ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/367350335

A security analysis comparison between Signal, WhatsApp and Telegram

Article · January 2023

CITATIONS
3

READS
1,235

3 authors, including:

Emil Simion
Polytechnic University of Bucharest
118 PUBLICATIONS **340** CITATIONS

SEE PROFILE

All content following this page was uploaded by Emil Simion on 23 January 2023.

The user has requested enhancement of the downloaded file.

A security analysis comparison between Signal, WhatsApp and Telegram

Corina-Elena Bogos[*], Razvan Mocanu[†], Emil Simion [‡]

January 21, 2023

Abstract

This paper aims to provide a security analysis comparison between three popular instant messaging apps: Signal, WhatsApp and Telegram. The analysis will focus on the encryption protocols used by each app and the security features they offer. The paper will evaluate the strengths and weaknesses of each app, and provide a summary of their overall security posture. Additionally, this paper will discuss other considerations such as user base, data collection and usage policies, and other features which may impact the security of the apps. The results of this analysis will provide insights for individuals and organizations looking to choose a secure instant messaging app for their communication needs. In this paper we reviewed the main encryption standards and we compared the features, traffic analysis, protocols, performance and recent security breaches for WhatsApp, Signal and Telegram. The paper includes packet sniffing using Wireshark and Fiddler.

Keywords: security analysis comparison, encryption protocols, packet sniffing.

1 Introduction

End-to-end encryption based instant messaging apps have captured the attention of many users due to increased security, ease of use and and privacy concerns. End-to-end encryption is used when data security is necessary, including in the communications, healthcare and e-commerce industries. Companies use the encryption to comply with data privacy and security laws. This paper starts by presenting end-to-end encryption, what it is and what other standards exist. Some differences between the features of Telegram , WhatsApp and Signal have been written. It is talked in detail from physical forensic analysis and results from other authors. The experiments are about traffic analysis and packet analysis. The paper contains information about all the protocols used by each application and the most recent vulnerabilities and security breaches.
Signal, WhatsApp, and Telegram are all popular instant messaging apps that offer end-to-end encryption to secure conversations. However, there are some key differences in their security features and implementation.

Signal is widely considered to have the strongest encryption and security features among the three. It uses the Signal Protocol, which is considered to be one of the most secure encryption protocols available. It also has a number of security-enhancing features such as the ability to verify the identity of contacts and the option to enable disappearing messages.

WhatsApp also uses end-to-end encryption, but it uses the less-proven WhatsApp Protocol for encryption. WhatsApp also has a larger user base and therefore a bigger attack surface.

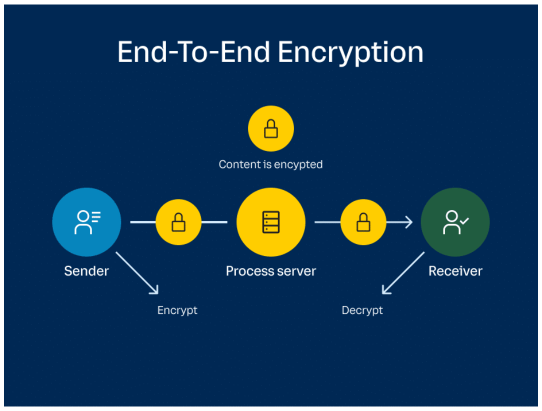
[*]Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Email:corina.iftinca.v@gmail.com
[†]Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Email: mocanurazvan123@gmail.com
[‡]Politehnica University of Bucharest, Email: emil.simion@upb.ro

However, it is owned by Facebook, which is known for its data collection and usage practices, this could raise some concerns about privacy

Telegram also uses end-to-end encryption, but it has some limitations compared to Signal and WhatsApp. Telegram's encryption is optional and is only available in its "Secret Chats" feature, which means that the majority of users may not be using end-to-end encryption. Telegram also stores the encryption keys on its servers, which can be considered as a security vulnerability.

1.1 What is End-to-End Encryption

The major security feature of secure instant messaging is end-to-end encryption, that is used for protecting the protocol security when considering malicious server-based attacks.
In an end-to-end encrypted system, the only entities that can access the data are the sender and the intended recipients – no one else. Neither hackers nor unwanted third parties can access the encrypted data on the server. In true end-to-end, encryption occurs at the device level. Messages and files are encrypted before they leave the phone or computer and are not decrypted until they reach their destination.
Communications encryption in which data is encrypted when being passed through a network, but routing information remains visible.[nis21]



End-to-end encryption schema.[Rin21]

1.2 Other encryption standards

Before the wide use of end-to-end encryption apps, most protocols used encryption at rest or encryption in transit. Both methods are inferior in terms of protection against eavesdropping or message modification attacks.

Data at rest is the data housed on computer data storage in any digital form, whether it's in cloud storage, file hosting services, or databases.Encryption at rest is designed to prevent the attacker from accessing the unencrypted data by ensuring the data is encrypted when on disk. Encryption in transit protects data in motion by encrypting and decrypting the data over every hop in a network. This means that encryption in transit has the decrypted data on the server and on the endpoint devices.

1.3 Feature Comparison

Instant messaging tools are frequently used, as part of social media. Table 1 shows side-by-side the feature comparisons for WhatsApp, Telegram and Signal to give some perspective on where they stand. All three are free, use end-to-end encryption, have the possibility of sending images, videos or different files. For video calls or voice calls Telegram is not recommended due to the fact that it doesn't have this features. Now, the instant messengers focus on security issues, with completely safe, private and self-destructing messages, so it is natural to wonder which of the three one is safer.

Features			
Feature Name	WhatsApp	Telegram	Signal
Backups	✓	✓	local backup
Block user	✓	✓	✓
Broadcast	✓	–	✓
Group chat	✓	✓	✓
Online status	✓	✓	–
Price	Free	Free	Free
Secure conversation	E2E encryption	E2E encryption (Not Default)	E2E encryption
Send images	✓	✓	✓
Send videos	✓	✓	✓
Send files	✓	✓	✓
Share contact	✓	✓	✓
Share location	✓	✓	✓
Video calls	✓	–	✓
Voice calls	✓	–	✓

Table 1: WhatsApp, Telegram and Signal specs comparison.

2 Security

2.1 Forensics Analysis

Computer forensics is the application tehniques and investigation methods to obtain and protect evidence from a particular computing device in a way that is suitable for presentation in a court of law. The goal of computer forensics is to perform a structured investigation and maintain a documented chain of evidence to find out exactly what happened on a computing device and who was responsible for it. In this following subsection, we will talk about Signal, WhatsApp and Telegram and how well the apps hid user data.

