WhatsApp End-to-End Encryption: Are Our Messages Private?

Research project by students of the SnE masters programme

Tom Carpay and Pavlos Lontorfos
Supervisors: Ruben De Vries and Soufiane el Aissaoui

February 5, 2019

As privacy becomes more important, so does private and secure communication. In this research we look at the End-to-End encryption of WhatsApp. Since the End-to-End encryption of WhatsApp is based on the Signal protocol, we analyze the implementation WhatsApp has chosen for its application. Fortified by a formal proof of its End-to-End encryption and security, we assume the Signal protocol to be secure and to be implemented correctly in the Signal application. We give a detailed description of the End-to-End algorithms used by the Signal protocol and design a set of experiments to compare traffic from both applications, attempt to decrypt this traffic, and formally analyze distinguishing features of WhatsApp.
The results from these experiments prove that the WhatsApp implementation of the Signal protocol diverges from the Signal application implementation. We believe this divergence is significant enough to invoke further research into WhatsApp.

1 Introduction

Facebook, the company that owns WhatsApp, has recently come under fire for a series of privacy exposure accusations [1] [2] [3]. In the acquisition of WhatsApp, Facebook promised that WhatsApp and Facebook would not share data. This promise is further bolstered by the switch to End-to-End encrypted communication by WhatsApp in 2016 [4]. Facebook still claims that it holds to this promise, but at this time there exists no scientific research verifying this claim.

In February 2016, WhatsApp has reported a billion active users, and peaked at 1.5 billion monthly active users in 2018 [5]. There have been multiple reports of WhatsApp being used by businesses, for example secure communications in medical teams [6] [7]. Given such a large and active community, a breach in either privacy or security of the WhatsApp message exchange protocol is likely to cause a major impact on day to day communication for both consumers and businesses.

Since 2016, WhatsApp has implemented the Signal protocol in order to guarantee End-to-End encryption for message exchange and forward secrecy. Signal is a communication protocol developed by Open Whisper Systems. The protocol was developed for the Signal application, which is their own chat application. WhatsApp reports that they use the Signal protocol as basis for the End-to-End encrypted communication in the app and discusses this in a published white paper [8]. This integration was also reported by Signal/Open Whisper Systems [9].
While the basis of End-to-End encrypted communication is the same for both WhatsApp and the Signal application, there exist deviations in the implementations between the two applications. These deviations will be discussed in depth in section 3. Since modifying a secure algorithm could decrease the strength of the security or even make the algorithm insecure, we aim to find the depth of the implementation deviations. We reason that communication can not be End-to-End encrypted if it is not secure.
Since WhatsApp switched protocol implementation, there have been several reports of attacks on the WhatsApp communication implementation [10] [11]. These reports give basis to further security research of WhatsApp.

The main question for this research will be: Is user-to-user message exchange via WhatsApp End-to-End encrypted?
To answer our main question, we firstly need to answer the following subquestions:

1. What are the algorithms used to create the Signal protocol?
2. To what extent are WhatsApp messages adhering to the Signal protocol specifications?
3. What are the differences between Signal and WhatsApp network traffic?

We approach this research with both a theoretical approach and a set of experiments, from which we can deduce the change in protocol, and ultimately, the End-to-End encryption.

The outline of this research is as follows: In section 2 we discus relevant research and in section3 we will provide a theoretical background of the algorithms that the Signal protocol is based on. Section 4 describes our assumptions, the blocking/non-blocking behavior, and the tools used. The different experiments done are described in section 5 and the results are discussed in section 6. Section 7 and section 9 provide insight into the research and the results and possible improvements, respectively. Finally, concluding remarks are given in 8.

2 Literature review

In 2016, Marlinspike and Perrin described the End-to-End encryption algorithms for the Signal protocol in two papers [12] [13], which both are discussed in section 3. The Extended Triple Diffie-Hellman describes a method for two parties to create common secret with the help of a third party key storage server. The Double Ratchet algorithm describes a method of End-to-End encrypted communication with the property of forward secrecy. Both these papers are the formal basis of our understanding of both the Signal protocol and application, and the WhatsApp application.
Cohn-Gordon et al. did extensive research into the implementation of Signal protocol [14] in 2017. In this research a formal proof is given for the security in communication of the Signal protocol, if implemented correctly. We use the results of this paper as the premise of the comparison of WhatsApp and Signal, and to make conclusions about our performed experiments.

In the published white paper WhatsApp reports to have used the Signal protocol as a basis for the End-to-End encryption for the application [8]. The white paper discusses the implementation for WhatsApp only, not the changes that were made from the Signal protocol base. This white paper is the main source of information used regarding the protocol used by WhatsApp for this research.

From the research done by Dai et al. [15], we learn that, while decompiling WhatsApp source is possible, the decompiled code is heavily obfuscated. While it could contribute, during this research we will not look at source code, decompiled or not, of both WhatsApp or the Signal application. The reasoning here is that due to the nature of this research project, we have a limited time frame, which we feel could be used more beneficially by looking at both a theoretical background and practical experiments.

3 Background

The theoretical foundation is heavily reliant on the work done by Cohn-Gordon et al.[14] and specifications given by Open Whisper Systems [12][13]. As discussed by Cohn-Gordon et al., the main improvement over other messaging protocols is the use of Extended Triple Diffie-Hellman, or X3DH, and the Double Ratchet system. Both are explained below.

