as an open problem in a blog entry [18] that contains a protocol weakness and invites readers to suggest solutions. All previously proposed ideas are reported to have practical limitations or are incompatible with the protocol's usability goals, in particular computational cost and bandwidth usage.

There are several reasons why Open Whisper Systems is concerned about a possible data leak that can only be exploited by their own server. Although it is open source [34], users have no means to verify that the program run is neither changed nor extended. But even if Open Whisper Systems can be trusted in this respect, the centralization comes with the risk of governmental agencies enforcing access. Another way to gain knowledge of server-side secrets is a vulnerability in transport encryption or a successful hack. Noted by Open Whisper Systems themselves [28] was the possibility of information disclosure when legally required.

1.2 Contribution

The goal of this work is to reduce the available metadata. As often seen in social applications, there is a trade-off between privacy and user acceptance. Contact discovery is one example where it is hard to find a solution that offers both. As it is stated in [18], Open Whisper Systems currently values usability more and uploads hashed address books of its users. Section 4 proposes a new method that significantly improves contact discovery without requiring additional user interaction.

Gathering the information contained in address books is not the only way to construct a social graph. There are other leaks to deduce edges from, including message passing. This type of information cannot only be collected by the Signal server, but also by exploiting side channels with a statistical attack. The push notification service of users' smartphones, for example Google Cloud Messaging [9], can take advantage of this weakness. Therefore, a protocol extension is discussed in section 5 to protect metadata against a well-equipped attacker. Again, the approach does not need user interaction, but comes with some communication overhead. Whether to use the old or new push service is a choice between conservation of energy and privacy.

All design decisions focus more on complicating mass surveillance than protecting the metadata of individual users. Remaining attacks are evaluated in section 6. Above all, the confirmation of a hypothesis about the existence of a link between two users is possible for a well-equipped attacker. Although reliable information about the social graph is not explicitly transmitted using the protocol extensions proposed in this thesis, evidence can still be collected in specific situations. In summary, the contribution of this work is as follows:

- An analysis of metadata leaks and possible attacks is given.
- The open problem of contact discovery is tackled.
- Changing the push service and adding pseudonymous authentication are shown to make it hard for mass surveillance to reconstruct the social graph.

2 Background & Related Work

After a quick overview of the Signal messenger's current features, different possible solutions and why they do not fit our goals will be discussed.

2.1 Signal Protocol

The Signal protocol was first introduced as the Axolotl or Double Ratchet Protocol by Trevor Perrin and Moxie Marlinspike [31]. It was designed to take into account demands of modern communication. As the current state of the art, it is implemented in a messenger of the same name, also used in WhatsApp and announced for the Facebook Messenger and Google Allo.

The protocol is based on a centralized architecture to allow for offline messaging without lacking usability. Communication between client and server is secured on transport. A message's content is also end-to-end encrypted between clients. This is achieved using a technique similar to Off-the-Record Messaging [3], providing forward secrecy and deniable authentication.

The difference in protocols lies in Signal's capability of offline communication. To allow for key agreements while one party is offline, each user prepares asymmetric key pairs necessary to start a conversation. The respective public key, from now on called prekey, is stored in the central server, ready to be downloaded and used by an interested contact. This means that the first message sent between two users is always preceded by a prekey request to the Signal server.

The protocol contains no peer-to-peer capabilities. This is due to the assumption that mobile devices can not be assumed to be online and transfer should work regardless of the stability of the client's internet connection. A message is therefore always stored in the server which then notifies the recipient about arrivals.

Contact discovery was mentioned above as a currently unsolved problem. In the current Signal software, it is done by uploading hashes of phone numbers to the server and comparing them to the list of registered users. The intersection is sent back to the client. From a privacy perspective, this implementation has a known weakness: Due to the limited space of valid inputs, the hash function can be inverted by a dictionary attack.

2.2 Private Information Retrieval

Open Whisper Systems discusses Private Information Retrieval in [18]. In this field of study, a user wants to query a database without revealing information about himself. In this special case, the server needs to answer queries containing keywords. This active field of current research ([23, 4, 29]) improves privacy often with high computational cost or using a multi-server architecture with the assumption of at least one honest participant. Besides the practical impairments (explained in more detail in [27]), the Signal protocol includes operations changing the server state. Sending the first message requires a prior request for a prekey [33]. The server needs to mark it as used, thus even the most trivial implementation of private information retrieval is not sufficient: sending each user the whole data base. Therefore, a better solution is needed that respects both scalability to millions of users and the limited bandwidth of mobile phone users.

2.3 Software Guard Extensions

An approach to tackle this problem in a different way is using a trusted hardware platform to run the server on [19]. Open Whisper Systems runs Intel's Software Guard Extensions (SGX) that protect the code and data from disclosure and modification [12]. The idea is to use an isolated secure enclave with encrypted memory to prove that the claimed code is indeed executed and the transferred data is kept secret.

This approach appoints Intel as a trusted third party and therefore still has a single point of failure. Another potential problem is the secure enclave's adversary model. Its developer guide states: "Intel SGX does not provide explicit protection from side-channel attacks. It is the enclave developer's responsibility to address side-channel attack concerns." [7]

Various side-channel attacks were demonstrated in recent research [11], [26]. But even a careful, side-channel free implementation of contact discovery does not account for prekey requests since they change the server state and therefore do not run in memory alone.

2.4 Anonymization Networks

This thesis will keep to a low latency approach and not suggest to delay the transport of messages in order to anonymize traffic. A well established mix network is The Onion Router [10]. Whereas other options are conceivable, Tor will be used throughout this work. The attacker model depends on this because the protocol does not intend to defend against a passive attacker with global view [15].

The use case of messaging allows higher wait times than other application level protocols (i.e. sip, streaming) and there are multiple approaches to improve Tor to make it resistant to more powerful adversaries. Current research on mix networks tries to keep their scalability and have a low enough latency for the use cases of Signal. Herd [14] is a new design for VoIP that is also about hiding connections between users. Vuvuzela [30] is concerned with metadata free messaging with a bigger latency. Both work with cover traffic and need to establish a new router network, whereas this paper builds upon existing technology.

3 Design Goals

The main goal is to prevent the mass collection of metadata and complicate linking communication partners. Supporting this objective, more security-related considerations were taken into account.

3.1 Goals

Usability: Like the Signal protocol itself states, security must be transparent to the user and just work. Extra user interaction is not necessary. Nevertheless, educated users can be given means to tune the client to their specific needs.

Parsimony: The client software is running on mobile devices, meaning that energy and bandwidth are limited resources. The proposed extensions will clearly mark spots with increased consumption. They work with a divided user base, allowing each user to opt out of the more expensive features, still leaving him with i.e. improved contact discovery.

Build upon existing infrastructure: A large user base is important for privacy enhancing software due to the size of the anonymity set. Whereas an application-specific medium-latency mix network designed specifically for instant messaging is likely to have better properties, the existing infrastructure of Tor [10] was valued more highly and will be suggested for IP obfuscation throughout the thesis.

Deployability: The extension must integrate well with the existing protocol. Registered users must be able to seamlessly continue using the software.