2.1.1 WhatsApp

A very useful feature for clients, the offline backups WhatsApp does are the key to obtaining encrypted data with sensitive information. For the forensic analysis of WhatsApp, the authors in paper [Tha13] used a UFED physical analyzer to obtain the file system extraction, database files with details of chat sessions. Xtract 2.0 was also used to organize the database files in a HTML. A vulnerability of the AES cypher implementation on android made it possible to obtain the key. All messages, phone numbers and statuses can be seen.

2.1.2 Telegram

In [ACG17] there was forensic analysis of artifacts generated by Telegram on Android smartphones. Telegram is the only instant messaging app in this paper that doesn't automatically use end-to-end encryption. Each user is identified by a Telegram ID and a Telegram username and a profile photo. Telegram also stores various data in the internal memory of the device like all the contacts of a user, the chronology of voice calls, photos, videos and messages in chats. Even telegram groups can be gathered from this analysis.

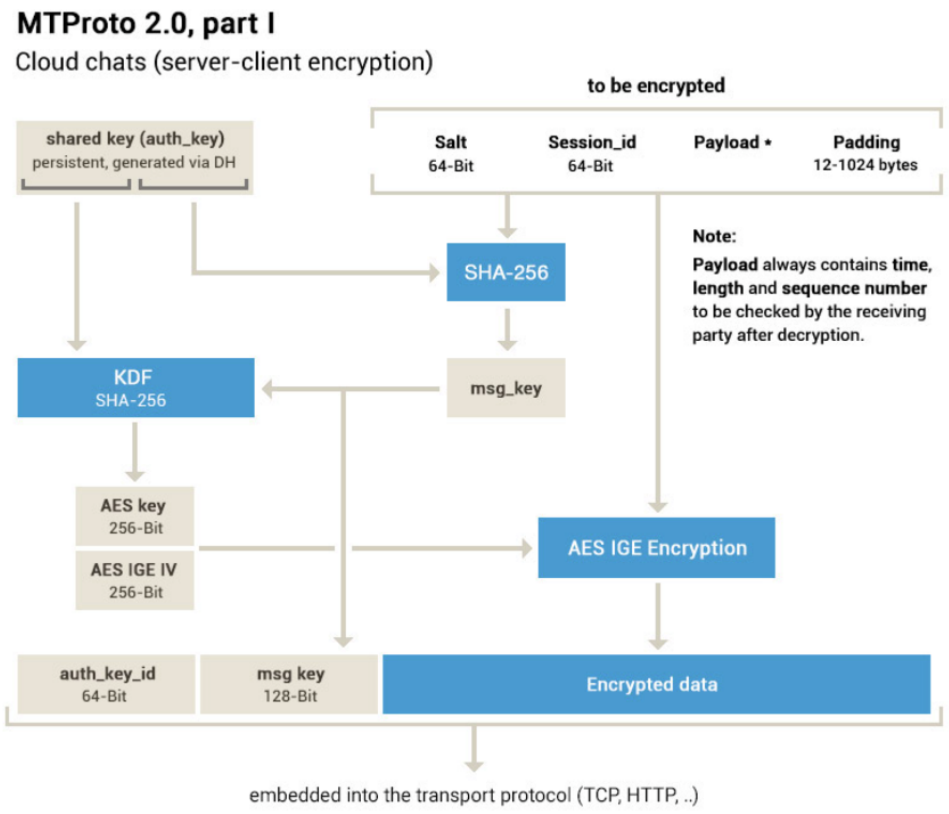
2.1.3 Signal

For signal data forensics was attempted using UFED 4PC (Universal Forensic Extraction Device) and UFED Physical Analyzer version 6.3.11.36 with an internal build version 4.7.1.477 in [Jud18].
The parent company of Signal, Whisper Systems has released the complete source code of the application so developers can report any problems back to the creators if a component is not functioning as it should. This enables developers to make their own copies of Signal using the same encryption code. Whisper Systems will provide support for their own applications ans server but not for user made ones. Because Signal stores messages, keys, and passphrases on the user device a physical attack would be the best course of action.
The extraction is physical and it gathers all the bits of information from the hard disk. Due to the increase security of phones, a big factor in the success of data forensics for signal is the model and operating system of the mobile device.
The Signal app has some glaring problems when there is access to the physical device. Deleted messages, timestamps, who sent the message and even its own status can be read on some phones models. Using the UFED analyser even Signal contact phone numbers and photos were possible to obtain.

2.2 Protocol Models

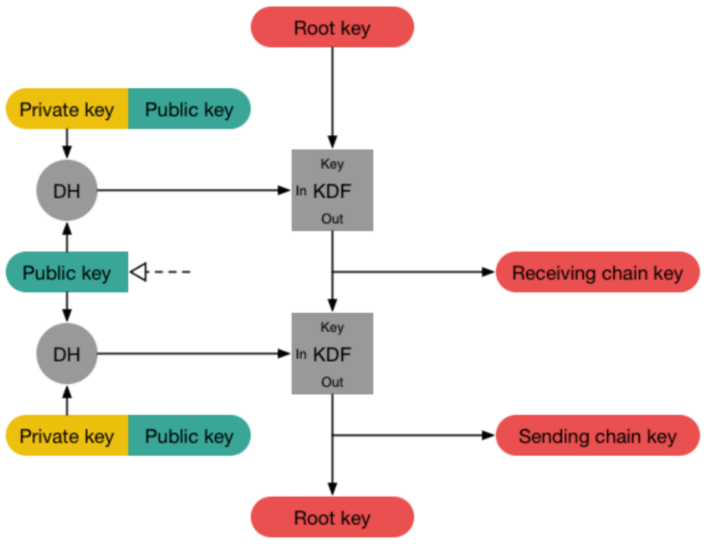
2.2.1 Telegram

Telegram uses the MTProto 2.0 protocol for both server client encryption and end-to-end encryption. Before a message (or a multipart message) is transmitted over a network using a transport protocol, it is encrypted in a certain way, and an external header is added at the top of the message that consists of a 64-bit key identifier auth_key_id (that uniquely identifies an authorization key for the server as well as the user) and a 128-bit message key msg_key. The indepth client, server, client interaction can be best understood from the figure below: [mtp]



