Signal Protocol Analysis

April 2022

Contents

1 Introduction

The Signal protocol is a cryptographic messaging protocol developed by Open Whisper Systems in 2013. It provides end-to-end encryption for instant messaging in WhatsApp, Facebook Messenger, Skype and many others, being used by over 1 billion active users. The protocol was first introduced in the open-source TextSecure app, which later became Signal.

End-to-end encryption is designed to prevent anyone else from eavesdropping—with a good E2EE protocol, neither the application server, nor anyone with malicious intents are able to read the messages that are sent. This is because the keys used for encrypting the messages are generated by the participants of the conversation on their own devices, using their pairs of private keys, and no one else can compute those keys.

Besides protecting your messages from unauthorized access, the protection from the servers is also valuable, since not all providers are trustworthy. Some server providers are processing their users' messages for commercial purpose: tailoring ads for each specific user. Sometimes they are doing this as requested in trials in court. This cannot be done with end-to-end encryption: for example, WhatsApp was unable to comply with Brazilian government demands for users' plaintext messages because of its end-to-end encryption, and received a series of fines as a consequence.

But the feature that makes Signal Protocol stand out from similar protocols that provide end-to-end encryption is its implementation of "perfect-forward-secrecy". With most encryption systems, when an app is installed on a phone, it creates a permanent key pair that is used to encrypt and decrypt messages. If that private key is compromised, that potentially leaves all messages vulnerable to decryption.

The Signal protocol, however, uses a so-called "ratchet" system that changes the key after every message. It does this by generating a collection of temporary key pairs for each user, in addition to the permanent keys. When someone sends a message to a contact over an app using the Signal protocol, the app combines the temporary and permanent pairs of public and private keys for both users to create a shared secret key that's used to encrypt and decrypt that message.

For perfect forward secrecy to protect the messages to its full potential, the users need to delete their messages periodically, since they remain visible on the device containing the conversation, which might be a problem if someone's phone is seized or stolen.

2 Applications

2.1 Signal

Signal is a cross-platform instant messaging service based on the open source Signal protocol, widely considered the world's most secure end-to-end encrypted messaging app. It is owned by the non-profit Signal Foundation and Signal

Messenger LLC. Its great reputation for security and privacy is backed by high-profile endorsements from people like NSA whistleblower Edward Snowden, CEO of Twitter and Square Jack Dorsey, and WhatsApp founder Brian Acton, who left WhatsApp in 2018 to serve as the Signal Foundation's executive director.

According to their website, the application offers "an unexpected focus on privacy, combined with all of the features you expect". And that promise is held since the application offers the same functionalities as more popular messaging apps: users can send one-to-one and group messages, including files, voice notes, pictures and videos, and even make one-to-one and group voice and video calls with up to 40 people on iOS, Android, and desktop.

For additional protection in case a user's phone is lost or stolen, the Signal applications on Android and iOS can be locked with the phone's pin, passphrase, or biometric authentication, and the user can define a "screen lock timeout" interval.

Other functionalities include disappearing messages (over a specified time interval or after a single viewing), excluding users' messages from non-encrypted cloud backups by default and allowing users to automatically blur faces of people in photos to protect their identities. The blurring feature was added during the worldwide protests against racism and police violence sparked by the killing of George Floyd by law enforcement to help activists safely share images of the demonstrations without compromising the identities of the participants.

Android users who want to use Signal even for communicating with people who do not have the application installed can do so by making Signal the default SMS/MMS application. This allows them to send and receive unencrypted SMS messages in addition to the standard end-to-end encrypted Signal messages.

Some of these features would be easier to implement in less secure platforms, but implementing these within Signal's privacy constraints, including a lack of metadata that other applications do not promise–can require significant feats of security engineering, and in some cases actual new research in cryptography.

An example would be stickers, which for Signal required designing a system where every sticker "pack" is encrypted with a "pack key." That key is itself encrypted and shared from one user to another when someone wants to install new stickers on their phone, so that Signal's server can never see decrypted stickers or even identify the Signal user who created or sent them. According to an interview with Signal's founder, Moxie Marlinspike, for group messaging Signal partnered with Microsoft Research to invent a novel form of "anonymous credentials", which allow a server to store information about who belongs in a group, but without ever learning the members' identities. "It required coming up with some innovations in the world of cryptography," Marlinspike says. "And in the end, it's just invisible. It's just groups, and it works like we expect groups to work."