3.2 Threat Model

The assumed adversary is active, meaning he can generate, modify, delete, delay and replay traffic. He only controls some fraction of the network, meaning the assumptions made by Tor are still valid. Another conclusion is that confirmation attacks are not prevented. Observing two specific users lets the adversary gather evidence about the existence of a link between these two.

The attacker may also compromise central infrastructure, namely the Signal server and the push service providers. Some of the measures included can not prevent abuse entirely, but detect it and report being targeted.

Not included are attacks on the client's mobile device, which is assumed to always work as intended. The possibility to recognize users as registered with Signal is a feature and necessary for contact discovery.

4 Contact Discovery

This section discusses an alternative to uploading a list of hashed entries of users' address books. This new approach is analyzed not only regarding explicitly available information, but also side channels are in our focus. It starts with a brief introduction to blind signatures that will later be used for a new concept called stamps. These will be used in a mechanism for unauthenticated rate control, allowing all queries to only be relatable to one user identity. The rest of the section tries to diminish the chances to associate multiple requests with each other.

4.1 Blind Signatures

Blind signatures were introduced by Chaum [5]. They are meant to hide the document's content to the signer. Independent of the cryptographic primitive, the general mechanics are the following: The server holds a key pair $(K_S^{\text{sig}}, K_S^{\text{ver}})$ and sends the verification key to each client that is requesting a blind signature of some text x. Since x is to be kept secret, it is modified (blinded) to a message x' and is sent to the server for signing. The server-side result is a blinded signature s' that is sent back to the client. After receiving s' and unblinding it to s, the client has a valid signature for x. The pair $t = (x, s)$ is called a stamp later on. A client checks t using the verification key to make sure that the server did not cheat and indeed used the corresponding signing key. The stamp has the desired property that it can only be issued by the server, but checked by everyone. The above steps can be seen in figure 1.

There exist blind signature schemes using elliptic curve cryptography [21]. Described in this section is the conceptionally clearest method. Other implementations are also conceivable, since Signal already uses elliptic curve cryptography for its end-to-end encryption.

4.2 Authentication

The basic idea is to allow unauthenticated users to query the server for their contacts one by one. This way, an individual request can neither be linked to its origin nor to other queries. While technically solving the issue of sending whole address books to the server, there are other ways of finding connections between accounts that will be discussed later in this section. As long as the server can not relate individual requests to each other, it only learns about the existence of nodes in the social graph, but not about their connection. Keeping the nodes hidden is not a goal of this thesis.

Allowing contact discovery queries without authentication exposes the user database to brute force attacks. A rate limit for each user can be accomplished using a new technique that will be called stamps throughout the thesis. A stamp needs to be sent alongside each query for contact discovery. Stamps are issued for successfully authenticated clients and used subsequently. They contain unique

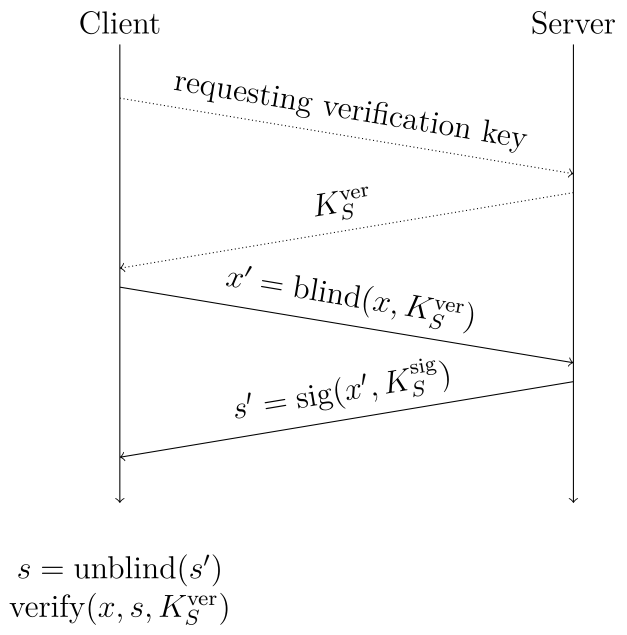
Client
Server
requesting verification key
K_S^{ver}
$x' = \text{blind}(x, K_S^{\text{ver}})$
$s' = \text{sig}(x', K_S^{\text{sig}})$
$s = \text{unblind}(s')$
$\text{verify}(x, s, K_S^{\text{ver}})$

Figure 1: Blind signature request. The communication is initiated by the client, but he needs the verification key before he can compute the blinded text x'. The first part of the handshake can be cached. The received value s' can be unblinded to obtain a valid signature for the initial x.

data that should not be linked to the sender's identity and are therefore created using blind signatures.

To make sure every stamp is used only once and as intended, the nonce x in $t = (x, s)$ is chosen at random and from a range big enough to make collisions highly unlikely. The server accepts each x only once. Depending on the choice of cryptographic scheme, the random number might need a padding to prevent clients from forging signatures [24].

As mentioned before, generating a stamp needs authentication. To preserve the disconnection of sender and message content, the issuance timing should be uncorrelated with the usage. Therefore, it is a requirement of the new protocol that signatures are valid for a long enough time. In contrast, it is in the server's interest to change its signing key in fixed intervals to make sure a user can not collect big amounts of usable signatures and misuse all of them in one burst. A compromise would be to introduce a new signing key e.g. every week and to only use the most recent one to issue signatures, but also accepting the second newest key for incoming requests (see figure 2). The server keeps a list of used nonces for both currently active signing keys and may discard the outdated ones. If the validity period of signing keys is known, a client already knows K_S^{ver} most of the time. It can omit the first part of the handshake when asking for stamps (depicted in figure 1).

The Signal server could deanonymize a particular user by creating a new key

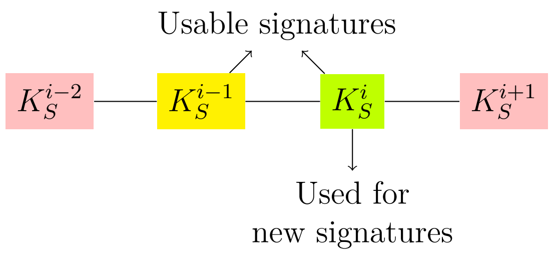

Figure 2: Signing key scheme. In a given episode, only the most recent valid key pair is used to issue stamps, and the previous one is still used for verification.

for each of his requests. This kind of misbehavior can be detected by sharing the signing key K_S^{ver} with other user regularly. The client could automatically send it to its contacts for comparison. Another option is to request K_S^{ver} through Tor (as indicated by the dotted lines ins figure 1). An advantage of this extra Tor circuit is that misbehavior is prevented instead of detected afterwards.

There might be more than one use case for stamps when multiple tasks need to be done anonymously with different allowed rates. To achieve this, the whole process can be duplicated, using new signature keys for each type of request.

4.3 IP Obfuscation

The first side channel that can be used to connect contact discovery queries is the IP address. Clients have the goal to be indistinguishable from each other and should therefore apply a common strategy. An obvious choice is to open a new Tor connection for each request. This discovery operation is not too frequent, so the bigger price for each one is acceptable.

In case an already open Tor connection should be used, other strategies are conceivable. In fact, every choice of exit nodes independent of the last request works, since transport layer encryption makes the server (or someone controlling it) the only possible attacker that has knowledge of the query's content. One example is a fixed exit node for all users in a given time slot.