2.2.2 Signal

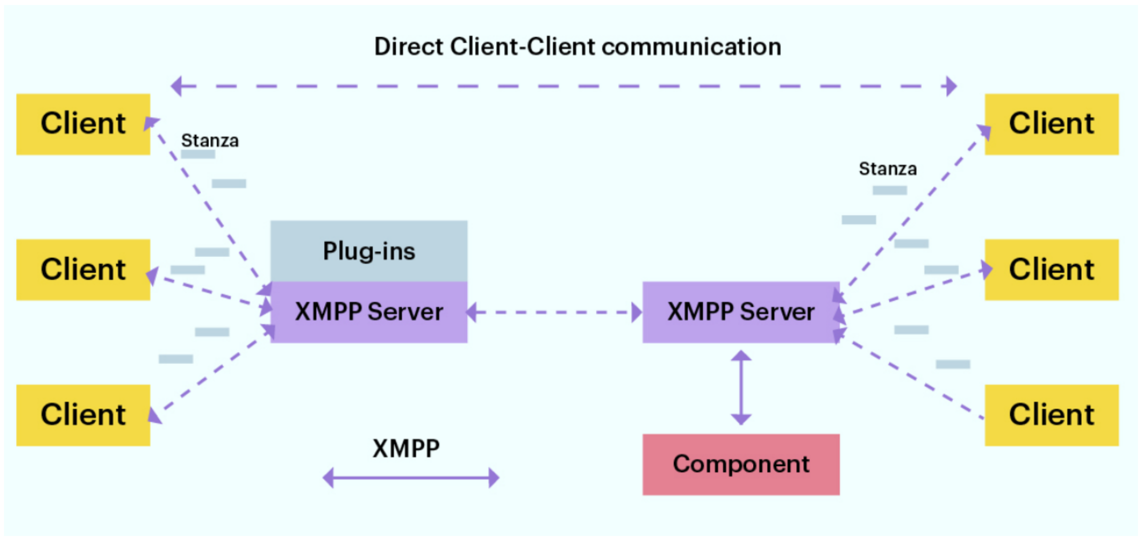
Signal uses the Double Ratchet algorithm, which is used by two parties to exchange encrypted messages based on a shared secret key. The parties derive new keys for every Double Ratchet message so that earlier keys cannot be calculated from later ones. The parties also send Diffie-Hellman public values attached to their messages. The results of Diffie-Hellman calculations are mixed into the derived keys so that later keys cannot be calculated from earlier ones. These properties give some protection to earlier or later encrypted messages in case of a compromise of a party's keys.[sig16]



KDF is defined as a cryptographic function that takes a secret and random KDF key and some input data and returns output data. The output data is indistinguishable from random provided the key isn't known (i.e. a KDF satisfies the requirements of a cryptographic "PRF"). If the key is not secret and random, the KDF should still provide a secure cryptographic hash of its key and input data. The HMAC and HKDF constructions, when instantiated with a secure hash algorithm, meet the KDF definition.

2.2.3 WhatsApp

For data transmission, WhatsApp uses an open and free protocol called XMPP. It is based on XML and allows exchanging text messages, audio/video data, and files. WhatsApp uses the Extensible Messaging and Presence Protocol (XMPP) as the messaging protocol for its platform. XMPP is an open standard for real-time communication, which is used for instant messaging and online presence detection. WhatsApp uses a modified version of XMPP, which is optimized for mobile devices and low-bandwidth networks. The XMPP protocol is responsible for facilitating the exchange of messages between users and for handling other features such as group chat, voice and video calls, and file sharing.[com16]

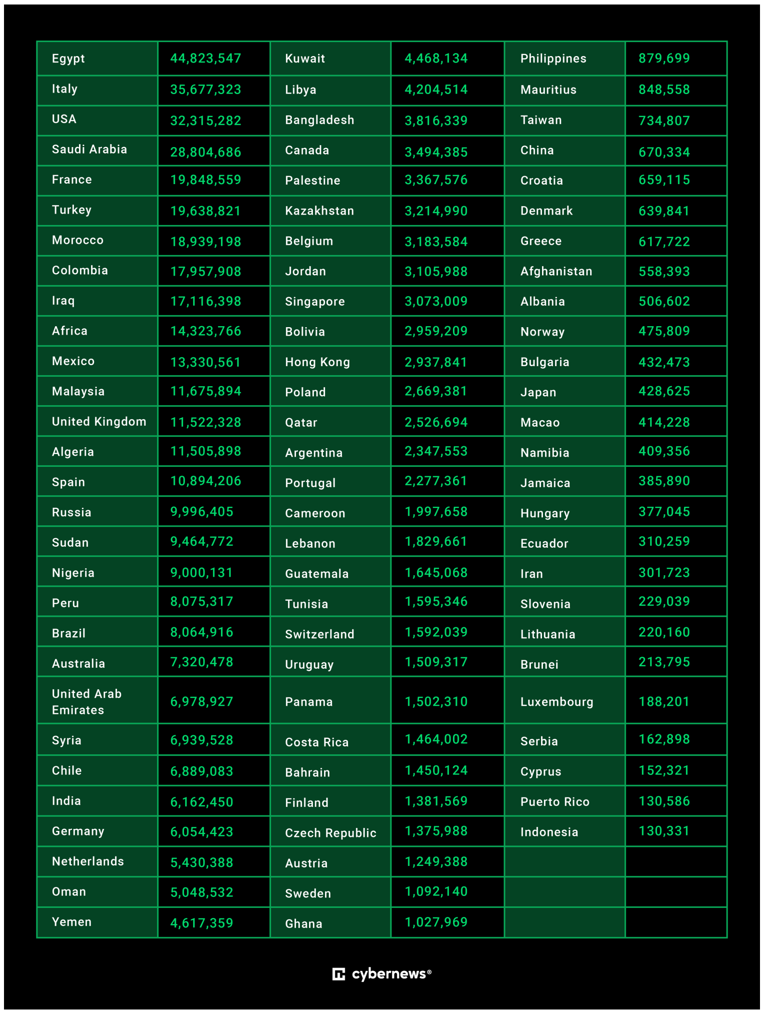


The Extensible Messaging and Presence Protocol (XMPP) is based on the client-server model of communication, where a user's device (client) connects to a server in order to send and receive messages. When a user wants to send a message, the client establishes a connection to the server and sends the message in an XML format, which is then processed by the server. The server then delivers the message to the intended recipient's client, which receives and displays the message. XMPP also includes a presence feature which allows clients to detect and track the online status of other users, this is done by sending presence configurations from the client to the server and from the server to the other clients that are interested in the user's status. Additionally, XMPP supports many other features such as group chat, file sharing, voice and video calls and more by using different types of stanzas and extensions. It's important to note that XMPP is an open standard, which means that any developer can create their own server and clients that are compatible with the XMPP protocol, this also allows different servers to communicate with each other.