3.1 Signal protocol

Extended Triple Diffie-Hellman
Open Whisper Systems specifies a key agreement protocol, where the system on

the other side can be offline [12], called Extended Triple Diffie-Hellman (X3DH). The basis of this protocol is to create a *shared secret* between two parties, where either party may not be available to send keys for a classic Diffie-Hellman exchange. This availability problem is solved by storing pre-created keys with a trusted third party, which in this case is a an authenticated key-exchange (AKE) server.
Every potential receiver, here called B, is required to publish three different keys to the AKE server. These keys consist of:

- an *identity key*
- a *signed prekey*
- a batch of *ephemeral one-time prekeys*

The *identity key* is generated at application install, device specific, and is never changed. The *signed prekey* is generated at install and changed periodically. The server should always have a sufficient amount of *ephemeral one-time prekeys* from B and replenish if necessary, and these should be destroyed every time a new sender, here called A, requests one.
The beginning of a session, a long term message exchange between two participants, and further messages exchange starts as follows. A requests the set of keys belonging to B and uses these keys to calculate a master secret. The master secret is derived from three different Diffie-Hellman calculations, using a combination of the keys belonging to B, and the *identity key* and an *one-time ephemeral key* of A. The combination of the three Diffie-Hellman calculation outcomes is defined as the *master secret*. As soon as A calculates and sends the keys, A deletes the both the *one-time ephemeral* key from both itself and B one to ensure forward secrecy.
To complete the session setup with B, A sends a message containing his own *identity key*, the ephemeral key he used, identifiers which of B's prekeys the sender used, and an initial message encrypted with the *master secret*. Upon receiving, B then uses A's keys to do the X3DH calculations and find the *master secret* key.

Double Ratchet
Once the session setup is complete, A and B have a common *master secret*, from which they both derive three keys, which are explained below:

- a *root key*
- a *receiving chain key*
- a *sending chain key*

Open Whisper Systems introduced a novel method of encrypted communication based on a shared secret: The Double Ratchet algorithm [13]. They state the algorithm offers resilience against adversaries, forward security, and break-in recovery. One of the bases of the algorithm lies in the Key derivation function (KDF) chains. Open Whisper Systems defines this as a function used in cryptography as that takes a KDF key and input data, and returns output data, which is used to generate another KDF key and an output key. From the first message on, every message that is sent, from either A to B, or B to A, contains

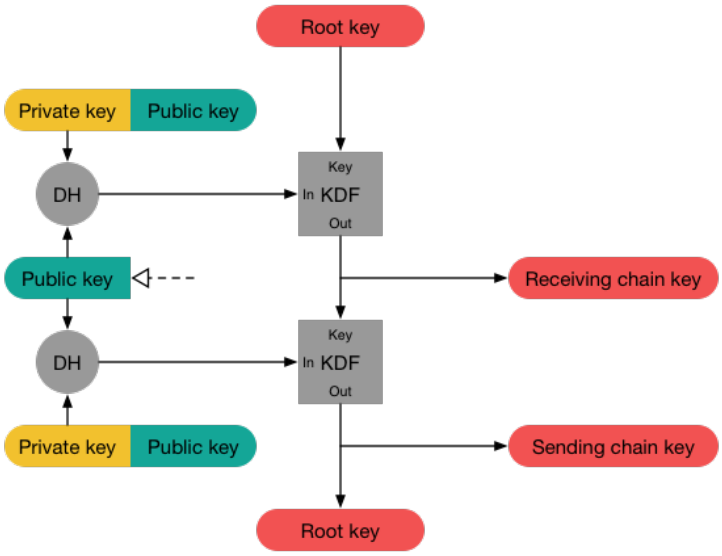


Figure 1: An illustration of a double ratchet system [16]. Here the root key is the shared same for both A and B, the creator of a private-public key pair is alternated, and the receiving and sending chain are mirrored for A and B.

a public key. This public key from the opposite side is used in combination with the previous self-generated private key, for a Diffie-Hellman derivation to find a common secret. This common secret is then used in combination with the root key in the KDF function to calculate a sending chain key.
Simultaneously, the same public key from the opposite side is used with the private key, from a new self-generated private-public key pair, for a Diffie-Hellman derivation to find a common secret. This common secret is then used together with the root key in the KDF function to calculate a receiving chain key. The sending chain key is calculated before the receiving chain key. The use of a new key and a previous key is referred to as a "single ratchet".
In the double ratchet algorithm, every new chain key is used together with the previous chain key in the KDF function to find the encryption or decryption key, respectively. This is done at both sides where the sending and receiving chain keys are mirrored, so A can encrypt a message with the sending chain key and B can decrypt it with the receiving chain key. A representation of the double ratchet algorithm can be found in figure 1.

The Signal protocol combines the X3DH and double ratchet algorithm to ensure end to end encrypted communication between A and B.

3.2 WhatsApp protocol/implementation

From the WhatsApp white paper [8], we learn about the End-to-End security implementation of WhatsApp. As discussed in section 2, the white paper mainly

discusses the implementation and the specific encryption algorithms used, which is not discussed in either of the Signal algorithms. The Signal terminology is changed, as the receiving chain and sending change are not name explicitly for example. Although the terminology changes from Signals', we conclude that according to the specifications of the white paper the session setup and message exchange is the same as in the Signal protocol.

In both WhatsApp and Signal, a new identity key is created during the installation of the application. This identity key is bound to the phone number, which is verified by the application. Since the phone number is verified, WhatsApp can be only activated on one phone at a given time.