2.2 WhatsApp

WhatsApp, one of the most well-known instant messaging applications in the world, first adopted the Signal protocol in 2014 to end-to-end encrypt all messages between Android phones, in what Marlinspike mentioned was "the largest deployment of end-to-end encryption ever." WhatsApp switched it on by default for all its users in 2016.

Of course there was a transitioning process. Before all users updated to the latest version of the software for their platform, there was still some plaintext on the network. To make this transition as clear as possible, WhatsApp clients notified users when their chats became end-to-end encrypted. Once a client recognized a contact as being fully E2EE capable, it did not permit transmitting plaintext to that contact, even if that contact were to downgrade to a version of the software that is not fully E2EE capable. This prevents the server or a network attacker from being able to perform a downgrade attack.

WhatsApp's biggest encryption loophole is that when conversations are backed up to iCloud or Google Cloud, those chats are no longer encrypted, unless this feature is activated in the application's settings.

One important feature of WhatsApp is true multi-device message synchronization. Until recently, the smartphone app was considered the primary source of truth for E2EE, and was the only one associated with a unique identity key used to initiate all conversations and to establish secure end-to-end connections for each user. For using other companion devices, the users needed to connect to the smartphone app and mirror its content - meaning those devices depended on the connection to the smartphone, which needed to be nearby and have enough battery.

WhatsApp's new multi-device architecture attempts to mitigate these issues without compromising security. Now each has its own identity key, while users have a list of authorized devices. To authorize a new device, a user will need to scan a QR code from their phone, which can include the use of biometric data where enabled.

The fundamental approach that the Signal protocol adopts to ensure all messages are sent to all devices in encrypted form is called client-fanout. It consists in sending a message to each of the devices in both the sender's and receiver's device lists. Each message sent to a given device is encrypted with the key established between the sender and the receiver devices.

2.3 Google

Google is another tech giant that partnered with Open Whisper Systems for three of their apps: Google Allo, a now-defunct messaging app that was launched in 2016 and which featured an optional Incognito Mode with end-to-end encrypted communication, Google Duo, used for video chat services, and Google Messages, which replaced Allo. For Google Messages, the company announced that they would be using the Signal Protocol to provide end-to-end encryption by default to all RCS(Rich Communications Services)-based conversations be-

tween users of their app, starting with one-to-one conversations. Something to note here is that this works only when both participants in the conversation use RCS, for which both Chat Features for Androind and a WI-FI/data connection is also needed.

This decision represented the biggest new collection of phones to adopt the standard in years, with hundreds of millions more devices.

2.4 Skype

In January 2018, Open Whisper Systems and Microsoft announced the addition of Signal Protocol support to an optional Skype mode called Private Conversations for audio calls, text, and multimedia in one-to-one conversations. Even with Private Conversations turned on, Skype will still be able to access some metadata about communications, for example the time they occur and their duration.

2.5 Facebook Messenger

Facebook followed by adding it as an opt-in "Secret Conversations" feature in Facebook Messenger a few months later. The only way Facebook could access the plaintext messages is when a user reports an abuse, which triggers sending the message to Facebook so it can be verified.

The reporting mechanism is called "franking" and it must satisfy three main guarantees: authenticity, confidentiality and third-party deniability. The authenticity property ensures that if a user submits a report then the message must have legitimately originated from the sender's device. The confidentiality property ensures that no outside party — including Facebook — should learn the content of a message unless a participant in a secret conversation voluntarily shares that information. Finally, the third-party deniability property ensures that no party outside of Facebook can cryptographically determine the validity of a report.

Unlike WhatsApp, this feature is not enabled by default and the conversations are not available on multiple devices.

3 Protocol Analysis

3.1 End-to-end Encryption

End-to-end encryption is a system of communication where the only people who can read the messages are the people communicating. No eavesdropper can access the cryptographic keys needed to decrypt the conversation, not even a company that runs the messaging service. Many companies tell their users that their communications app is encrypted, but this claim leads to an important question: who has the key?. In most cases the company holds the cryptographic key used to encrypt your messages, witch means that anyone who gains access to the company's data can also gain access to you messages.