2.3 Security Breaches

2.3.1 WhatsApp

On November 25th 2022, a massive active user list was dropped because of API scrapping Meta refused to patch. The numbers of active users were sold for over 2000 dollars and will be use for vishing or phishing attacks. [cyb22]



Egypt	44,823,547	Kuwait	4,468,134	Philippines	879,699
Italy	35,677,323	Libya	4,204,514	Mauritius	848,558
USA	32,315,282	Bangladesh	3,816,339	Taiwan	734,807
Saudi Arabia	28,804,686	Canada	3,494,385	China	670,334
France	19,848,559	Palestine	3,367,576	Croatia	659,115
Turkey	19,638,821	Kazakhstan	3,214,990	Denmark	639,841
Morocco	18,939,198	Belgium	3,183,584	Greece	617,722
Colombia	17,957,908	Jordan	3,105,988	Afghanistan	558,393
Iraq	17,116,398	Singapore	3,073,009	Albania	506,602
Africa	14,323,766	Bolivia	2,959,209	Norway	475,809
Mexico	13,330,561	Hong Kong	2,937,841	Bulgaria	432,473
Malaysia	11,675,894	Poland	2,669,381	Japan	428,625
United Kingdom	11,522,328	Qatar	2,526,694	Macao	414,228
Algeria	11,505,898	Argentina	2,347,553	Namibia	409,356
Spain	10,894,206	Portugal	2,277,361	Jamaica	385,890
Russia	9,996,405	Cameroon	1,997,658	Hungary	377,045
Sudan	9,464,772	Lebanon	1,829,661	Ecuador	310,259
Nigeria	9,000,131	Guatemala	1,645,068	Iran	301,723
Peru	8,075,317	Tunisia	1,595,346	Slovenia	229,039
Brazil	8,064,916	Switzerland	1,592,039	Lithuania	220,160
Australia	7,320,478	Uruguay	1,509,317	Brunei	213,795
United Arab Emirates	6,978,927	Panama	1,502,310	Luxembourg	188,201
Syria	6,939,528	Costa Rica	1,464,002	Serbia	162,898
Chile	6,889,083	Bahrain	1,450,124	Cyprus	152,321
India	6,162,450	Finland	1,381,569	Puerto Rico	130,586
Germany	6,054,423	Czech Republic	1,375,988	Indonesia	130,331
Netherlands	5,430,388	Austria	1,249,388		
Oman	5,048,532	Sweden	1,092,140		
Yemen	4,617,359	Ghana	1,027,969		

cybernews®

In Table 2 are presented the most significant vulnerabilities that appeared in the last 3 years on WhatsApp. [CVE22c]

Security vulnerabilities for WhatsApp			
CVE ID	Vulnerability type	Score	Short description
CVE-2021-24043	-	6.4	A missing bound check in RTCP flag parsing code could have allowed an out-of-bounds heap read if a user sent a malformed RTCP packet during an established call.
CVE-2021-24042	-	7.5	The calling logic for WhatsApp could have allowed an out-of-bounds write if a user makes a 1:1 call to a malicious actor.
CVE-2020-1909	Exec Code Mem. Corr.	7.5	A use-after-free in a logging library in WhatsApp could have resulted in memory corruption, crashes and potentially code execution.
CVE-2020-1907	Exec Code Overflow	7.5	A stack overflow could have allowed arbitrary code execution when parsing the contents of an RTP Extension header.
CVE-2020-1891	-	7.5	A user controlled parameter used in video call could have allowed an out-of-bounds write on 32-bit devices.

Table 2: Vulnerabilities found on WhatsApp.

2.3.2 Telegram

In Table 3 are presented the most significant vulnerabilities that appeared in the last 2 years on Telegram. [CVE22b]

Security vulnerabilities for Telegram			
CVE ID	Vulnerability type	Score	Short description
CVE-2021-40532	-	7.5	Telegram Web K Alpha before 0.7.2 mishandles the characters in a document extension.
CVE-2021-37596	XSS	4.3	Telegram Web K Alpha 0.6.1 allows XSS via a document name.
CVE-2021-36769	-	5.0	An attacker can cause the server to receive messages in a different order than they were sent a client.
CVE-2021-31321	Overflow	5.8	A remote attacker might be able to overwrite Telegram's stack memory out-of-bounds on a victim device via a malicious animated sticker.
CVE-2021-31321	Overflow	5.8	A remote attacker might be able to overwrite heap memory out-of-bounds on a victim device via a malicious animated sticker.

Table 3: Vulnerabilities found on Telegram.

2.3.3 Signal

In Table 4 are presented the most significant vulnerabilities that appeared in the last 5 years on Signal. [CVE22a]

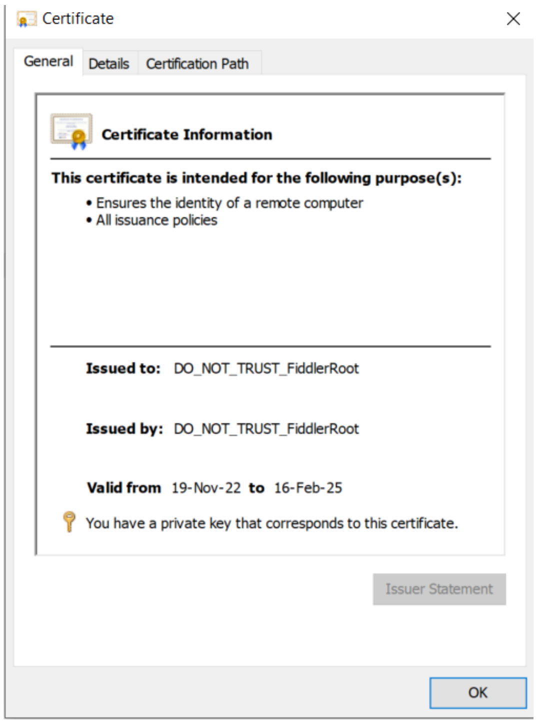
Security vulnerabilities for Signal			
CVE ID	Vulnerability type	Score	Short description
CVE-2022-28345	-	5.0	An attacker can spoof, for example, example.com, and masquerade any URL with a malicious destination.
CVE-2020-5753	-	5.0	Signal Private Messenger allows a remote non-contact to ring a victim's Signal phone and disclose currently used DNS server.
CVE-2019-17192	DoS	7.5	A remote attacker could easy cause a denial of service.
CVE-2019-17191	-	5.0	The Signal Private Messenger allows a caller to force a call to be answered and the audio channel may be open before the callee can block eavesdropping.
CVE-2018-16132	-	7.8	A large image sent to a user to exhaust all available memory when the image is displayed, resulting in a forced restart of the device.

Table 4: Vulnerabilities found on Signal.

3 Experimental Results

3.1 Package Sniffing

In this section all HTTPS traffic was decrypted using a tool from Fiddler that created an obviously fake certificate that we allowed on our testing machine.


Certificate

General | Details | Certification Path

Certificate Information

This certificate is intended for the following purpose(s):
- Ensures the identity of a remote computer
- All issuance policies

Issued to: DO_NOT_TRUST_FiddlerRoot

Issued by: DO_NOT_TRUST_FiddlerRoot

Valid from 19-Nov-22 **to** 16-Feb-25

You have a private key that corresponds to this certificate.

Issuer Statement

OK

Continuing this we will have to get a hold of the secret keys, but access to the physical device is needed.