3.3 Blocking/Non-blocking behavior

Signal and WhatsApp follow a different approach in the way they handle the changes of a users device.
Signal implements a blocking mechanism, which means that when a receivers' identity key changes, received messages cannot be decrypted anymore: the communication is blocked. As a result from the blocking, all messages that have been sent but have not arrived (pending) are lost and not sent again without user interaction. When the sender broadcasts a new identity key, a new session is established with the new keys and the communication is continued.
WhatsApp implements a non-blocking mechanism. When receivers' identity key changes, the server stores the messages pending instead of deleting them. When the receiver broadcasts his new identity key, the server sends the pending messages back to the sender for re-encryption with receivers' new identity key. Then, the messages are sent again, without user interaction, to the receiver who is able to decrypt and read them.
The Non-blocking mechanism creates a window for possible man-in-the-middle attacks. Suppose that Alice sends a message to Bob while he is offline. Then, Eve registers Bob's phone number with the WhatsApp server (by exploiting vulnerabilities of the mobile network [17], a voicemail attack [11], or by having authorized access to the servers like WhatsApp itself). Alice's WhatsApp client will now automatically, without user interaction, re-encrypt the messages with the attackers key and send it to Eve, who receives it. Only after the act, Alice will get a notification that the encryption keys have changed, if Alice has enabled the notification option, which is disabled by default. WhatsApp justifies this behaviour with two main arguments. Firstly, considering the size of the user space, they chose a simpler user experience than secure implementation. Additionally, they claim that implementation of a blocking mechanism, like in Signal, could expose more information to the server about who has enabled safety number change notifications and who hasn't, effectively telling the server who it could attack transparently with a MITM attack and who it could not [18]. If safety number notification was enabled by default, this leak of information could be avoided. Our Non-blocking experiments which are described in section 5, focus on this behaviour.

4 Methods

For the purpose of this research we assume that the Signal protocol is secure and end-to-end encrypted. This assumption is fortified by the formal approach by Cohn-Gordon et al. [14]. We also assume that the Signal app uses the Signal protocol and the implementation follows the specifications of the protocol. While the latter assumption could be seen as obvious, though at this time there exists no formal proof of this statement.

The WhatsApp environment is a black box in the sense that the implemented code, on both client and server side, is proprietary. Even if the client side code is decompiled, it is obfuscated. In addition, the server side of the communication is inaccessible from us. We will therefore look at experiments from which we can deduce the communication implementation of WhatsApp.
Although the theoretical background shows no difference in implementation between Signal and WhatsApp, as mentioned in section 3, there are known deviations in the implementation of the Signal protocol in WhatsApp, such as non-blocking behavior which is further described below. With the performed experiments we examine the extent of the deviation from the Signal protocol.

Metadata
One more difference we found between the two protocols is in the way they handle the metadata. WhatsApp keeps track and stores most of the metadata acquired by their users. Specifically, they store profile photos, group information, IP addresses, call date and duration, and address book [19]. This information, if leaked, is enough to build a complete profile about someone. If someone placed a phone call from the top of the Golden Gate bridge, while stationary, to a suicide hot-line, the one who intercepts the metadata would not need to hear the conversation to draw conclusions about its contents [20].
Signal claims that, by design, it does not store any of the personal data that could be derived from the metadata, like conversation lists, group memberships, group titles, and time and duration of calls etc. In addition, since October 2018 Open Whisper Systems announced that they developed methods in order to further hide information from the metadata, by encrypting them with the main message, like the sender of the message. They call this technology "Sealed sender" and is already implemented in the application [21].

Hardware and Software Used
For all our experiments we used the latest version available for both messaging applications. At the time of writing, WhatsApps' latest version is 2.18.380 and for Signal the latest version is 4.32.8, and both applications require any Android 4.0 version or above.
We also used four different android devices for our experiments:

- Oneplus 3 smartphone
- LG smartphone
- Nexus 5X smartphone
- Nexus tablet

For capturing the traffic we used a Dell Laptop with Linux Mint OS and we

captured the traffic using Wireshark.

5 Experiments

In this section, there is a detailed description of all the experiments that will be executed during the research. These experiments will help us enrich our knowledge about the implementation of WhatsApp messaging application. Two of those experiments are focused on the protocol analysis of Signal and WhatsApp and the rest of the experiments focus on the way the WhatsApp server handles the undelivered messages.

Traffic Comparison experiment
The *Traffic Comparison experiment* will reveal the extent of similarity between the two message exchange implementations. If the results from this experiment show the traffic to be similar, we can deduce the implementations must be similar as well. However, if the results prove to be dissimilar we can deduce that the protocol is implemented differently for both applications, and could therefore be insecure or not End-to-End encrypted. In this experiment we compare the network traffic of both applications. We divert all WhatsApp and Signal traffic through a computer, which acts as a Wi-Fi hotspot, and then, capture all traffic using Wireshark. We analyze how each application connects to the server, how it exchanges a message, and how the traffic differentiates between the two implementations. For each iteration of this experiment, we follow the same procedure. We start the application, we send a message through the application, WhatsApp or Signal, and then exit the application. During this time we capture all the packets exchanged between the phone and the server via a WiFi hotspot, where we drop the irrelevant ones such as random android traffic produced at this time for example, and we analyze the ones produced by our application. We define such a collection of traffic as a conversation.
The criteria which we evaluate are mainly the pattern of the conversations and the pattern of protocols used by the applications on a message to message basis. The pattern of conversations entails the amount of times one side of the communication sends a message before it receives one, and the size of these messages. The pattern of protocols used entails an arrangement in which of the messages in the conversation are encrypted and which messages are not.
To make the criteria intuitively visible, we create an image of the "conversation" of each packet stream. These images contain information about the sender of every message and the size of each message.

Packet Decryption Experiment
With the *Packet Decryption experiment* we capture all the traffic packets exchanged by the two applications, decrypt them and analyze the data and metadata that a server has access to. To achieve this, we simulate a man in the middle attack to both protocols using Burp Suite. Specifically, we set up a proxy sever to a laptop and all the traffic of WhatsApp and Signal applications is diverted to this server. In addition the android device has to be rooted in order to install a certificate signed by Burp as a trusted certificate. Next, we use the Xposed android application with the sslUnpinning2 module in order to disable the certificate pinning of the applications and make them trust our

server as it was the original communication server.