The increasing demands on privacy-concious users has led to the creation of a feature known as "end-to-end encryption". The "end-to-end" implies that the messages are encrypted is such a way that only the intended recipient can decrypt it. The cryptographic keys being known only to the end point computers while the company's server acts just as a messenger who is unable to decrypt the message.

3.2 History

Early instant messaging services did not provide much with regards to security. Some systems did provide traffic encriptyon between the user and service provider, however, the service provider could read the plaintext of users messages.

One of the first secure protocols for instant messages was Off-the-Record Messaging (OTR), a plugin for a variety of instant messaging applications. Using this plugin users could could authenticate each other using public keys or a shared secret passphrase, obtain end-to-end confidentiality and integrity. One important contribution of OTR was the ratcheting technique: along with each message round trip, users established a fresh ephemeral Diffie–Hellman shared secret. This technique became known as ratcheting because it was impossible to work backward from a later state to an earlier state and decrypt past messages. Off The Record did not see widespread adoption, but its ratcheting technique can be seen in modern security protocols.

The first secure instant messaging protocol which saw widespread adoption was apple's iMessage, a proprietary protocol which provided end-to-end ecryption. A notable characteristic of iMessage is it's automatic management of the distribution of users' long-term keys.

TextSecur was a secure messaging app and the predecessor to Signal. It contained the first implementation of Signal's "Double Ratchet", which effectively combines ideas from OTR's asymmetric ratchet and a symmetric ratchet. TextSecure's combined ratchet was referred to as the "Axolotl Ratchet". TextSecure was later merged with RedPhone, a secure telephony app, and was renamed Signal1, the name of both the instant messaging app and the cryptographic protocol.

The Signal cryptographic protocol has seen huge widespread in personal communications applications. It is now used by WhatsApp, Wire and Facebook Messenger, as well as a host of variants in different "secure messaging" apps.

3.3 Signal Protocol

The most proeminent development in the end-to-end encriptyon space has been the Signal messaging protocol, "a ratcheting forward secrecy protocol that works in synchronous and asynchronous messaging environments". Signal aims to implement end-to-end encryption and to use advanced security propertyes like perfect forward secrecy and "future secrecy".
The Signal protocol can be divided in the following stages:

- Initial key exchange, also known as extended triple Diffie-Hellman(X3DH) protocol, that combines long-term, medium-term and ephemeral Diffie-Hellman keys to produce a shared secret "root" value.
- Asymmetric ratchet stage, where users alternatively send new ephemeral keys using previously generated root keys to generate forward-secret chaining keys.
- Symmetric ratchet stage, where users use key derivation functions to ratchet forward chaining keys to create symmetric encryption keys.

Forward secrecy is provided by the fact that every new message sent by a user is encrypted using a fresh key, while the alternative exchange of new ephemeral Diffie-Hellman keys inject additionally entropy into this process, is intended to achieve perfect forward secrecy

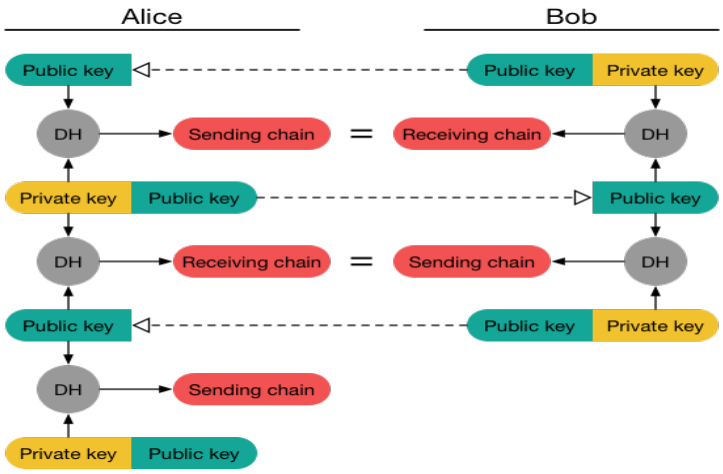

Figure 1: Simplified Signal protocol communication

At a high level, Signal is an asynchronous secure channel protocol, with keys computed by a multi-stage Authenticated Key Exchange (AKE) protocol, between an initiator Alice, a responded Bob, with the help of a key distribution server which only stores and relays information between parties, but does not perform any computation.