3.1.1 WhatsApp

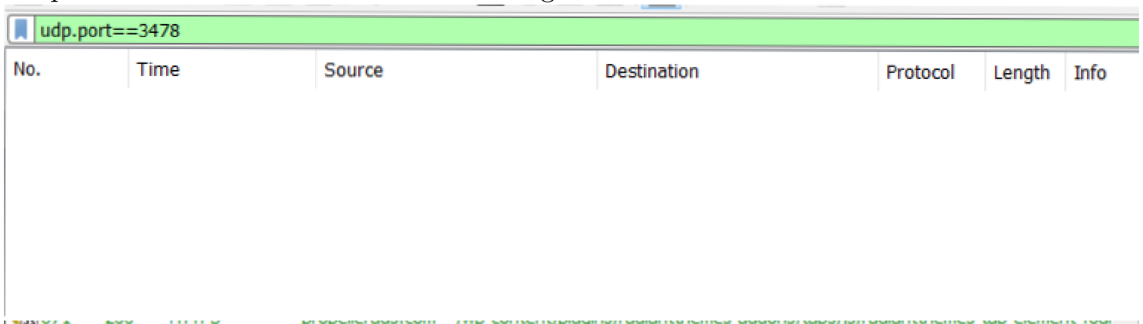
Popular IM applications like Telegram, WhatsApp, and Signal deploy encryption (either end-to-end or end-to-middle). WhatsApp calls are using only UDP and TCP, an analysis about the collected traffic can be collected and categorized. [SSA19]

We used WhatsApp online app and windows version and we tracked the both on Wireshark and Fiddler. The paper has identified TCP ports 443, 4244, 5222, 5223, 5228, 5242 and UDP ports 3478. We have used Fiddler Classic packet sniffer to obtain the SSL handshake for WhatsApp at TCP port 443. We have obtain the same results on Wireshark. UDP packets always contact STUN servers which are:

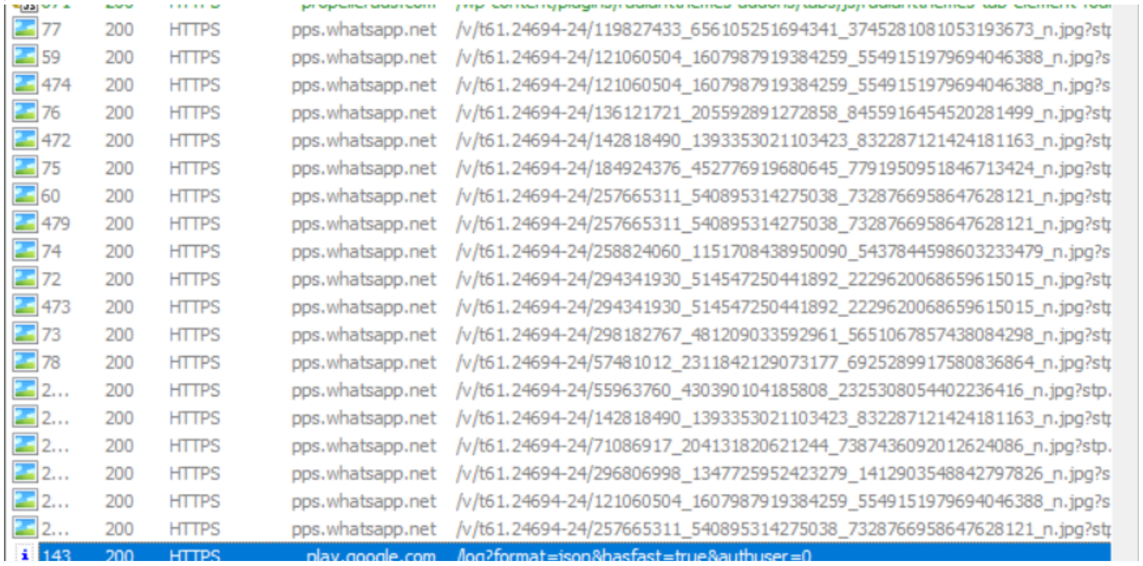
- 31.13.78.51
- 31.13.79.52
- 157.240.7.51
- 157.240.13.51
- 157.240.16.51
- 157.240.23.52

3.1.2 Surfshark Results

No packets were found for UDP after using Wireshark.

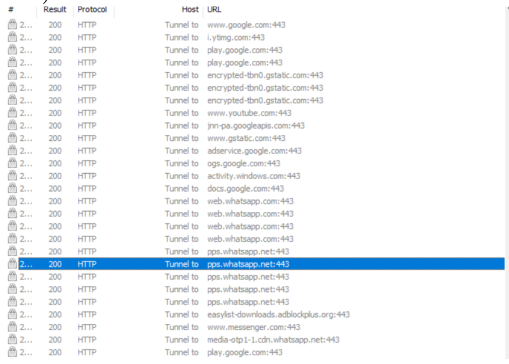


```
udp.port==3478
No.     Time     Source     Destination     Protocol     Length     Info
```

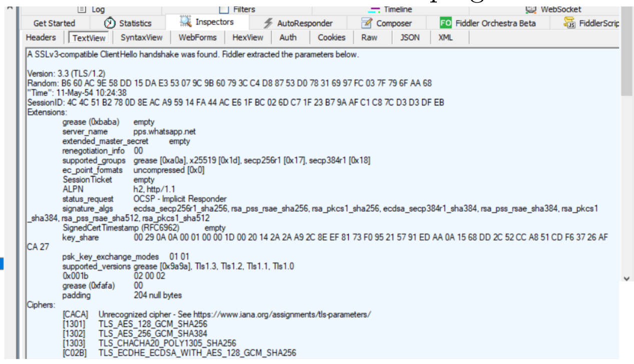


```
77   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/119827433_656105251694341_374528108105319367З_n.jpg?st
59   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/121060504_1607987919384259_5549151979694046388_n.jpg?s
474  200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/121060504_1607987919384259_5549151979694046388_n.jpg?s
76   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/136121721_205592891272858_8455916454520281499_n.jpg?st
472  200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/142818490_1393353021103423_832287121424181163_n.jpg?st
75   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/184924376_452776919680645_7791950951846713424_n.jpg?st
60   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/257665311_540895314275038_7328766958647628121_n.jpg?st
479  200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/257665311_540895314275038_7328766958647628121_n.jpg?st
74   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/258824060_1151708438950090_5437844598603233479_n.jpg?s
72   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/294341930_514547250441892_2229620068659615015_n.jpg?st
473  200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/294341930_514547250441892_2229620068659615015_n.jpg?st
73   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/298182767_481209033592961_5651067857438084298_n.jpg?st
78   200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/57481012_2311842129073177_6925289917580836864_n.jpg?st
2... 200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/55963760_430390104185808_2325308054402236416_n.jpg?stp.
2... 200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/142818490_1393353021103423_832287121424181163_n.jpg?st
2... 200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/71086917_204131820621244_7387436092012624086_n.jpg?stp.
2... 200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/296806998_1347725952423279_1412903548842797826_n.jpg?s
2... 200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/121060504_1607987919384259_5549151979694046388_n.jpg?s
2... 200  HTTPS  pps.whatsapp.net  /v/t61.24694-24/257665311_540895314275038_7328766958647628121_n.jpg?st
143  200  HTTPS  play.google.com   /log?format=json&hasfast=true&authuser=0
```