Basic Blocking experiment
The *Basic Blocking experiment* verifies the WhatsApp non-blocking behavior and serves as a baseline for the following variations of this experiment. The setup for this experiment, as shown in 2, entails three phones with WhatsApp installed: A, B, and C. Phones A and B establish a session between them by exchanging at least one message with each other. Then, we deactivate phone B and send a message to B from Phone A. As described in section 3 this establishes a session between the two phones. We migrate the SIM card from phone B to phone C with a fresh installation of WhatsApp, which means a new identity key is generated, and we verify the number. The phone C will then receive the pending message that was originally encrypted with B's public key. Phone A will not get any notification that the receivers' identity key has changed.

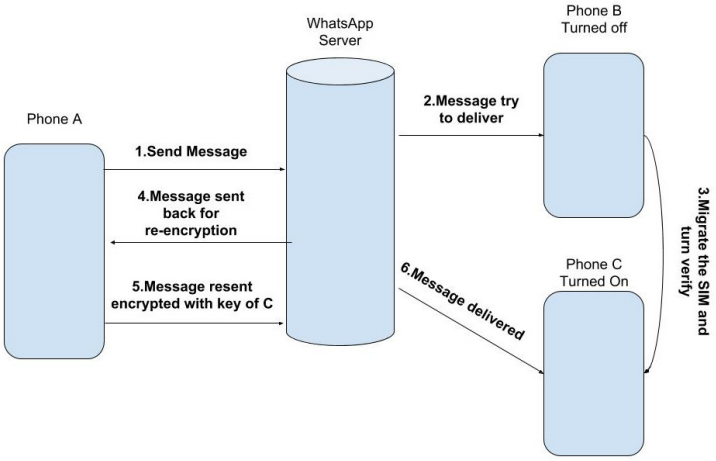


Figure 2: The setup describing the order of events for the *Basic block experiment*.

Sender Offline Blocking experiment
In the *Sender Offline Blocking experiment* our goal is to find out if the re-encryption actually happens at senders' phone A or if it happens at the WhatsApp server. A similar set-up as the *Basic Blocking experiment* will be used, though now, before the SIM card gets migrated we will deactivate the phone A which is the senders' phone. Then, phone C will get back online and the phone number will be verified. The whole setup is shown in 3. If this message is successfully delivered, we can derive that the phone A is not needed for re-encryption of our message and that the re-encryption happens on server-side. As a result, we prove that the private key of phone A is stored on the server and that the protocol is not End-to-End encrypted. In the opposite case, where the message is not delivered while phone A is offline, we can conclude that the displayed behavior is an example of the End-to-End encryption being implemented correctly.

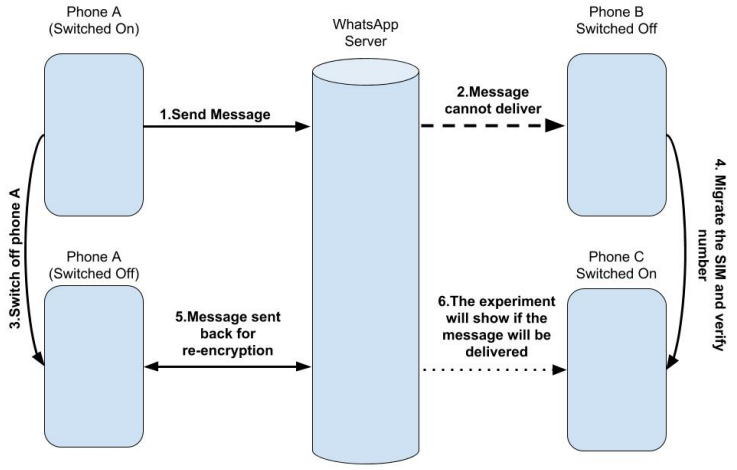


Figure 3: The message exchange order of the *Sender offline blocking experiment*, showing also details about the setup used. Three different phones are used: A,B, and C

Sender Migration Blocking experiment

With the *Sender Migration Blocking experiment* we prove that when we migrate the sender to a different phone, the old encryption keys are lost, and as a result, the previous messages that were not delivered, cannot be decrypted anymore. Supplementary to the *Sender Offline Blocking* experiment we will try one more scenario. We perform the same procedure as before, but now, when phone A is deactivated, we also migrate it to a new phone and re-install the WhatsApp application, as shown in 4. Next, we put both new devices back online. Same as before, this experiment examines if phone A is actually needed in order to decrypt the message. With this experiment we prove that when we migrate the sender to a different phone, the old encryption keys are lost, and as a result, the previous messages that were not delivered, cannot be decrypted anymore.

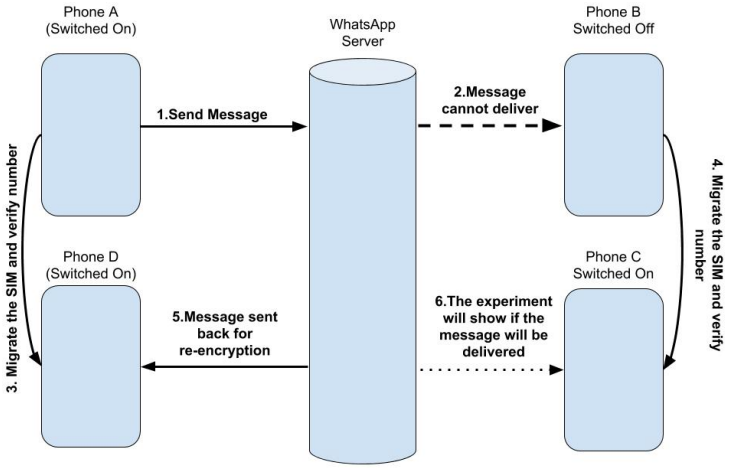


Figure 4: The message exchange order for *Sender Migration Blocking experiment*. For this one we use one more phone, D, in which we migrate the SIM card of the sender.