4.4 Timing

Another side channel that can be exploited by statistical attacks is the timing of events that can be linked to identities. With the idea of stamps in mind, communication with the server is always only linked to the identity of one user, not disclosing an edge of the social graph. All types of message delivery will be considered in the next chapter. Therefore, it remains to analyze

(D) discovery,

(P) prekey requests,

(R) registration,

(S) stamp generation,

where the former two explicitly need to contain the respective contact's address information, but lack authentication due to the stamp mechanism. In contrast, the latter two are authenticated and thus tell the server about the sender's identity.

In the following proofs of the desired properties, the users are assumed to be always online. Clearly, a user sending all of his requests in a short time window makes them easily deanonymizable by correlation, making assumptions about online behavior necessary.

4.4.1 Discovery & Prekey Requests

Types (D) and (P) are similar in this approach, since both contain a contact identifier. Therefore, they can be unified, meaning discovery has no binary answer anymore, but either returns a prekey or failure. Merging improves clarity of the proofs and simplifies the protocol.

Uncorrelatedness of requests of these types is achieved by sending them distributed using a Poisson point process with its memoryless property [13]. This distribution can be used to model independent events. In each time interval I, the number of requests N_I has the distribution

$$\mathbb{P}(N_I = n) = \frac{(\lambda|I|)^n}{n!} e^{\lambda|I|}.$$

The parameter λ controls the rate of requests. Sampling from this distribution is done using the exponential wait time between points in the process. Its probability mass function for sampling is then

$$f(x) = \lambda e^{-\lambda x}$$

and for each discovery query, the client samples the wait time accordingly and chooses a contact to discover uniformly from the remaining address book. Only scheduling one request at a time also has the advantage of easily extending to the non-ideal user who has offline periods. That way, there is no risk of executing multiple queries when getting online.

Since less requests per contact are made as soon as a discovery was successful, the rate λ is proposed as the product $\lambda_0 \cdot n_c$. Here λ_0 is the base rate that is identical for all clients. n_c is the size of the address book left undiscovered. That way, the average number of requests over time for each of these contacts stays constant.

The choice of λ_0 is a trade-off between usability and privacy. A higher rate leads to faster discovery, but comes with more work for the client. Also, when leaving the model of permanently online users, deviations are less likely detected by the server for smaller rates.

4.4.2 Registration

Independence of discovery and registration is hard to achieve. There are two cases that will be analyzed separately: The target is also registered with Signal, or not. In the first case, cover traffic is used to significantly reduce the observable evidence. The second case however is treated less efficiently using a bloom filter. Its feasibility will be discussed at the end of the section in 4.5.

Regarding the first case, registered contacts are already discovered by others and will mostly not be queried again on a regular basis. Thus, a server will observe no requests for longstanding users except by fresh clients. Since this is a very rare event, it suffices to reduce the evidence for links between users obtained by exploiting correlation of discovery and registration. This can be done using cover traffic with clients generating it by requesting prekeys of themselves. The limited supply of public keys which are consumed by the queries needs to be taken into account and replenished appropriately. From the server perspective, legitimate and fake traffic can not be distinguished due to the hidden sender identity.

To prove the efficiency of cover traffic, assume a client A performs a discovery on client B. In the meantime, clients $C_1, \ldots, C_n$ as well as B are querying themselves with an expected value of η requests in the measured time interval, according to a Poisson point process. Due to the stamp system, a malicious observer only sees the total number of requests for each user. The attacker tries to exploit the knowledge of this sum and guesses its maximum as the true recipient. Thus his guess is correct only if B has received the maximum number of requests. For $n \to \infty$, the limit of this probability is 0 (for a proof, see the Appendix). Figure 3 shows the discovery probabilities, compared to a random guess, for various η. It suggests that even small amounts of cover traffic suffice to greatly reduce the risk of a successful correlation attack after registration. Also, it shows that a small number of users can make themselves nearly indistinguishable from each other.

The proof assumes an attacker model with knowledge of a legitimate discovery event for a given time interval, the length of which can be roughly estimated using λ_0. As with the other parameters, the choice of η is a compromise between bandwidth consumption and privacy enhancement.

4.4.3 Stamp Generation

It is essential that requests for stamp generation cannot be associated with discovery. This is easy to achieve with a schedule that is independent of stamp consumption. The requested amount of stamps needs to be fixed, exceeding what is necessary for the real address book size. A usability friendly choice for the timing is whenever the phone is charging, therefore not consuming battery life when needed.

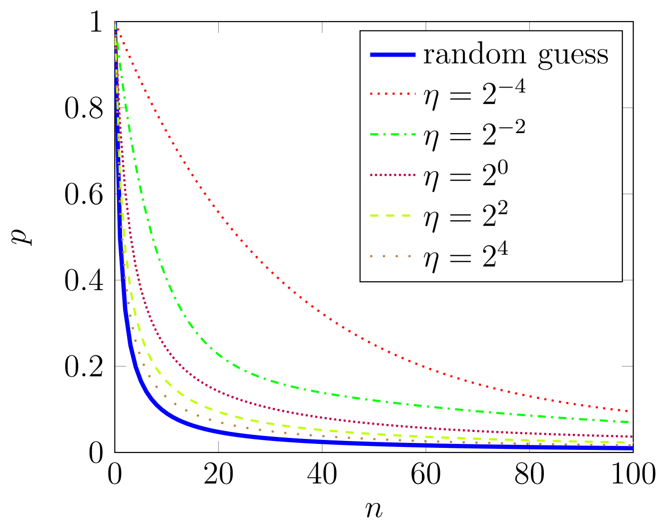

Figure 3: Probability p of finding the correct recipient of a contact discovery compared to a random guess. n is the number of users participating in obfuscation, η is the cover traffic rate.

4.5 Bloom Filter

The last remaining problem mentioned in this section regarding the timing of requests are discovery requests by freshly registered clients on users that do not generate cover traffic. There are multiple reasons for the existence of these unprotected phone numbers. Most commonly, the target does not have an account (yet). Luckily, a binary answer is sufficient in this case, since prekeys are not delivered anyway. Inactive accounts are potentially problematic, requiring either the server to regularly clean up abandoned phone numbers or users to delete them.

An easy solution exists only in the first case of unregistered targets. A bloom filter [2] fits its needs perfectly. It prevents most requests from being made, when chosen big enough to have a low false positive rate. On the other hand, false negatives do not exist. This way, all contacts are still accessible, as long as the bloom filter is built from current data.

That is why bloom filters are not proposed as a solution for contact discovery in general. Keeping it up to date costs bandwidth scaling with filter size and update frequency. In this approach, only the correlation with registration is of interest, so a single bloom filter download along with the installation is sufficient.

The best choice for the number of hash functions used in a bloom filter depends on m and n. Assuming the perfect number of hash functions (which can only be approximated), the relationship between false positive rate p, array size m in bits and number of elements n is:

$$-\log p = \frac{m}{n}(\log 2)^2$$

A policy of keeping p at 0.01 for 10 million registered users leads to a minimum bloom filter size of 11.4 MiB. This necessary download size scales linearly with the number of users, effectively inflating the installation by approximately one byte for each account. An extension of the protocol using sharded bloom filters might be necessary in case of increased usage. Thereby users are divided into several bloom filters, with the idea that only a few of them need to be downloaded. A bigger user base also increases the anonymity set, making sharding a good compromise.