We have listen to port 443 on TCP using Fiddler and found the handshake for a WhatsApp conversation. It used to be possible to track a conversation and decrypt messages if one of the recipients used their secret in a Wireshark plugin located at this address: https://github.com/davidgfnet/wireshark-whatsapp We need a lower android version (under 7.0) and a functional older version of Wireshark that will work with the plugin.



```
#     Result  Protocol       Host       URL
2...  200     HTTP      Tunnel to  www.google.com:443
2...  200     HTTP      Tunnel to  i.ytimg.com:443
2...  200     HTTP      Tunnel to  play.google.com:443
2...  200     HTTP      Tunnel to  play.google.com:443
2...  200     HTTP      Tunnel to  encrypted-tbn0.gstatic.com:443
2...  200     HTTP      Tunnel to  encrypted-tbn0.gstatic.com:443
2...  200     HTTP      Tunnel to  encrypted-tbn0.gstatic.com:443
2...  200     HTTP      Tunnel to  www.youtube.com:443
2...  200     HTTP      Tunnel to  jnn-pa.googleapis.com:443
2...  200     HTTP      Tunnel to  www.gstatic.com:443
2...  200     HTTP      Tunnel to  adservice.google.com:443
2...  200     HTTP      Tunnel to  ogs.google.com:443
2...  200     HTTP      Tunnel to  activity.windows.com:443
2...  200     HTTP      Tunnel to  docs.google.com:443
2...  200     HTTP      Tunnel to  web.whatsapp.com:443
2...  200     HTTP      Tunnel to  web.whatsapp.com:443
2...  200     HTTP      Tunnel to  web.whatsapp.com:443
2...  200     HTTP      Tunnel to  web.whatsapp.com:443
2...  200     HTTP      Tunnel to  pps.whatsapp.net:443
2...  200     HTTP      Tunnel to  pps.whatsapp.net:443
2...  200     HTTP      Tunnel to  pps.whatsapp.net:443
2...  200     HTTP      Tunnel to  pps.whatsapp.net:443
2...  200     HTTP      Tunnel to  pps.whatsapp.net:443
2...  200     HTTP      Tunnel to  easylist-downloads.adblockplus.org:443
2...  200     HTTP      Tunnel to  www.messenger.com:443
2...  200     HTTP      Tunnel to  media-otp1-1.cdn.whatsapp.net:443
2...  200     HTTP      Tunnel to  play.google.com:443
```



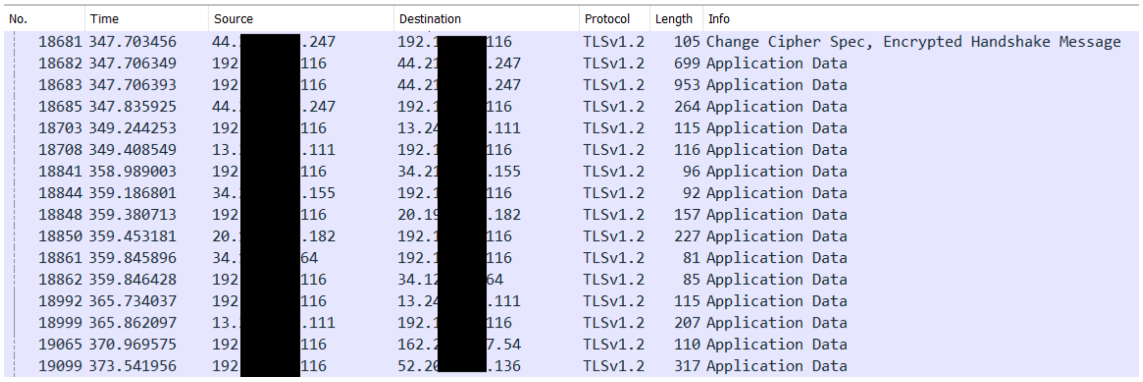
```
A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below.

Version: 3.3 (TLS/1.2)
Random: B6 60 AC 9E 58 DD 15 DA E3 53 07 9C 9B 60 79 3C C4 D8 87 53 D0 78 31 69 97 FC 03 7F 79 6F AA 68
"Time": 11-May-54 10:24:38
SessionID: 4C 4C 51 B2 78 0D 8E AC A9 59 14 FA 44 AC E6 1F BC 02 6D C7 1F 23 B7 9A AF C1 C8 7C D3 D3 DF EB
Extensions:
        grease (0xbaba)     empty
        server_name         pps.whatsapp.net
        extended_master_secret      empty
        renegotiation_info  00
        supported_groups    grease [0xa0a], x25519 [0x1d], secp256r1 [0x17], secp384r1 [0x18]
        ec_point_formats    uncompressed [0x0]
        SessionTicket       empty
        ALPN                h2, http/1.1
        status_request      OCSP - Implicit Responder
        signature_algs      ecdsa_secp256r1_sha256, rsa_pss_rsae_sha256, rsa_pkcs1_sha256, ecdsa_secp384r1_sha384, rsa_pss_rsae_sha384, rsa_pkcs1
_sha384, rsa_pss_rsae_sha512, rsa_pkcs1_sha512
        SignedCertTimestamp (RFC6962)       empty
        key_share           00 29 0A 0A 00 01 00 00 1D 00 20 14 2A 2A A9 2C 8E EF 81 73 F0 95 21 57 91 ED AA 0A 15 68 DD 2C 52 CC A8 51 CD F6 37 26 AF
CA 27
        psk_key_exchange_modes    01 01
        supported_versions grease [0x9a9a], Tls1.3, Tls1.2, Tls1.1, Tls1.0
        0x001b              02 00 02
        grease (0xfafa)     00
        padding             204 null bytes
Ciphers:
        [CACA]    Unrecognized cipher - See https://www.iana.org/assignments/tls-parameters/
        [1301]    TLS_AES_128_GCM_SHA256
        [1302]    TLS_AES_256_GCM_SHA384
        [1303]    TLS_CHACHA20_POLY1305_SHA256
        [C02B]    TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
```