6 Results

Traffic Comparison experiment

For the extent of this experiment we will denote the phone running either application as "client" and the application server of either application as the "server". As discussed in section 5, we create representations of the conversations.

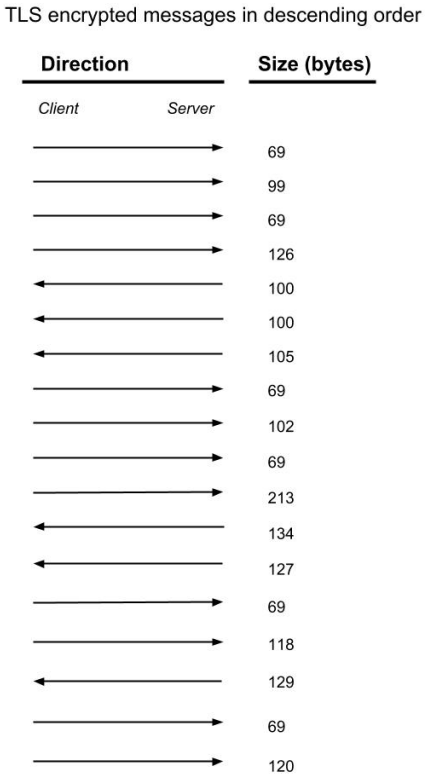


Figure 5: This is an illustration of the captured WhatsApp traffic conversation between the client and the server. Note that all messages are encrypted and no TCP ACK messages are included to decrease clutter.

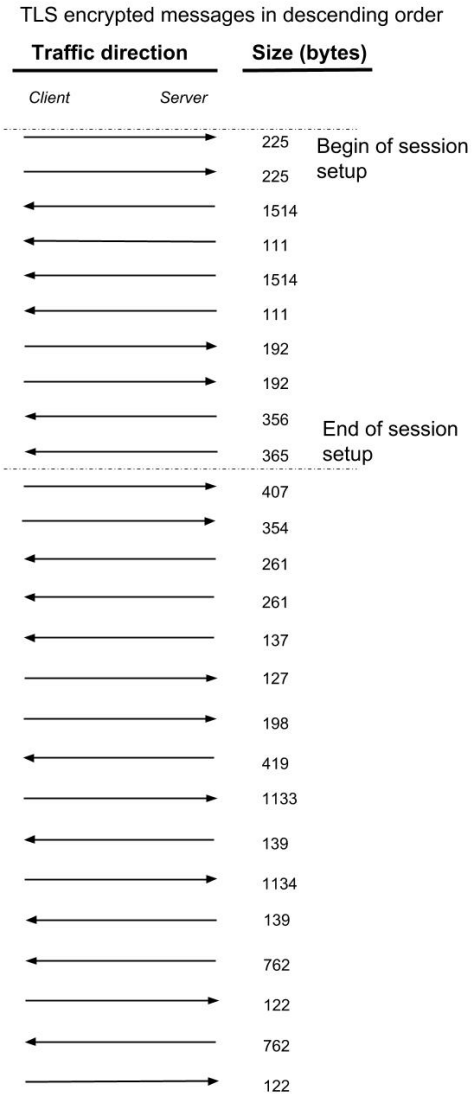
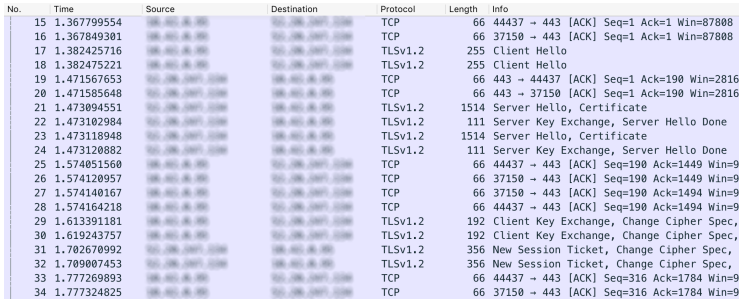


Figure 6: This is an illustration of the captured Signal traffic conversation between the client and the server. Note that all messages are TLS encrypted and no TCP ACK messages are included to decrease clutter and the session setup at the beginning of the conversation.

Figure 5 shows a representation of WhatsApp exchanging one single message and figure 6 shows a similar representation for the Signal conversation. Figure 6 shows that every time the Signal application is started the client and the server exchange messages to create a session. This session setup is not present in the WhatsApp conversation.