5 Message Delivery

This section discusses approaches to reduce leaks of metadata through message requests. Currently, they contain the sender's identity together with all recipients. The removal of authentication by using stamps on requests to the server for message passing does not work in practice. It neither works against a more sophisticated attacker nor is it easy to use with mobile devices for the following reasons:

- Although the server must not know the sender, the true recipient needs to find out which conversation the new message belongs to.
- The number of stamps necessary is harder to predict than for contact discovery. A large amount of valid stamps needs to be stocked, leading to constant work even for unused clients and the server.
- A common policy for Tor circuits needs to be established for message transmission, so their recipients can not be linked together. This increases the phone's bandwidth usage as well as energy consumption.
- Introducing random delays when sending messages is a big sacrifice to usability. Not doing so exposes metadata using timing attacks.

The last point is the most important one. It is evident considering an observer's perspective. Using the new stamp framework, the server only learns about the recipient, but not the sender. Two users sending messages to each other generate traffic with (approximately) alternating arrival. This is most probably not the only traffic at this time span, but an analysis offers evidence for a connection between parties. In the long run, the server collects enough information to generate a social graph of his users.

Even worse, there are two features that aggravate the above weakness. A client is notified when a message is delivered. Since a sender is not revealing his identity to the server in this case, those acknowledgements are necessarily done by the recipient using the messaging mechanism. An easily recognizable pattern can therefore be observed by the passive attacker. This claim will be backed up by an implementation of a correlation based attack in 5.1.

The other adverse feature is the most obvious obstacle. To save bandwidth, group messages are currently uploaded just once and distributed afterwards. Sending the message to each group member independently is intractable, especially for large groups. In the worst case, big files with characteristic sizes are immensely hard to hide.

The attacks above exploit the availability of the receiver identity. Due to transport encryption, an attacker is required to be inside the Signal server or to somehow else learn about this information. Besides a server administrator, there are other parties that could learn from the correlation mentioned above. Client action is initiated by a push message. These are provided by the operating system, so it is a Google service for android and delivered by Apple in the iOS version. Collecting metadata on the timing of notifications, both of these companies are able to construct a social graph for their phone users, even when explicitly opting out of uploading the address book into their respective cloud. Therefore, some bigger protocol changes are necessary.

This section continues with a description of the statistical attack on message timing mentioned above. Afterwards, two protocol improvements are discussed to reduce the risk of becoming a victim of this attack. The first makes information about recipients' identities unavailable to the server using pseudonyms. The other complicates timing attacks performed by central push notification services, but comes with the greatest increase in energy usage of all proposals in this thesis.

5.1 Statistical Attack

The guess that message timings can be exploited is substantiated by the following experiments. The attacker gets to know the points in time of message arrival for each user and tries to reconstruct the social graph. Figure 4 shows the performance of edge recognition. The above mentioned feature of acknowledgements is modeled here by using data on Tor latencies collected by [17]. After a message is received, the acknowledgement passes two Tor circuits to get back to the sender. This delay is sampled from a set of real world latencies [25].

The first thing to note about that attack is the social graph found by message timings is different from the one discussed before. By nature, it is not about address book entries, but users who actively converse with each other. Therefore, the simulation uses a model where friends regularly send messages to each other. An accuracy of 100% in the simulation means that all edges are guessed correctly with regards to the new definition of the social graph.

Some specifics about the simulation are also worth noting. One design choice is to make users indistinguishable from each other in their behavior. The model does not factor in, among others

- differences between time of day in user activity,
- varying message frequencies for chats or
- clients being offline.

Some of these could be exploited by an attacker, whereas less frequented chats certainly delay the correct answer. All experimental results should be regarded as a proof of concept, meaning that exact numbers on detection rates for a given time frame depend on parameters like the number of messages sent per user or the speed at which an answer is written. Identical user behavior should increase confidence in these experiments showing that an attack purely driven by message timings is possible and effective. One other effect is also visible in figure 4: The number of friends is binomially distributed, with varying average. The total message frequency

