WhatsApp Security Paper Analysis

Calvin Li, Daniel Sanchez, Sean Hua

1 Introduction

WhatsApp, since its inception six years ago, has quickly grown into a global phenomenon, becoming one of the most popular mobile-based communications applications in the world today. With a user base that eclipsed one billion in February [1], WhatsApp provides a service that potentially endangers the privacy of over 10% of the entire human population. In order to address these security concerns, it was announced in early April that the application now offers full *end-to-end encryption* (E2EE) [2], meaning all messages, calls, and files, both in a one-on-one and group setting, are completely secure from hackers or even WhatsApp itself.

In addition to this announcement, WhatsApp released a technical white paper [3] detailing the newly incorporated security protocol, from session establishment to actual message encryption, all to support its claim of achieving E2EE. We are here to examine the low-level technical features of each component of the proposed security scheme as well as the algorithms that were utilized, ultimately analyzing if the system as a whole does indeed provide the level of security that it takes credit for.

Lastly, we want to critique the technical white paper itself, specifically on its clarity and thoroughness regarding potential threats and how the E2EE scheme protects against them.

2 Security Protocol Overview

2.1 Establishing Sessions

Establishing an encrypted session is vital in the Signal Protocol, not just because it allows two parties to get in touch and communicate, but also because the details of the session setup allow for the communication between the parties to remain secure.

2.1.1 Sessions Terminology

These are the three keys that are used to establish a session between two users:

`Identity Key`
A long term Curve25519 key pair, generated at install time.
Purpose: Signing the `Signed Pre Keys`, used in creating `master_secret`

`Signed Pre Key`
A medium-term Curve25519 key pair, generated at install time, signed by the `Identity Key`, and rotated on a periodic timed basis.
Purpose: Used in creating `master_secret`

`One-Time Pre Key`
A one time use key pair obtained from a queue of Curve25519 key pairs. Generated at install time, and replenished as needed.
Purpose: Used in creating `master_secret` when available

Here are some general terms that will be used later in the paper:

Initiator
Starts the session establishment process.

Recipient
The other user with whom the *initiator* is trying to establish a session with.

EDCH
Elliptic Curve Diffie Helman, an anonymous key agreement protocol that allows two parties, each having an elliptic curve public-private key pair, to establish a shared secret over an insecure channel.

HKDF
HMAC-based Extract-and-Expand Key Derivation Function. Key derivation function used to derive `Root Key` and `Chain Key` (see section **2.3.1**) from `master_secret`

2.1.2 Session Setup Process

To establish a session:

1. Initiator requests `Identity Key`, `Signed Pre Key`, and `One-Time Pre Key` from recipient
2. Server returns the requested values. If the queue of `One-Time Pre Keys` is empty, no `One-Time Pre Key` is returned.
3. Initiator saves these values as I_r , S_r , O_r
4. Initiator generates ephemeral Curve25519 key pair E_i

5. Initiator loads its own `Identity Key` as I_i
6. Initiator calculates
   `master_secret` = ECDH(I_i , S_r) | | ECDH(E_i , I_r) | | ECDH(E_i , S_r) | | ECDH(E_i , O_r)
7. Initiator uses HKDF to create `Root Key` and `Chain Keys` from `master_secret`

2.2 Receiving Sessions

After the initial session setup process, until the recipient responds, all information necessary to build a corresponding session is sent in the header of all messages from the initiator (E_i , I_i). The steps to create a corresponding session are then:

1. Recipient calculates the corresponding `master_secret` using it's own keys and the public keys advertised in the header of the message.
2. The recipient deletes the `One-Time Pre Key` used by the initiator.
3. Initiator uses HKDF to derive a corresponding `Root Key` and `Chain Keys` from `master_secret`

2.3 User-to-User Messaging Protocol

After establishing sessions, users can securely send messages to each other using a double-ratchet implementation to provide a unique one-time key for every individual message sent. This system claims that these ephemeral keys cannot be compromised even if other similar keys are.

2.3.1 Messaging Terminology

There are three keys that are used for encrypting messages sent between two users:

`Root Key`
32-byte key derived from `master_secret`
Purpose: Deriving `Chain Keys`

`Chain Key`
32-byte key derived from `Root Key`
Purpose: Deriving `Message Keys`

`Message Key`
80-byte key derived from `Chain Key`
32-byte for AES-256 key (message encryption)
32-byte for HMAC-SHA256 key (message authentication)
16-byte for IV (random initialization)
Purpose: Encrypting and authenticating a message (one-time use)

2.3.2 Double-Ratchet Algorithm

The Double-Ratchet Algorithm gets its name from its "ratcheting" mechanism on the `Chain Key` to get new `Chain Keys`, occurring when a user sends a message and when a user receives a message afterwards. When a user sends a message, it hashes the `Chain Key` to get the next `Chain Key`. Afterwards, when the user receives a message, it will also receive an ephemeral Curve25519 key used for "ratcheting" the `Chain Key` and `Root Key` forward.