Signal assumes each party has a long-term public/private key pair, referred to as the identity key. However, since the parties might be offline at any point in time, standard AKE solutions cannot be directly applied.

Signal implements an asynchronous transmission protocol, by requiring potential recipients to pre-send batches of ephemeral public keys, during registration or later. When the sender wishes to send a message, she obtains keys for

the recipient from an intermediate server, which only acts as a buffer, and performs an AKE-like protocol using the long-term and ephemeral keys to compute a message encryption key.

This basic setup is then extended by making the message keys dependent on all previously performed exchanges between the parties, using a combination of "ratcheting" mechanisms to form "chains".

If we interpret Signal as a multi-stage authenticated key exchange protocol,we can define the main steps are as follows:

- **Registration**: At installation both Alice and Bob independently register their identity with a key distribution server and upload some long-term, medium-term, and ephemeral public keys. These keys are periodically updated.
- **Session setup**: Alice requests and receives a set of Bob's public keys from the key distribution server and use them to setup a long-lived messaging session and establish initial symmetric encryption keys. This process is called TripleDH handshake (X3DH).
- **Synchronous messaging** or asymmetric-ratchet: When Alice wants to send a message to Bob, and has just received a message from him, she exchanges Diffie–Hellman values with Bob, generating new shared secrets and uses them to begin new chains of message keys.
- **Asynchronous messaging** or symmetric-ratchet: When Alice wants to send a message to Bob (or vice versa) but has not received a message from Bob since her last sent message to him, she derives a new symmetric encryption key from her previous state using a pseudo random function family.

3.3.1 Registration

At installation and periodically afterwards, all users generate some cryptographic keys and register themselves with the key distribution server.

The Diffie Helman keys generated by a user are as follows: A long-term key, known as the "*identity*", a medium-term key called a "*signed prekey*" and multiple short-term "*one-time prekeys*". The public keys corresponding to the generated ones and a signature on the prekey using the identity are then uploaded to the server.

This collection is called a "pre-key bundle".

3.3.2 Setup Session

In this stage, public keys are exchanged and used to initialise shared secrets. The underlying key exchange protocol is a one-round DH protocol called the Signal Key Exchange or X3DH4, comprising an exchange of various DH public keys, computation of various DH shared secrets and then application of a key derivation function.

For asynchronicity Signal uses prekeys: initial protocol messages which are stored in the intermediate server, allowing users to establish a session with offline peers by retrieving one of their cached messages (in the form of a DH ephemeral public key). In adition tho the ephemeral keys users also publish a medium-term key. In the case that the ephemeral keys are exausted the medium-term key will be used instead.

The session setup in the Signal protocol has 2 stepts:

1. **Obtaining ephemeral** values, usually from a key distribution server.
2. Treating the received value as the first message and completing the exchange to **derive a master secret**.

Receiving Ephemeral prekeys. The most common way for Alice to obtain Bob's session specific data is to querry the key distributuion server for the pecomputed values of the pre-key bundle.

The server gives Alice bob's identity public key, his current signed prekey and a one-time prekey if there are any available. The signed prekey being used in the medium-term is shared with all users who send messags to bob, while one-time keys are shared with only one user and are deleted by the server after they are sent.

Alice's initial message contains identifiers for the prekeys, so that Bob knows witch ones were used.

Deriving Master Secrets. After receiving Bob's pre-key bundle Alice computes her own *ephemeralkey* and computes a session key. She then concatenates the resulting values and passes them through a key derivation function that derives an initial *rootkey* and sending *chainkey*.Finally, she generates a new ephemeral DH key, known as her *ratchetkey*.

For the key exchange to be complete Bob must receive Alice's public *ephemeral key* and public *ratchet key*. Alice attaches these values to all the messages she sends to Bob, until she receives a message from him, this way she can conclude that Bob has received the public keys.