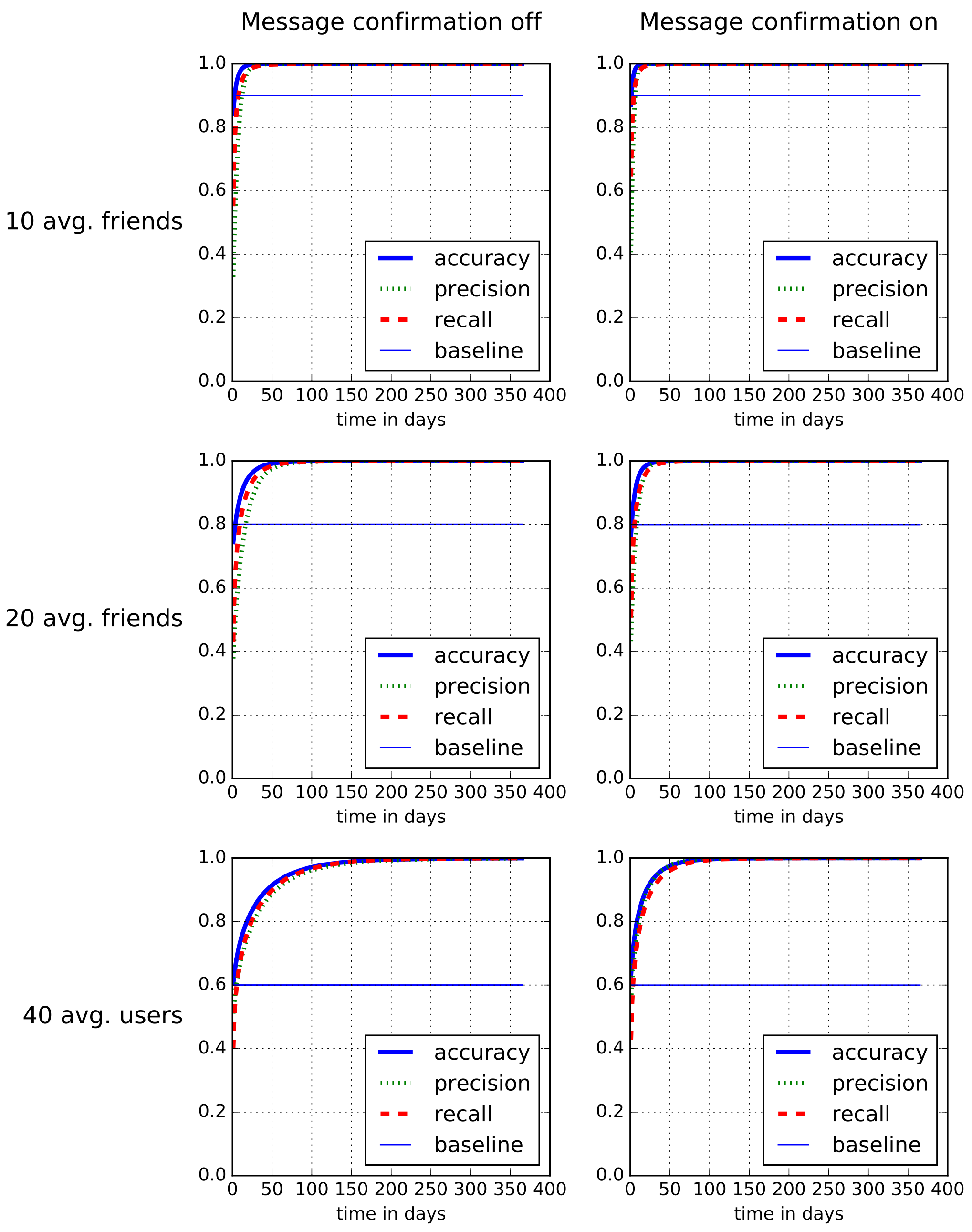

Figure 4: The graphs show the performance of the attack on 100 active members. From top to bottom, the average number of friends for each user increases. The left column has automatic delivery acknowledgements turned off, whereas the current implementation is shown on the right. The time is measured in days, starting with the attacker collecting data. The dotted blue line is the baseline accuracy of always predicting no connection. The solid blue line is the attacker's accuracy. Additionally, precision is given as a green dotted line and recall is dashed and red.

per user is kept constant across experiments, resulting in less messages per chat for bigger numbers of friends, delaying the attacker.

The attacker works as follows: Each time a message arrives, other users are checked for recent events that might be correlated. A Bayesian update on the probability of friendship between those users is then performed based on the existence and correlation of those events. Their likelihoods are estimated with training data on message timings.

The detection rate seems to converge for different numbers of node degrees in the social graph. Acknowledgements of messages generally speed the attack up. This still holds when not all users can be observed at the same time. This is the case for push notification services, between which the user base is split. Figures 6, 7 and 8 suggest that the attack still remains successful for different ratios of registered and observed users.

5.2 Pseudonymous Authentication

Similar to the objective of the previous section, the goal is to hide the social graph. Nonetheless, this section's protocol alterations can be used independently of contact discovery. Without losing features like group chat, this section will propose a set of changes that trades small concessions in usability for the chance to hide important conversations. This advances the state of the art because to this point, only the content, but not their existence was kept secret. For some user, the set of contacts is called $\mathcal{C}$ and the subset that is to be kept hidden C. The approach is optional and highly flexible, so a user can decide to

- not use it at all ($C = \emptyset$),
- only hide a subset of conversations ($\emptyset \neq C \subset \mathcal{C}$) or
- use it on all of his contacts ($C = \mathcal{C}$).

Previously, problems arose when third parties were able to collect information about the time a user receives a message. To remove this connection, neither the Signal server nor a push service should be able to identify the client it is delivering to, when the communication partner is in C. This is solved by introducing pseudonyms and using an open connection through Tor instead of the current push service.

The idea is that a client tells its contacts in C to use a new identifier to reply to. This identifier is a random string that will be called pseudonym from now on. A special message is added to the protocol to inform others about the connection between pseudonym and real user. Beforehand, to be able to receive messages, the new pseudonym needs to be introduced to the server. Let us call the new identity a. Establishing a includes getting a stamp t_{id}, generating a key pair (K_a^{pub}, K_a^{priv}) and anonymously sending a, t_{id} and K_a^{pub} to the Signal server. Therefore, a new Tor circuit is opened which is solely used for communication concerning a (see figure 5). In case of a (highly unlikely) report of a collision, a is simply chosen anew.

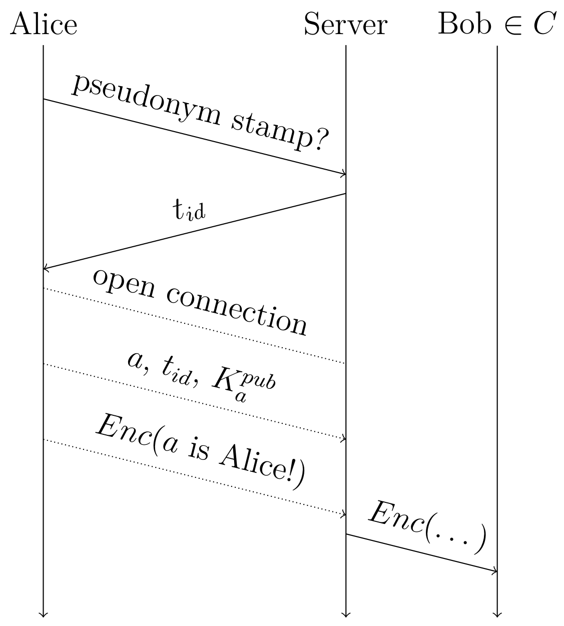

Figure 5: Alice introduces her pseudonym to the server and to Bob, one of her contacts. Consistent to the previous definition, all dotted lines indicate a connection using Tor. The message between server and Bob is also delivered through Tor if his pseudonym is already established in the other direction.

As mentioned in the discussion of different use cases for stamps, a new blind signing key pair is necessary to only allow a low rate of generating new pseudonyms, depending on the server's capabilities.

5.3 Multiple Pseudonyms

Changing the pseudonym requires a client to use both the new and the old one for a short period of time to not miss any messages on their way. For the same reason, a contact should resend everything that was not acknowledged to the new address.

The number of pseudonyms a client can use is limited only by the number of stamps and his willingness to use bandwidth and energy for keeping the push connections open through Tor. There is another reason why one might want to have several different names to receive messages with. Any malicious user could publish a known connection between account and pseudonym. If a user wants to talk to someone anonymously without trusting him, creating a new key pair for a critical conversation helps.

The timing of requests sent by a user and his pseudonyms might again be evidence for attributing them correctly. As a consequence, pseudonyms are not meant to be long-lived. For replacing them regularly, a new pseudonym can be sent to a contact. On a specific time equal for all clients (e.g. an exact weekday and time of day for weekly exchanges), everyone starts to send messages only to the most recently received pseudonyms. As a measure of precaution, a client should open and close its push connections at different points in time.

5.4 Push Service Replacement

The most inconvenient protocol change proposed in this thesis is replacing the push service. The current implementation is believed to be the most reliable and battery life preserving. The messages containing a pseudonym as the recipient can not be delivered the original way since the server can not assign it to any user account (as intended). The client now needs to keep a connection open as long as he wants to be notified of new messages from C, using K_a^{priv} for authentication.

To attain the new anonymity goal, using a separate Tor circuit for communication that belongs to a is necessary. Every other use of this pseudonym has to also use IP obfuscation. This applies to sending a message to someone in C as well as receiving them from C.

6 Evaluation

A brief discussion of possible weaknesses will outline the limitations of metadata leak prevention. It will start with an assessment of practical problems outside the model assumptions and continue with some attacks that are outside the scope of this thesis.

6.1 Always Online Assumption

The independence of samples in the Poisson point process generated for discovery requests depends on the ability to send queries according to schedule. In the model, all requests are uncorrelated and sent with the same rate, making their senders indistinguishable. Consider the other extreme: a setup where only one user is online at a time, immediately allowing the server to link all incoming requests to that account. This shows that user activity plays an important role in anonymity.