No.	Time	Source	Destination	Protocol	Length	Info
15	1.367799554	[redacted]	[redacted]	TCP	66	44437 → 443 [ACK] Seq=1 Ack=1 Win=87808
16	1.367849301	[redacted]	[redacted]	TCP	66	37150 → 443 [ACK] Seq=1 Ack=1 Win=87808
17	1.382425716	[redacted]	[redacted]	TLSv1.2	255	Client Hello
18	1.382475221	[redacted]	[redacted]	TLSv1.2	255	Client Hello
19	1.471567653	[redacted]	[redacted]	TCP	66	443 → 44437 [ACK] Seq=1 Ack=190 Win=2816
20	1.471585648	[redacted]	[redacted]	TCP	66	443 → 37150 [ACK] Seq=1 Ack=190 Win=2816
21	1.473094551	[redacted]	[redacted]	TLSv1.2	1514	Server Hello, Certificate
22	1.473102984	[redacted]	[redacted]	TLSv1.2	111	Server Key Exchange, Server Hello Done
23	1.473118948	[redacted]	[redacted]	TLSv1.2	1514	Server Hello, Certificate
24	1.473120882	[redacted]	[redacted]	TLSv1.2	111	Server Key Exchange, Server Hello Done
25	1.574051560	[redacted]	[redacted]	TCP	66	44437 → 443 [ACK] Seq=190 Ack=1449 Win=9
26	1.574120957	[redacted]	[redacted]	TCP	66	37150 → 443 [ACK] Seq=190 Ack=1449 Win=9
27	1.574140167	[redacted]	[redacted]	TCP	66	37150 → 443 [ACK] Seq=190 Ack=1494 Win=9
28	1.574164218	[redacted]	[redacted]	TCP	66	44437 → 443 [ACK] Seq=190 Ack=1494 Win=9
29	1.613391181	[redacted]	[redacted]	TLSv1.2	192	Client Key Exchange, Change Cipher Spec,
30	1.619243757	[redacted]	[redacted]	TLSv1.2	192	Client Key Exchange, Change Cipher Spec,
31	1.702670992	[redacted]	[redacted]	TLSv1.2	356	New Session Ticket, Change Cipher Spec,
32	1.709007453	[redacted]	[redacted]	TLSv1.2	356	New Session Ticket, Change Cipher Spec,
33	1.777269893	[redacted]	[redacted]	TCP	66	44437 → 443 [ACK] Seq=316 Ack=1784 Win=9
34	1.777324825	[redacted]	[redacted]	TCP	66	37150 → 443 [ACK] Seq=316 Ack=1784 Win=9

Figure 7: This a screen capture of the Signal conversation from Wireshark. The capture shows the double session setup at the beginning of the conversation. This setup information is only visible for the Signal conversation. Note that the TCP connections are visible in this image.

From the comparison of the figures we can make three observations. Every encrypted message in the WhatsApp conversation receives a TCP ACK message as a reply. While most encrypted messages in the Signal conversation also receive TCP ACK messages as a reply, some do not. An example can be seen in figure 7 at packet number 29 and 30. We can notice that during the setup of the session in Signal, two distinct sessions are opened.

The second observation we can make is that every message in the WhatsApp conversation from the client to the server is proceeded by a message with a size of 69 bytes.
The third observation we can make is that the overall size of the WhatsApp conversation was smaller than the overall size of the Signal conversation. We notice this is due to the dual session setup, where large files are exchanged. From the sizes of the messages in the WhatsApp conversation we can deduce that no encryption keys are being sent.

Overall, we can conclude that the protocol implementations of both applications are not the same.

Man-In-The-Middle attack using Burp experiment
We split the results of this experiment in two parts: the Signal and WhatsApp results: With Signal, we are able to set up the attack and read the metadata exchanged and the exact way the protocol is set up. It is worth mentioning that some of the data that we are able to capture is the phone number of the sender and the receiver, part of the initialization key, and the timestamp of the message as you can see in Figures 8 and 9.

Figure 8: One of the messages we captured with Burp Suite, showing the receivers phone number and the timestamp of the message.

```
GET /v1/websocket/?login=+31647229265&password=twqrl/6AmDskyse3mqQDxtaR HTTP/1.1
X-Signal-Agent: OWA
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: s5Q7IWYO08+06EbkKQxQsw==
Sec-WebSocket-Version: 13
Host: textsecure-service.whispersystems.org
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.9.0
```

Figure 9: One of the messages we captured with Burp Suite, showing the senders phone and the password used in order to initialize the communication keys.

With WhatsApp we couldn't intercept any of the traffic with Burp Suite. The problem for us is that WhatsApp implements an extra layer of obfuscation by hiding the Signal protocol under Noise protocol [22]. The method is mentioned as Noise Pipes in WhatsApps' whitepaper, during their communication to WhatsApp server. This adds an extra layer of security to the application but the security of Signal protocol is not based on this characteristic. As a matter of fact, we are not able to make an objective comparison between the two protocols with this experiment and we propose it as a future work. We do note that this layer is not specified by the Signal protocol, which further enhances the conclusion from the Traffic comparison experiment.

Basic Blocking experiment

The results from the basic blocking experiment confirmed the non-blocking mechanism of WhatsApp. When we reactivated the WhatsApp account at the new phone C, we immediately received the message that phone A sent, as expected, which can be seen in figure 10. It is worth mentioning that phone A doesn't get any indication that the keys have changed by default unless the user change this setting manually. This behaviour introduces a possible risk. If the receivers' WhatsApp gets compromised, then the sender will never find out that the communication has been diverted to a different phone. Although, the victim will get a notification to his phone that the WhatsApp has been activated to a new device, which can reveal possible suspicious behaviour.

On the other side, Signals' protocol is blocking. When the same experiment is performed the message is never delivered. In addition, every time the SIM card is swapped between devices devices, we get a security related message that the keys have changed as shown in figure 11. This message is shown before any user message exchange has occurred and it is the default behaviour of the application.

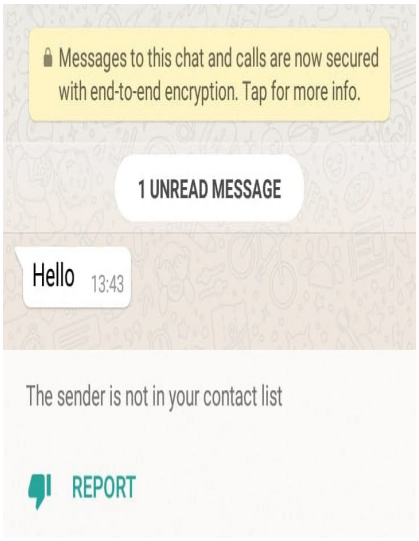


Figure 10: This is a message that phone C gets after it verifies the phone number with the WhatsApp server.