The first ratchet phase is referred to as the **Hash Ratchet**, since the `Chain Key` is hashed with HMAC-SHA256 to get the new `Chain Key`. In this phase, the `Chain Key` is used to derive the one-time use `Message Key` by the following:

```
Message Key = HMAC-SHA256(Chain Key, 0x01)
```

Subsequently, the Chain Key is ratcheted forward by the following:

```
Chain Key = HMAC-SHA256(Chain Key, 0x02)
```

The second phase, referred to as the **DH Ratchet** (Diffie-Hellman), occurs when a user receives the message. In addition to receiving the encrypted message, it also sends a public ephemeral Curve25519 key. Using that ephemeral key along with its own (which the user also sends), they both calculate the `Chain Key` and `Root Key` as follows:

$$\texttt{ephemeral\_secret} = \texttt{ECDH}(\texttt{Ephemeral}_{\texttt{sender}}, \texttt{Ephemeral}_{\texttt{recepient}})$$
$$\texttt{Chain Key, Root Key} = \texttt{HKDF}(\texttt{Root Key, ephemeral\_secret})$$

Afterwards, it can use these new `Chain Keys` create more one-time use `Message Keys` to encrypt future messages.

2.4 Group Messaging Protocol

For the group messaging protocol, WhatsApp uses similar "ratchet" schemes to provide security to the messages to the group. For any group, each user has a pairwise encryption scheme by having a `Sender Key` from all users in the group, used for generating `Message Keys` for encryption. Whenever a group member decides to leave, they will delete the `Sender Key` for that given member. The encrypted messages are sent to the WhatsApp server, which will then fan-out the messages *N* times for the *N* other group members.

2.4.1 Establishing Sender Key

When a new user joins a group and sends its first message, it will first generate a random 32-byte `Chain Key` and Curve25519 `Signature Key` key pair. Then, the `Chain Key` and public

`Signature Key` will be combined together to create the `Sender Key`. This `Sender Key` will be distributed among the *N* users group using the same user-to-user protocol as mentioned above.

2.4.2 Sending Messages in Group Chat

The `Chain Key` will create the `Message Key` to then encrypt the message. Following, it will ratchet forward and update the `Chain Key`. The encrypted message will then be signed by the sender using the `Signature Key` from the `Sender Key`, and the ciphertext will be sent to the server to fan-out to the other members in the group.

3 Paper Analysis

Overall, we think the paper does a great job of describing the protocol used for WhatsApp encryption. They provide details about the encryption schemes for establishing sessions and creating ciphertext, along with some slight justifications on why their implementation ensures the confidentiality and integrity of their users' messages. However, I think there are several issues that their paper does not explicitly outline, some of which are crucial to protecting those claims.

3.1 Lack of Threat Model

One major flaw of the paper is the lack of a threat model presented to the reader. A threat model provides an overview of what readers can expect an adversary to do to negate the confidentiality and integrity of the messages being sent. Without the paper explicitly mentioning one, the reader cannot be certain how safe the protocol actually is. Perhaps, with a simple adversary who only eavesdrops into the network, there may not be any security flaws. However, with a more aggressive adversary who may try to pose as an imposter (e.g. Man In the Middle Attack) and intercept messages, WhatsApp protocol may not be able to provide that same security.

Granted, the paper does mention occasionally how components of their protocol provide security against certain types of adversaries. For instance, they claim the Double Ratchet Algorithm for user-to-user messaging prevents even WhatsApp from deciphering the messages. However, we believe that the threat model should be the foundation of the paper, and their descriptions and analysis of the protocol should be dependent on this threat model.

4 Security Analysis

4.1 Establishing a Threat Model

For our threat model, we believe that the adversaries can be aggressive and want to break the protocol by either (1) decrypting the ciphertext of many messages or (2) impostering as another user. We are focusing on the confidentiality and integrity of WhatsApp security protocol and want to see if any adversary, including WhatsApp, can interfere with the messaging between users. We assume that all attacks can be done in computationally polynomial time.

4.2 Security of Sessions

Establishing the sessions is the basis of the security of the messaging protocol, since the `Root Key` and `Chain Keys` are derived from `master_secret`, and `master_secret` can only be derived by using the corresponding session keys of the two users communicating. While the `Identity Key`, `Signed Pre Key`, and `One-Time Pre Key` are public, the ephemeral key pair E_i is generated when `master_secret` is generated, and is only passed into the ECDH function, and never sent across the channel by itself. Without this key, it would be very difficult to recreate `master_secret`. In addition, the `Signed Pre Key` is rotated on a periodic basis, and the `One-Time Pre Key` is removed from the server after use. As such, these values would be very difficult to guess after they have already been used.