After receiving the ephemeral and chain keys Bob first checks if he knows the private keys corresponding to the *pre-key bundle* used by Alice. If so he performs the receiving algorithm for the key exchange and derives the same *root* and *chain keys*.

3.3.3 Synchronous Messaging

Synchronous messaging represents a classical way of encrypted communication over a channel. In the case of the Signal Protocol, both parties will reply to each other messages at the moment it will be received. Therefore, for every message sent by one party, the *Send* ratchet of the sender and the *Rec* ratchet of the receiver will be modified accordingly, at the same time (synchronously) as you can see from Figure 2 below. There are two important aspects required to be true in order for the decryption and encryption of the messages to be valid.

The first one is represented by the convention that both parties start from the same root key. As such, if the root key is not the same, the double ratcheting algorithm will be applied, but we the decryption process will result in a different message than the one initially sent.

The second one is represented by the synchronous modifications applied to each party own ratchets correspondingly during a message transmission. When the ratchets are not in sync, then we are talking about asynchronous messaging.

3.3.4 Asynchronous Messaging

In the previous section, we discussed how the communication process works in a synchronous manner, that is when both parties are replying to each other messages in an instant. However, there are multiple scenarios when one of the parties is not available at the moment of sending the message from the other party (e.g. offline), thus the parties ratchets will be out of sync. This represents a certain type of communication called asynchronous messaging.

Intuitively, because the ratchets are out of sync, the encryption and decryption processes will not yield the expected results. In order to tackle this problem, it's the responsibility of the intermediary server to store a buffer with the already sent messages that were not received by a party. When the party will be available for receiving messages, the server will send the encrypted messages together with the appropriate public keys. Another problem here is the order of which the messages will be received from the server. It may be not in the order sent by the other party. The solution for those scenarios is to include the number of already sent messages together with the message. With this approach, the receiving party can check its own ratchet in order to compute the number of missing messages and establish the order of their transmission. The receiving party has the option of choosing to decrypt the current received message by advancing the ratchet with the required number of steps or wait for the other messages to arrive.

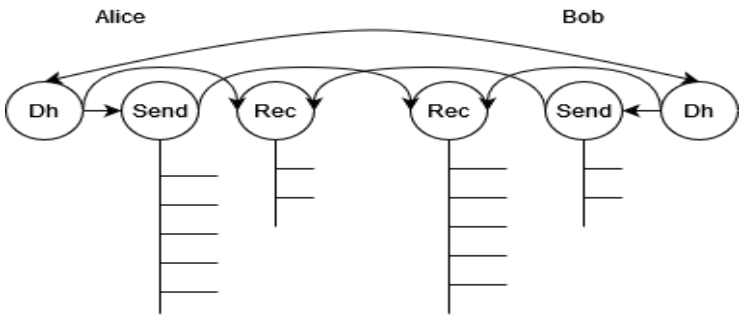

Figure 2: Alice & Bob instant messaging communication

3.3.5 Group messaging

So far, we analysed how the signal protocol works behind the scenes when the encrypted communication involves only two parties. Still, there are several real world cases when multiple parties are involved in the group communication. The naive extension of the signal protocol when approaching group messaging type of scenarios is represented by the creation of pairwise required signal keys between each individual party. As such, every advantages specific to the two party type of communication is retained in those group messaging scenarios.

The figure 3, below, represents a group type of messaging, where Alice, Bob and Charlie exchanged keys between each other in a pairwise form. Whenever Alice sends a message to the group, it sends, individually, to both Bob and Charlie using the specific public keys, and same goes for Bob and Charlie.