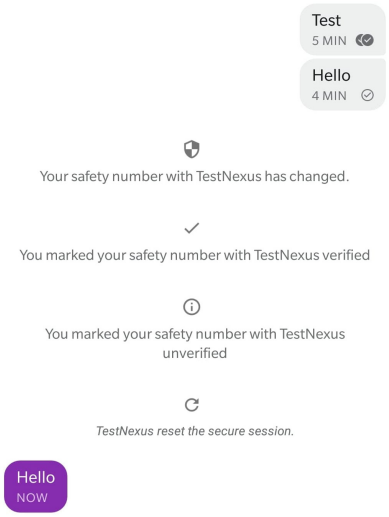


Figure 11: This is a message that phone C gets after it verifies the phone number with the Signal server.

Sender Offline Blocking experiment

The results of the *sender offline blocking experiment* give insight of secure protocol implementation. When we switch on phone B while phone A is still inactive, phone B immediately receives the message seen in figure 12. When the phone A comes back online, the message is delivered immediately, after it is sent to phone A. Despite the fact that this is the expected behaviour of the non-blocking protocol, it gives extra information to a possible attacker. If the phone C gets compromised, the attacker knows that: a) a message is pending to be delivered and b) the phone A is currently offline. This information allows the attacker to try the voicemail exploit [11] in phone A.

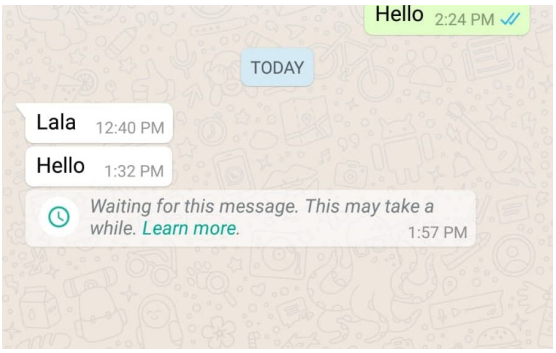


Figure 12: The message that phone C gets after he verifies the phone number.

Sender Migration Blocking experiment
The results of the sender migration blocking experiment give us additional information about the implementation. Like the previous experiment, when we turn on phone B while A is inactive, we get a notification that a message is pending. When we turn on phone A in a new device, and as a result with a different master key, we can continue the communication with phone A but the message we sent before will stay undelivered as shown in figure 13. Specifically, the message will never be able to be re-encrypted and according to WhatsApp, it will stay in their servers, waiting for delivery for 30 days before it is deleted [19]. Unfortunately, due time constraints, we were not able to verify that the message will actually be deleted after 30 days.

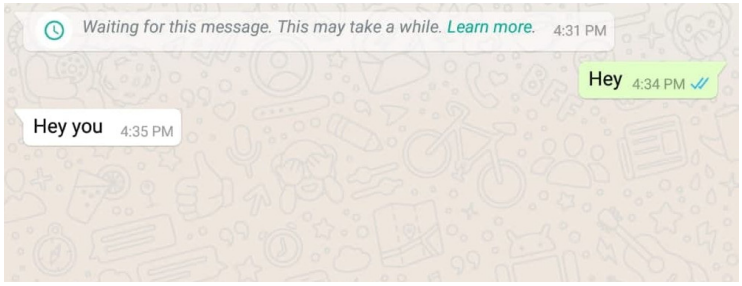


Figure 13: The message that phone C gets after he verifies the phone number.

As stated in section 5, we can conclude that this is an example of End-to-End encryption of WhatsApp traffic.

7 Discussion

In our *Traffic Comparison experiment* we compared the traffic from both Signal and WhatsApp applications. We expected that the traffic exchanged would look more similar. Contradictory, we found out that there was deviation between the two applications in the average packet size as well as the connection establishment procedure.
In addition, in this experiment we got some extra information which we chose to disregard, such as timing. We believe that the added value of this information would only add noise to the comparison and should be omitted. We decided to focus on the size of the packets and the order of the messages that are exchanged.
In addition, it would be beneficial for our research if we were able to decrypt WhatsApps' packets. This experiment would have given us better understanding on the way the protocol is implemented, and we would be able to make a more extensive comparison between the two applications.
Lastly, our research and experiments gave us insights about the End-to-End encryption of WhatsApp. While we can not prove the End-to-End encryption of WhatsApp, given our experiments and the risks involved for the company and parent company, we believe the claim that WhatsApp is End-to-End encrypted. Our results prove that WhatsApp deviates from the Signal application, on which the WhatsApp implementation is based, and therefore we believe that WhatsApp should be further examined.

8 Conclusion

In this research, we have reviewed the End-to-End encryption implementation of the WhatsApp application. We have examined previous research regarding the Signal protocol security, attacks on the WhatsApp protocol implementation, and the WhatsApp encryption overview. To substantiate our understanding, we have examined the Signal protocol components, and the WhatsApp implementation of the Signal protocol in section 3. There we answer the first sub-question: What are the algorithms used to create the Signal protocol?
Since WhatsApp is proprietary software, we chose to create experiments that allowed us to deduce the communication implementation of WhatsApp. With the assumptions made in section 4, we created several experiments that gave us insights in the WhatsApp protocol implementation.
We have done a traffic comparison of standard message behavior of both applications, and found them to be different. We have verified the non-blocking behavior of WhatsApp, we created an experiment to test a possible End-to-End encryption flaw, which proved to be secure. We created an experiment that gives an example of the correct implementation of the Signal protocol by WhatsApp. With these experiments we answer the third sub-question: What are the differences between Signal and WhatsApp network traffic?
We have attempted to decrypt the traffic of both applications. While we decrypted the SSL layer of the Signal traffic, the proprietary noise pipe implementation used by WhatsApp withheld us from further attempts of decryption. We therefore have no decisive answer for the second sub-question: To what extent are WhatsApp messages adhering to the Signal protocol specifications?
While our research provides deductive results, the experiments and background give us an insight into the WhatsApp protocol implementation which leads us to believe that the risk for the companies involved is too great to falsify the claim of End-to-End encryption. In conclusion, we believe that, at this time, WhatsApp is End-to-End encrypted.