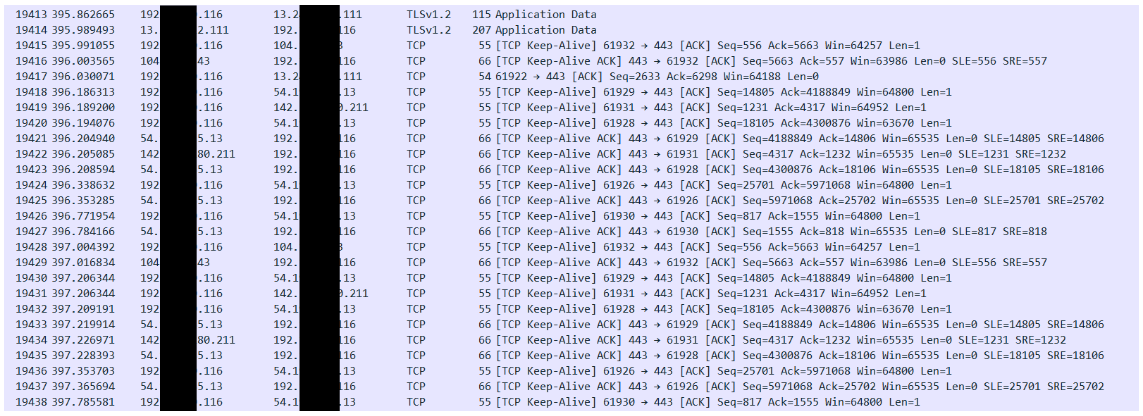
3.1.3 Signal

Traffic analysis of the Signal app can be tried by incorporating the firewall approach for the investigation. The firewall helps to understand the pattern of connectivity and communication activities, forcing the Signal client to connect to its server in a controlled environment to reveal the obscured design of Signal app. [AHS+21]

3.1.4 Surfshark Results



```
No.     Time        Source               Destination          Protocol Length Info
18681 347.703456   44.[redacted].247    192.1[redacted]116    TLSv1.2  105 Change Cipher Spec, Encrypted Handshake Message
18682 347.706349   192[redacted]116     44.21[redacted].247   TLSv1.2  699 Application Data
18683 347.706393   192[redacted]116     44.21[redacted].247   TLSv1.2  953 Application Data
18685 347.835925   44.[redacted].247    192.1[redacted]116    TLSv1.2  264 Application Data
18703 349.244253   192[redacted]116     13.24[redacted].111   TLSv1.2  115 Application Data
18708 349.408549   13.[redacted].111    192.1[redacted]116    TLSv1.2  116 Application Data
18841 358.989003   192[redacted]116     34.21[redacted].155   TLSv1.2   96 Application Data
18844 359.186801   34.[redacted].155    192.1[redacted]116    TLSv1.2   92 Application Data
18848 359.380713   192[redacted]116     20.19[redacted].182   TLSv1.2  157 Application Data
18850 359.453181   20.[redacted].182    192.1[redacted]116    TLSv1.2  227 Application Data
18861 359.845896   34.[redacted]64      192.1[redacted]116    TLSv1.2   81 Application Data
18862 359.846428   192[redacted]116     34.12[redacted]64     TLSv1.2   85 Application Data
18992 365.734037   192[redacted]116     13.24[redacted].111   TLSv1.2  115 Application Data
18999 365.862097   13.[redacted].111    192.1[redacted]116    TLSv1.2  207 Application Data
19065 370.969575   192[redacted]116     162.2[redacted]7.54   TLSv1.2  110 Application Data
19099 373.541956   192[redacted]116     52.20[redacted].136   TLSv1.2  317 Application Data
```