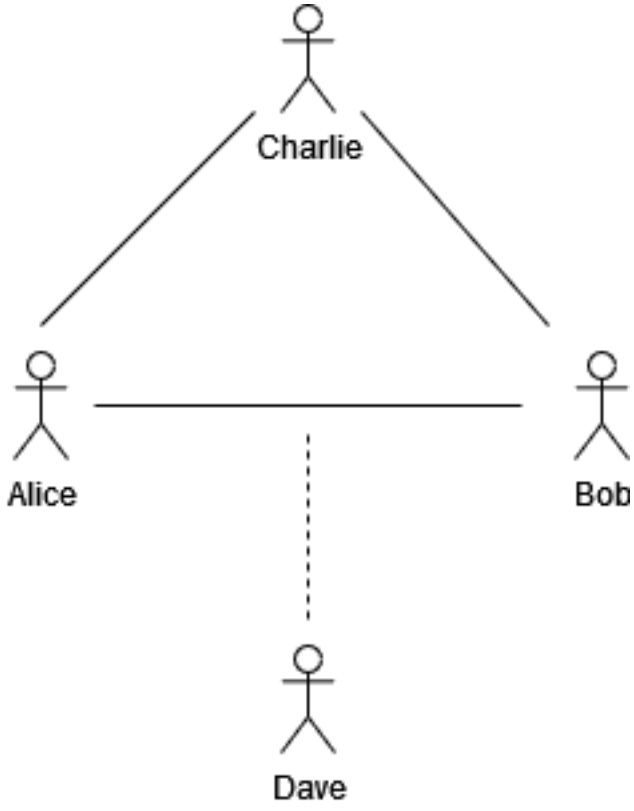

Figure 3: Group instant messaging

However, when a large number of parties are involved in the group messaging channel, the number of pairwise keys defined for each individual from the group grows significantly. This was unacceptable in the case of WhatsApp and Signal applications, because of the bandwidth large costs that resulted from this ap-

proach. Several optimisations were introduced in order to counter the large use of bandwidth, but the costs remain significant.

Multiple solutions were published, and one of them proposes the creation of sender keys. The senders keys are sent once from each party to the other members of the group and they will not be modified. With this approach, we can easily identify all parties from the group and we reduce the costs by a significant margin, but the Diffie-Hellman is no longer applied whenever a party sends a message. As such, we lose the ability of having any mechanism of enforcing post-compromise security if a key is cracked. Due to the nature of ratcheting process, we lose the future secrecy characteristic.

Currently, there is no perfect solution in respect of the security and performance characteristics. Signal has a limitation of 1000 members per group while WhatsApp has 256 members limitation. Those numbers will certainly increase in the future because of the ever increasing computational power for our devices.

3.3.6 Security Vulnerabilities - UKS

While complex in it's mathematical nature, exploits and vulnerabilities still exists for the Signal Protocol. One popular attack is the Unknown Key-Share (UKS). The way the USK attack works is not by breaking the keys using computational power and read messages sent by one or both parties. It is an identity masking type of attack.

For instance, the party P_s (sender) will pass a message to the party P_a (attacker). The attacker will lie about its public identity key, and, instead, will respond with the public identity key of party P_v (victim). Therefore, P_s will have the illusion that the message intended for P_a was sent to that party. Instead, P_v will receive the message.

There are multiple solutions to verify the identity of the receiver and the sender. In the case for WhatsApp and Signal, security codes are created based on the identity public key shared between each party, but those can be checked only out-of-band.

4 Conclusions

Signal protocol became an industry standard for end-to-end encryption, which is something that all instant messaging applications should adopt to protect their users' privacy. Since it is an open-source protocol and it was also adopted by many tech giants, it has been heavily scrutinized throughout the years and no major flaws were found in the design.

Nowadays, the application of the Signal Protocol for group messaging scenarios is the most researched one. A lot of effort is involved in order to create the best approach for the aforementioned scenarios regarding security and also performance results.

References

[1] Wikipedia

[2] A Formal Security Analysis of the Signal Messaging Protocol

[3] WhatsApp Adopts the Signal Protocol for Secure Multi-Device Communication

[4] Hacker Lexicon: What Is the Signal Encryption Protocol?

[5] Signal Is Finally Bringing Its Secure Messaging to the Masses

[6] Demystifying the Signal Protocol for End-to-End Encryption (E2EE)

[7] The battle inside Signal

[8] Signal announces new face-blurring tool for Android and iOS

[9] Signal official website

[10] WhatsApp Fixes Its Biggest Encryption Loophole

[11] WhatsApp launches multi-device beta with support for end to end encryption

[12] Messenger Secret Conversations Technical Whitepaper

[13] Signal Private Group Messaging