An attacker able to collect data over a long time period can gather enough statistical evidence to draw conclusions from these irregularities. To a persistent attacker, each user's anonymity set is effectively reduced to similarly behaving people.

6.2 Inactive Users

This particular vulnerability was mentioned in 4.4.2. An attacker inside the server maintains a list of all users that do not receive regular requests for discovery. Whenever one of them is targeted nonetheless, the most probable origin is a newly registered client, since the others are likely to already have their answer. A new entry in an old user's address book is the alternative.

A striking example is a user migrating to a new phone number. The server can relate his old and new contact details. It is unlikely that this old number is freshly entered in any address book, but someone could still register to Signal and try to find this user.

In conclusion, this attack only adds to the already existing false positive rate of the bloom filter. It leaks information about users currently unrelated to the Signal service. Note that only the server itself can exploit that weakness and in any case only gets evidence, no certainty. A high rate of user registration further reduces the individual probabilities.

6.3 Misbehaving Parties

There are multiple occasions where a protocol participant can harm anonymity without immediately being detected, especially for the server. The easiest to detect is using different signature keys for stamps. Users send each other their received public keys and compare. Differences outside of the periodic changes are definite proof of a malicious or compromised server.

The second opportunity for a server is to send bloom filters with increased false positive rate. Users would need to collect a lot of statistical evidence with exact knowledge of the target false positive rate to get suspicious. On the other hand, the attack itself only gives a chance to gain knowledge about specific edges of the social graph. An attacker could put any phone number into the bloom filter, forcing individual requests. A newly registered user could be confirmed to contact that number with high probability, using the timing side channel described in 4.4.2. Mass surveillance is only a bit more efficient, but this attack is suited more for targeted attacks.

Another party that could cause harm with its misbehavior is a contact leaking pseudonyms of other users. The similar attack of publishing one's own address book is impossible to prevent in general. Compared to using a new pseudonym for each contact, the harm done is equivalent. The number of pseudonyms reflects a user's need for protection against this specific attack.

6.4 Graph Similarity Attack

Another way to find connections between true identities and pseudonyms is to use an existing social graph from another application or network and matching it to the graph of pseudonyms. As it is pointed out in [16], graphs inherently complicate privacy guarantees because of the correlation of data points. One risk is to start the deanonymization with the help of dishonest participants, but also passive attacks are feasible [22]. Both of these problems are diminished by allowing multiple pseudonyms for a single individual, but can not be completely ruled out especially in a targeted attack.

6.5 Entry Guard Timing Attack

The practically most relevant attack is about breaking the anonymity of Tor. This usually needs the attacker to control the entry guard and the exit node of a Tor circuit [8]. Using the timing attack on messages described in 5.1, measuring arrival times suffices to learn about edges of the social graph. To sum up, the following requirements need to be fulfilled for successfully deducing a link between two users:

- Both connect to Tor through an entry guard controlled by the attacker long enough.
- The attacker can connect both users' IP addresses with their Signal account.
- The concerned users exchange messages regularly.
- The Signal messages can be distinguished from the remaining Tor traffic.

The Tor policy on entry guards guarantees that clients stay on the same entry guard for some time [1]. The advantage therein is that only a fraction of the user base can be spied on at the same time. This comes with a downside: Affected users will be under surveillance long enough to collect the amount of evidence needed to gain confidence about their interconnectedness. Real world data on the effectiveness of the timing attack could be used for further research on the best choice for entry guard rotation in this application.

A powerful attacker might put in a lot of effort to achieve all requirements. Even then, the protocol changes proposed so far should not be considered useless. Mass surveillance is still more difficult, since information is collected more slowly and only on a small part of the social graph. Then again, other messengers who claim to not leak metadata suffer from the same attack. Notable examples are Briar and TorChat. Both of these are working on a peer-to-peer basis, using Tor hidden services for message reception. This reflects a known weakness of the Tor protocol: Controlling the entry guard of a hidden service allows to deanonymize it easily [35].

7 Conclusion

The goal of this thesis was to keep a malicious server from finding information about a user's contacts. The decisions made were balancing usability and privacy in a way that fits most users, giving choices where possible.

A new approach to contact discovery was presented, using a stamp system for accounting. Implementing it as a mandatory feature protects all users that are not registered with Signal. If they are listed in someone's address book, the server does not learn who they are linked with anymore. Connections between active and inactive users are not necessarily hidden, but usability-wise, this feature is still cheap compared to its benefits.

Considering an attacker from within the Signal server, more privacy weaknesses were found. These are addressed in a proposal to allow for users to register pseudonyms. The reasoning is very close to the idea to let Signal users register with an email address instead of a phone number. The difference remains that users can still use their regular address books when using pseudonyms. Implementing it as an opt-in feature would serve people with high privacy demands immediately, while Open Whisper Systems evaluates its user acceptance to decide on switching to an opt-out policy. Therefore, deploying the protocol extensions only requires a software update. No additional infrastructure is necessary for the proposed extensions.

References

[1] The Tor Blog. Improving Tor's anonymity by changing guard parameters. `https://blog.torproject.org/improving-tors-anonymity-changing-guard-parameters`, 2013. Accessed October 17, 2017.

[2] Burton H. Bloom. Space/Time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, 1970.

[3] Nikita Borisov. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, pages 77–84. ACM Press, 2004.

[4] Nikita Borisov, George Danezis, and Ian Goldberg. DP5: A private presence service. *Proceedings on Privacy Enhancing Technologies*, pages 4–24, 2015.

[5] David Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum, 1983.

[6] David Cole. 'We Kill People Based on Metadata'. `http://www.nybooks.com/daily/2014/05/10/we-kill-people-based-metadata/`, 2014. Accessed October 17, 2017.

[7] Intel Corporation. Intel Software Guard Extensions Developer Guide. `https://download.01.org/intel-sgx/linux-1.9/docs/Intel_SGX_Developer_Guide.pdf`, 2017. Accessed October 17, 2017.

[8] George Danezis. The traffic analysis of continuous-time mixes. In *Proceedings of Privacy Enhancing Technologies workshop*, volume 3424 of *LNCS*, pages 35–50, May 2004.

[9] Google Developers. Cloud Messaging. `https://developers.google.com/cloud-messaging/`, 2016. Accessed October 17, 2017.

[10] 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.

[11] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, pages 1–27, 10 2016.

[12] Intel. Intel Software Guard Extensions. `https://software.intel.com/en-us/sgx/details`, 2015. Accessed October 17, 2017.

[13] J.F.C. Kingman. *Poisson Processes*. Oxford Studies in Probability. Clarendon Press, 1992.

[14] Stevens Le Blond, David Choffnes, William Caldwell, Peter Druschel, and Nicholas Merritt. Herd: A scalable, traffic analysis resistant anonymity network for VoIP systems. *SIGCOMM Comput. Commun. Rev.*, 45(4):639–652, 8 2015.

[15] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright. Timing attacks in low-latency mix systems. In *Proceedings of the 8th International Financial Cryptography Conference*, volume 3110, pages 251–265. Springer, 2004.

[16] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 93–106, New York, NY, USA, 2008.

[17] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. A case study on measuring statistical data in the Tor anonymity network. In *Proceedings of the Workshop on Ethics in Computer Security Research*, LNCS. Springer, January 2010.