9 Future work

During the decryption experiment, we were able to extract the key from the application which was obfuscated as a series of bytes. In order to use it for decryption of messages, and further verification of the protocol usage, reverse engineering of the WhatsApp is required.

Mark Zuckerberg, Facebook's chief executive, plans to merge all messaging applications owned by the company,WhatsApp,Messenger and Instagram, together. These services will continue to work as standalone applications, but their underlying technical infrastructure will be unified [23]. The integration of the applications, according to Zuckerberg, will be completed by the end of 2019 or early in 2020. We believe that a change in the infrastructure of the three major messaging application could be an interesting topic of research.

WhatsApp allows verification of a new number by phone call. During our research we found that this could be abused by compromising the voicemail of the users phone [11], and as a results it adds vulnerabilities to the application. It should be examined if this feature gives value to the application and how it could be implemented in a secure way.

References

[1] Wired, "FACEBOOK EXPOSED 87 MILLION USERS TO CAMBRIDGE ANALYTICA," 2018, last accessed 6 February 2019. [Online]. Available: https://www.wired.com/story/facebook-exposed-87-million-users-to-cambridge-analytica/

[2] ——, "HOW FACEBOOK HACKERS COMPROMISED 30 MILLION ACCOUNTS," 2018, last accessed 6 February 2019. [Online]. Available: https://www.wired.com/story/how-facebook-hackers-compromised-30-million-accounts/

[3] ——, "FACEBOOK EXPOSED 6.8 BILLION USERS' PHOTOS TO CAP OFF TERRIBLE 2018," 2018, last accessed 6 February 2019. [Online]. Available: https://www.wired.com/story/facebook-photo-api-bug-millions-users-exposed/

[4] Whatsapp, "end-to-end encryption," 2016, last accessed 6 February 2019. [Online]. Available: https://blog.whatsapp.com/10000618/end-to-end-encryption

[5] Statista, "Most popular mobile messaging apps worldwide as of October 2018, based on number of monthly active users (in millions)," 2019, last accessed 10 January 2019. [Online]. Available: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/

[6] M. J. Johnston, D. King, S. Arora, N. Behar, T. Athanasiou, N. Sevdalis, and A. Darzi, "Smartphones let surgeons know whatsapp: an analysis of communication in emergency surgical teams," *The American Journal of Surgery*, vol. 209, no. 1, pp. 45–51, 2015.

[7] M. Mars and R. E. Scott, "Whatsapp in clinical practice: A literature," *The Promise of New Technologies in an Age of New Health Challenges*, p. 82, 2016.

[8] WhatsApp, "Whatsapp encryption overview," April 5, 2016, p. 12.

[9] M. Marlinspike, "Whatsapp's signal protocol integration is now complete," 2016, last accessed 9 January 2019. [Online]. Available: https://signal.org/blog/whatsapp-complete/

[10] P. Rösler, C. Mainka, and J. Schwenk, "More is less: On the end-to-end security of group chats in signal, whatsapp, and threema," 2018.

[11] M. Vigo, "Compromising online accounts by cracking voicemail systems)," 2018, last accessed 21 January 2019. [Online]. Available: https://www.martinvigo.com/voicemailcracker/

[12] M. Marlinspike and T. Perrin, "The x3dh key agreement protocol," *Open Whisper Systems*, 2016.

[13] T. Perrin and M. Marlinspike, "The double ratchet algorithm," *GitHub wiki*, 2016.

[14] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the signal messaging protocol," in *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on.* IEEE, 2017, pp. 451–466.

[15] Z. Dai, T.-W. Chua, D. K. Balakrishnan, V. L. Thing *et al.*, "Chat-app decryption key extraction through information flow analysis," 2017.

[16] Open Whisper Systems, "Double ratchet algorithm," 2019, [Online; accessed 6 February, 2019]. [Online]. Available: https://signal.org/docs/specifications/doubleratchet/

[17] P. Technologies, "WhatsApp Encryption Rendered Ineffective by SS7 Vulnerabilities," 2016, last accessed 26 January 2019. [Online]. Available: https://www.ptsecurity.com/ww-en/about/news/117350/

[18] M. Marlinspike, " There is no WhatsApp 'backdoor')," 2017, last accessed 22 January 2019. [Online]. Available: https://signal.org/blog/there-is-no-whatsapp-backdoor/

[19] WhatsApp, "Information for Law Enforcement Authorities)," 2016, last accessed 26 January 2019. [Online]. Available: https://faq.whatsapp.com/en/android/26000050/?category=5245250

[20] R. Falkvinge, "WhatsApp Encryption Shows Value Of Metadata)," 2014, last accessed 26 January 2019. [Online]. Available: https://www.privateinternetaccess.com/blog/2014/11/whatsapp-encryption-shows-value-of-metadata/

[21] jlund, "Technology preview: Sealed sender for Signal)," 2018, last accessed 26 January 2019. [Online]. Available: https://signal.org/blog/sealed-sender/

[22] T. Perrin, "The Noise Protocol Framework," 2018, last accessed 30 January 2019. [Online]. Available: https://noiseprotocol.org/noise.html

[23] N. Y. Times, "Zuckerberg Plans to Integrate WhatsApp, Instagram and Facebook Messenger," 2019, last accessed 1 February 2019. [Online]. Available: https://www.nytimes.com/2019/01/25/technology/facebook-instagram-whatsapp-messenger.html