4.3 Confidentiality of Messaging Protocol

For the security of the Double-Ratchet Protocol, they argue that it provides forward secrecy and "future secrecy". Both of these properties ensures the confidentiality of encrypted messages, even if one of the keys are compromised.

Forward secrecy is the property that when an adversary compromises a key, he/she cannot compromise previous keys. This property is especially important with WhatsApp protocol, because although the Message Keys are ephemeral, if compromising one can result into compromising many previous ones, an adversary can gain access to all previous Message Keys and decrypt previously sent messages. The Double-Ratchet Protocol handles this scenario with both the ratcheting actions. The Hash Ratchet changes the Chain Key with a hash function, and because of properties of hashing, knowing the new Chain Key provides no extra information about the old Chain Key. In addition, by updating the Chain Key again with the DH Ratchet makes it also untraceable to the old Chain Key and Root Key.

"Future secrecy", a term coined up by Open Whisper Systems, refers to the property that when an adversary compromises a key, he/she cannot compromise future keys. This property also applies to WhatsApp protocol, since with just a Hash Ratchet, it is fairly simple for an adversary to perform the ratchet step to get the future keys. As a result, compromising one key will lead to

compromising all future keys, which would also be problematic for the confidentiality of a user's messages. However, because of the algorithm's DH Ratchet, which uses ephemeral keys that cannot be rederived, the future Chain Keys are unretrievable given a current Chain Key. An adversary would have to know the shared ephemeral secret (see section **2.3.2**) to calculate future Chain Keys.

4.4 Integrity of Overall Sessions and Messaging Protocol

In terms of providing integrity of the messages received, WhatsApp provides a method for users to verify the authenticity of a session between a user by scanning a QR code. If a user scans the QR code of another, it verifies that the `Identity Key` one user has is indeed the correct `Identity Key` of that user. This procedure provides a measure for exposing the existence of a Man In the Middle (MITM) attack.

However, the user must actively look to verify the authenticity of a session in order to guarantee it. This method does not provide a preventive measure of stopping a MITM attack. As a result, given our threat model, a user cannot guarantee the integrity of the messages received. An adversary can pose as the middle man for a session, and then decide to end the session. After the session is recreated, the user will never realize that an adversary was intercepting all of the messages.

As a result, WhatsApp can further the security protocol by including a preventable measure for MITM attacks. This is an example of how users often can trust the session they make at the beginning, also known as Trust On First Use (TOFU). Instead of trusting that the session made is indeed the person, there should be a verification step to confirm that it is indeed the correct user someone is trying to communicate with.

5 Future Work

5.1 Code Analysis

A big contributor of WhatsApp's actual implementation of the security protocol is available online as open-source via Open Whisper Systems. For a more holistic analysis of the security implications of the WhatsApp's proposed E2EE, we'd like to take a close look at the code and examine if session initialization and message encryption do indeed incorporate the techniques stated in the paper. This will also give us the opportunity to discover any potential vulnerabilities that the paper might have missed.

6 Conclusion

WhatsApp is now one of the world's biggest platforms for online communication, and at a time when security risks are scrutinized more than ever, possibly exposing a user's private messages or files to a hacker or cybercriminal would be catastrophic. In response to these concerns, WhatsApp announced in April that the application has incorporated full end-to-end encryption, offering protection against any potential malicious entities, including itself.

After analyzing the technical white paper that WhatsApp released in conjunction with its big security announcement, we have come to a confident conclusion that it is impossible for any adversary to, in computationally polynomial time, intercept an encrypted message and subsequently decrypt it. A large factor in providing this security layer comes from WhatsApp's clever use of ephemeral and dynamic keys, with special recognition to the Double Ratchet algorithm that guarantees forward and future secrecy, so that even if a Chain Key is compromised, it is impossible to calculate past or future ones.

However, one potential point of concern is WhatsApp's session establishment. After a session is first initialized between two users, the same session is used for all subsequent interactions between the two, barring an external change like app uninstallation/reinstallation or a device change. This leaves the system vulnerable to Man In The Middle attacks, which are extremely dangerous. WhatsApp does offer a Key Verification feature to make sure that the person on the other end of the chat is who you think he/she is, but because this is not mandatory, users will be put at risk if they simply reply on the Trust On First Use principle that is inherent in session establishment. We propose to simply add a verification step before session creation to confirm that the connection is authentic.

7 References

[1] **Statt, Nick. "WhatsApp has grown to 1 billion users". The Verge. <http://www.theverge.com/2016/2/1/10889534/whats-app-1-billion-users-facebook-mark-zuckerberg>**

[2] **Koum, Jan. "end-to-end encryption" WhatsApp Blog. <https://blog.whatsapp.com/10000618/end-to-end-encryption>**

[3] **"WhatsApp Encryption Overview: Technical White Paper". WhatsApp. <https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf>**