[18] Moxie Marlinspike. The Difficulty Of Private Contact Discovery. `https://signal.org/blog/contact-discovery/`, 2014. Accessed October 17, 2017.

[19] Moxie Marlinspike. Technology preview: Private contact discovery for Signal. `https://signal.org/blog/private-contact-discovery/`, 2017. Accessed October 17, 2017.

[20] Matthew Miller. The 'Snowden Effect' Is Crushing US Tech Firms In China. `http://www.businessinsider.com/the-snowden-effect-is-crushing-us-tech-firms-in-china-2014-1?IR=T`, 2014. Accessed October 17, 2017.

[21] Nikolay A. Moldovyan. Blind signature protocols from digital signature standards. *International Journal of Network Security*, pages 22–30, 2011.

[22] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 173–187, Washington, DC, USA, 2009. IEEE Computer Society.

[23] Rahul Parhi, Michael Schliep, and Nicholas Hopper. MP3: A more efficient private presence protocol. *CoRR*, 2016.

[24] David Pointcheval. How to encrypt properly with RSA. *RSA Laboratories' CryptoBytes*, 5(1):9–19, 01 2002.

[25] The Tor Project. CollecTor. `https://collector.torproject.org/archive/torperf/`, 2017. Accessed October 17, 2017.

[26] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Malware guard extension: Using SGX to conceal cache attacks. *CoRR*, abs/1702.08719, 2017.

[27] Radu Sion and Bogdan Carbunar. On the computational practicality of private information retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium*, 2007.

[28] Open Whisper Systems. Signal Privacy Policy. `https://signal.org/signal/privacy/`, 2017. Accessed October 17, 2017.

[29] Raphael R. Toledo, George Danezis, and Ian Goldberg. Lower-cost epsilon-private information retrieval. *CoRR*, 2016.

[30] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 137–152, 2015.

[31] WhisperSystems. Advanced cryptographic ratcheting. `https://signal.org/blog/advanced-ratcheting/`, 2013. Accessed October 17, 2017.

[32] WhisperSystems. Free, Worldwide, Encrypted Phone Calls for iPhone. `https://signal.org/blog/signal/`, 2014. Accessed October 17, 2017.

[33] WhisperSystems. API Protocol. `https://github.com/WhisperSystems/TextSecure-Server/wiki/API-Protocol`, 2016. Accessed October 17, 2017.

[34] WhisperSystems. TextSecure-Server. `https://github.com/WhisperSystems/TextSecure-Server`, 2016. Accessed October 17, 2017.

[35] Lasse Øverlier. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 100–114, 2006.

Appendix

Proof for 4.4.2

For simplicity, assume a correct answer for ties in the maximum argument (leading to overestimating the true value). Let $q_k := \mathbb{P}(N = k) = \frac{\eta^k}{k!}e^{-\eta}$ be the probability of k events in a Poisson point process with rate η. The probability of B having the maximum number of requests then is the sum of all possible outcomes for B, with all clients C_i querying themselves at most once more.

$$p(n) := \sum_{k=0}^{\infty} q_k \left(\sum_{l=0}^{k+1} q_l \right)^n$$

It is left to show that for all $\varepsilon > 0$ there exists an $n \in \mathbb{N}$ such that $p(n) < \varepsilon$. Now let $\psi := \frac{\varepsilon}{2}$. Since $\lim\limits_{n\to\infty} q_k = 0$ and $\sum_{k=0}^{\infty} q_k = 1$, it is always possible to choose a k_0 such that $\sum_{k=k_0+1}^{\infty} q_k < \psi$. Using $q_k > 0$ and therefore $\sum_{k=0}^{k_0+1} q_k < 1$, there is always an n with $\left(\sum_{k=0}^{k_0+1} q_k\right)^n < \psi$. It follows:

$$\begin{aligned}
p(n) &= \sum_{k=0}^{k_0} q_k \left(\sum_{l=0}^{k+1} q_l \right)^n + \sum_{k=k_0+1}^{\infty} q_k \left(\sum_{l=0}^{k+1} q_l \right)^n \\
&< \sum_{k=0}^{k_0} q_k \cdot \psi + \sum_{k=k_0+1}^{\infty} q_k \cdot 1 \\
&< \psi \sum_{k=0}^{k_0} q_k + \psi < \psi + \psi < \varepsilon
\end{aligned}$$

Message Timing Attack Graphs for 5.1

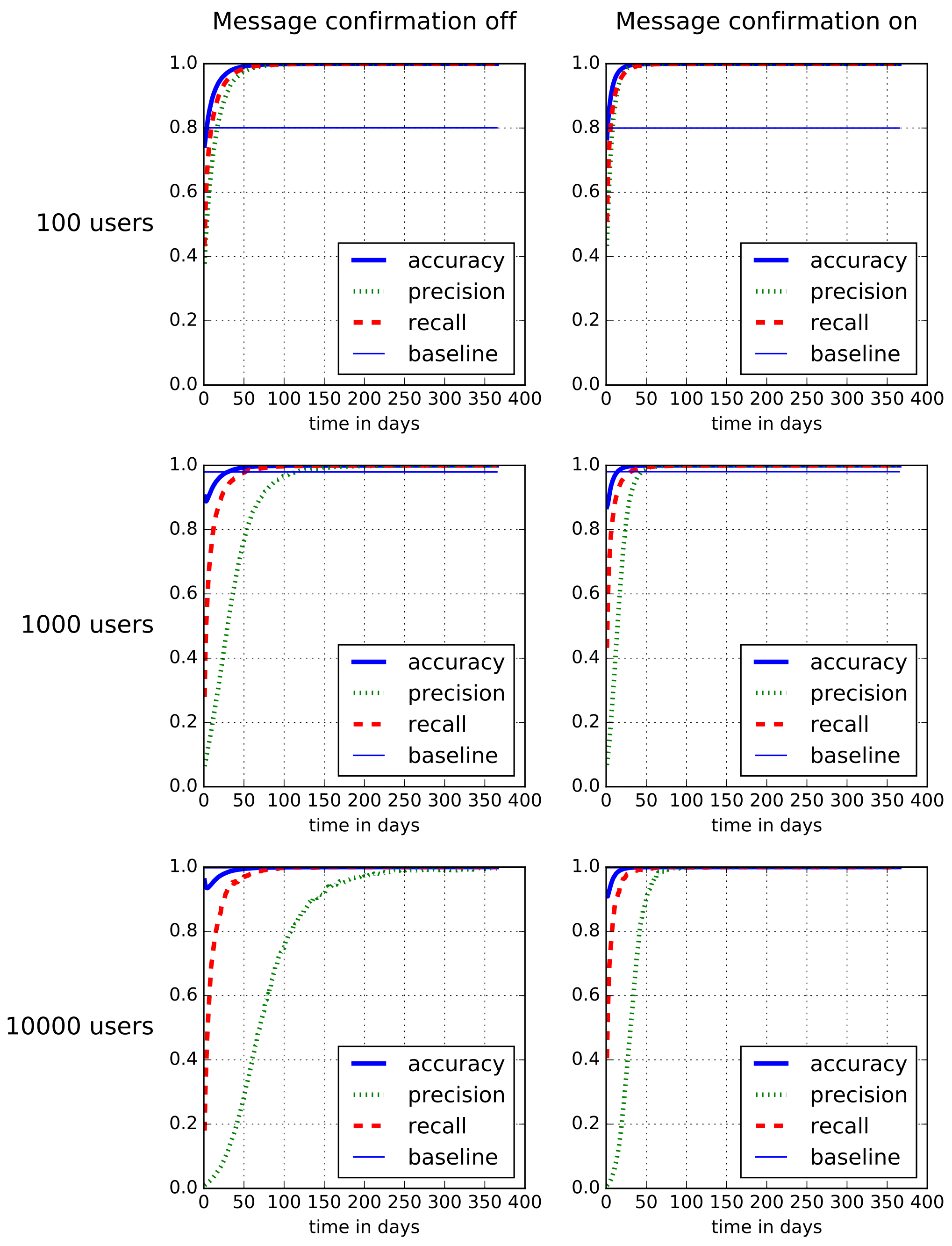

Figure 6: The graphs show the performance of the attack on 100 observed users with an average of 20 friends each. From top to bottom, the total number of active Signal users increases.

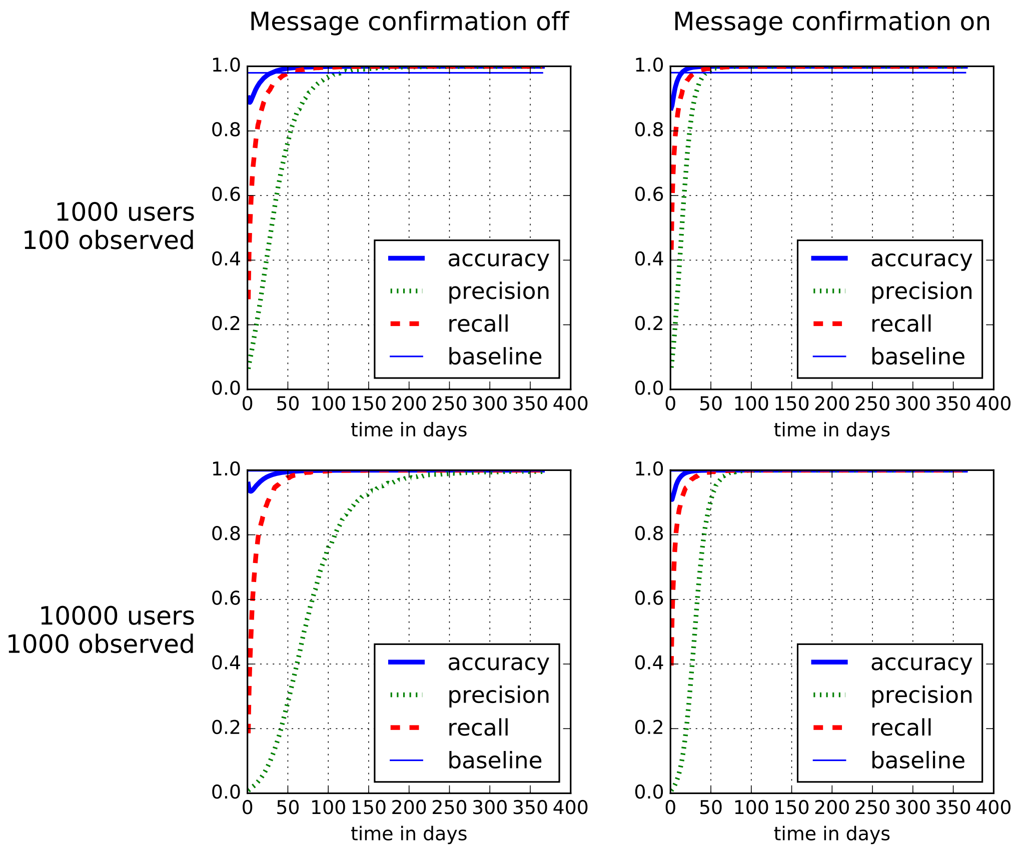

Figure 7: The graphs show the performance of the attack on a tenth of the members. Each user has an average of 20 friends each. From top to bottom, the total number of active Signal users increases.

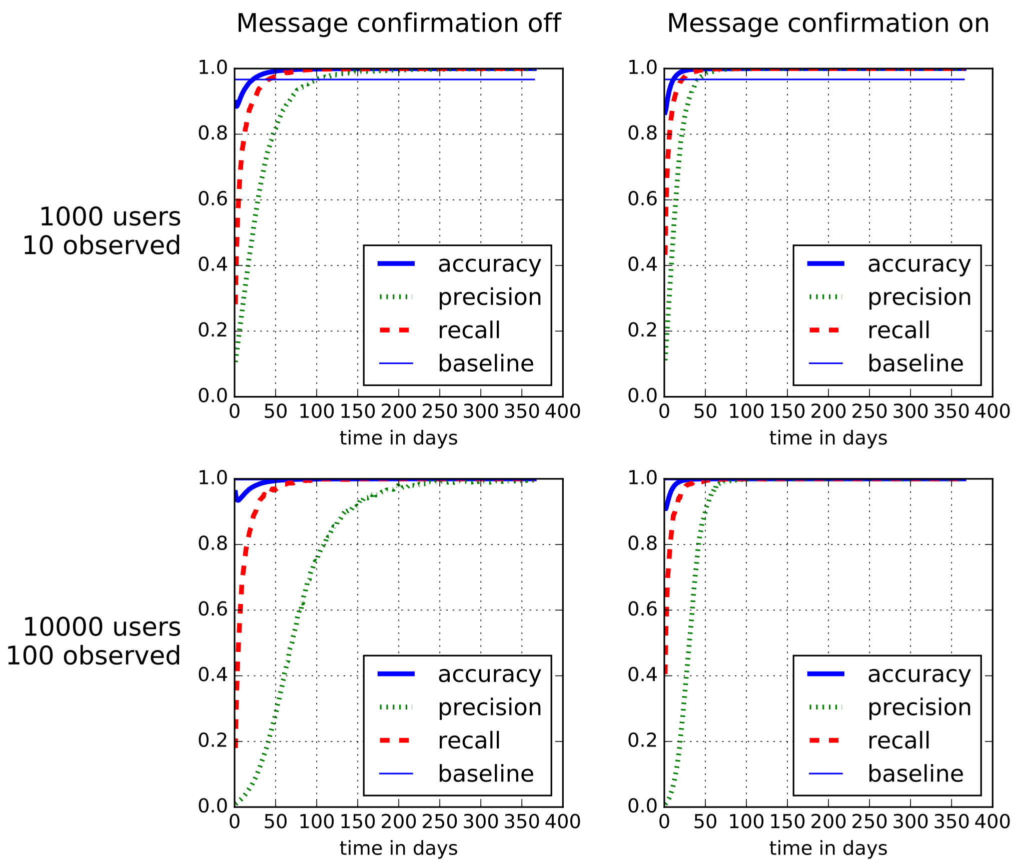

Figure 8: The graphs show the performance of the attack on one percent of the members. Each user has an an average of 20 friends each. From top to bottom, the total number of active Signal users increases.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 15. Dezember 2017 ....