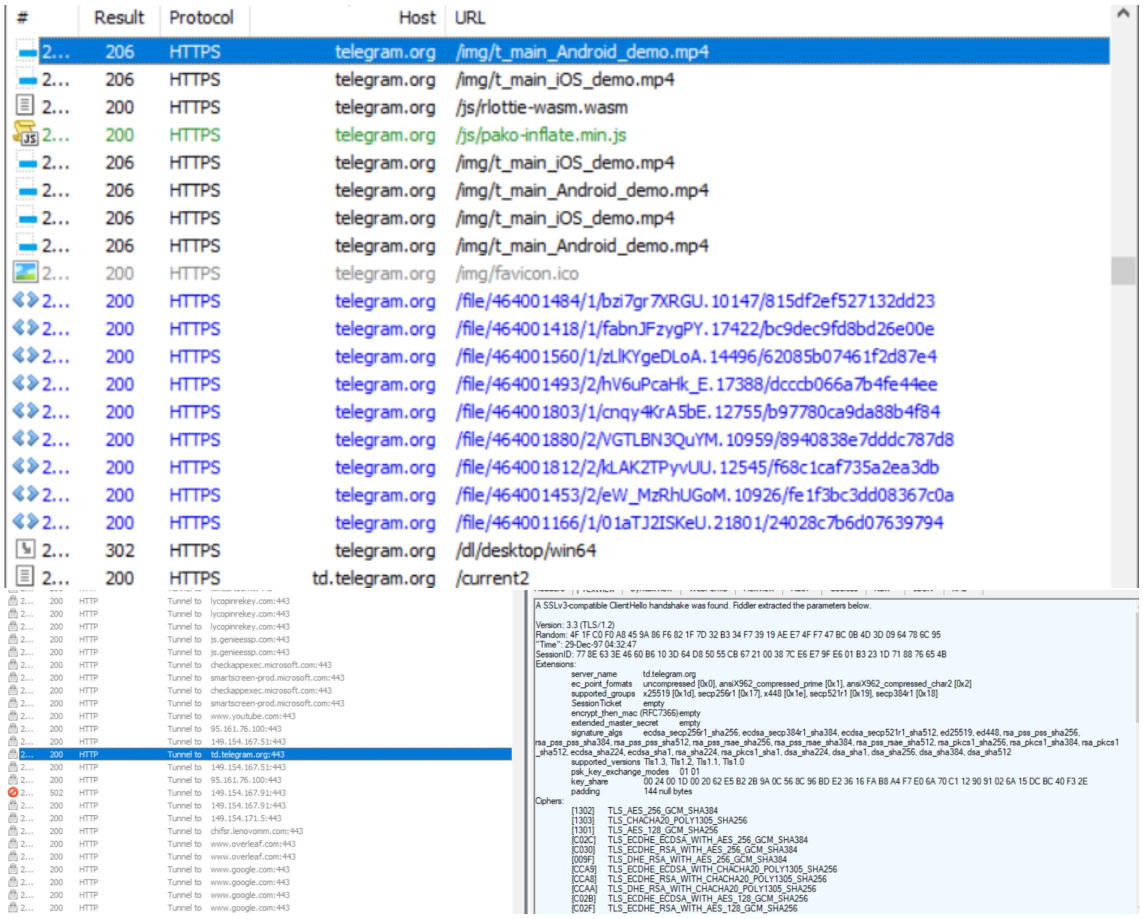
```
19413 395.862665  192[redacted].116     13.2[redacted].111   TLSv1.2  115 Application Data
19414 395.989493  13.[redacted]2.111    192.[redacted]116    TLSv1.2  207 Application Data
19415 395.991055  192[redacted].116     104.[redacted]3      TCP       55 [TCP Keep-Alive] 61932 → 443 [ACK] Seq=556 Ack=5663 Win=64257 Len=1
19416 396.003565  104[redacted]43       192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61932 [ACK] Seq=5663 Ack=557 Win=63986 Len=0 SLE=556 SRE=557
19417 396.030071  192[redacted].116     13.2[redacted].111   TCP       54 61922 → 443 [ACK] Seq=2633 Ack=6298 Win=64188 Len=0
19418 396.186313  192[redacted].116     54.1[redacted].13    TCP       55 [TCP Keep-Alive] 61929 → 443 [ACK] Seq=14805 Ack=4188849 Win=64800 Len=1
19419 396.189200  192[redacted].116     142.[redacted].211   TCP       55 [TCP Keep-Alive] 61931 → 443 [ACK] Seq=1231 Ack=4317 Win=64952 Len=1
19420 396.194076  192[redacted].116     54.1[redacted].13    TCP       55 [TCP Keep-Alive] 61928 → 443 [ACK] Seq=18105 Ack=4300876 Win=63670 Len=1
19421 396.204940  54.[redacted]5.13     192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61929 [ACK] Seq=4188849 Ack=14806 Win=65535 Len=0 SLE=14805 SRE=14806
19422 396.205085  142[redacted]80.211   192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61931 [ACK] Seq=4317 Ack=1232 Win=65535 Len=0 SLE=1231 SRE=1232
19423 396.208594  54.[redacted]5.13     192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61928 [ACK] Seq=4300876 Ack=18106 Win=65535 Len=0 SLE=18105 SRE=18106
19424 396.338632  192[redacted].116     54.1[redacted].13    TCP       55 [TCP Keep-Alive] 61926 → 443 [ACK] Seq=25701 Ack=5971068 Win=64800 Len=1
19425 396.353285  54.[redacted]5.13     192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61926 [ACK] Seq=5971068 Ack=25702 Win=65535 Len=0 SLE=25701 SRE=25702
19426 396.771954  192[redacted].116     54.1[redacted].13    TCP       55 [TCP Keep-Alive] 61930 → 443 [ACK] Seq=817 Ack=1555 Win=64800 Len=1
19427 396.784166  54.[redacted]5.13     192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61930 [ACK] Seq=1555 Ack=818 Win=65535 Len=0 SLE=817 SRE=818
19428 397.004392  192[redacted].116     104.[redacted]3      TCP       55 [TCP Keep-Alive] 61932 → 443 [ACK] Seq=556 Ack=5663 Win=64257 Len=1
19429 397.016834  104[redacted]43       192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61932 [ACK] Seq=5663 Ack=557 Win=63986 Len=0 SLE=556 SRE=557
19430 397.206344  192[redacted].116     54.1[redacted].13    TCP       55 [TCP Keep-Alive] 61929 → 443 [ACK] Seq=14805 Ack=4188849 Win=64800 Len=1
19431 397.206344  192[redacted].116     142.[redacted].211   TCP       55 [TCP Keep-Alive] 61931 → 443 [ACK] Seq=1231 Ack=4317 Win=64952 Len=1
19432 397.209191  192[redacted].116     54.1[redacted].13    TCP       55 [TCP Keep-Alive] 61928 → 443 [ACK] Seq=18105 Ack=4300876 Win=63670 Len=1
19433 397.219914  54.[redacted]5.13     192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61929 [ACK] Seq=4188849 Ack=14806 Win=65535 Len=0 SLE=14805 SRE=14806
19434 397.226971  142[redacted]80.211   192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61931 [ACK] Seq=4317 Ack=1232 Win=65535 Len=0 SLE=1231 SRE=1232
19435 397.228393  54.[redacted]5.13     192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61928 [ACK] Seq=4300876 Ack=18106 Win=65535 Len=0 SLE=18105 SRE=18106
19436 397.353703  192[redacted].116     54.1[redacted].13    TCP       55 [TCP Keep-Alive] 61926 → 443 [ACK] Seq=25701 Ack=5971068 Win=64800 Len=1
19437 397.365694  54.[redacted]5.13     192.[redacted]116    TCP       66 [TCP Keep-Alive ACK] 443 → 61926 [ACK] Seq=5971068 Ack=25702 Win=65535 Len=0 SLE=25701 SRE=25702
19438 397.785581  192[redacted].116     54.1[redacted].13    TCP       55 [TCP Keep-Alive] 61930 → 443 [ACK] Seq=817 Ack=1555 Win=64800 Len=1
```

3.1.5 Telegram

Specifically, we will talk about surveillance parties that are capable of identifying members of target instant messaging communications (e.g., politically sensitive IM channels) with very high accuracies, and by only using low-cost traffic analysis techniques. [BSH+20]

3.1.6 Surfshark Results

We succeeded in capturing the SSL handshake and public images from telegram. It uses the same port as Whatsapp on TCP:443.



```
#     Result  Protocol             Host  URL
2...  206     HTTPS         telegram.org  /img/t_main_Android_demo.mp4
2...  206     HTTPS         telegram.org  /img/t_main_iOS_demo.mp4
2...  200     HTTPS         telegram.org  /js/rlottie-wasm.wasm
2...  200     HTTPS         telegram.org  /js/pako-inflate.min.js
2...  206     HTTPS         telegram.org  /img/t_main_iOS_demo.mp4
2...  206     HTTPS         telegram.org  /img/t_main_Android_demo.mp4
2...  206     HTTPS         telegram.org  /img/t_main_iOS_demo.mp4
2...  206     HTTPS         telegram.org  /img/t_main_Android_demo.mp4
2...  200     HTTPS         telegram.org  /img/favicon.ico
2...  200     HTTPS         telegram.org  /file/464001484/1/bzi7gr7XRGU.10147/815df2ef527132dd23
2...  200     HTTPS         telegram.org  /file/464001418/1/fabnJFzygPY.17422/bc9dec9fd8bd26e00e
2...  200     HTTPS         telegram.org  /file/464001560/1/zLlKYgeDLoA.14496/62085b07461f2d87e4
2...  200     HTTPS         telegram.org  /file/464001493/2/hV6uPcaHk_E.17388/dcccb066a7b4fe44ee
2...  200     HTTPS         telegram.org  /file/464001803/1/cnqy4KrA5bE.12755/b97780ca9da88b4f84
2...  200     HTTPS         telegram.org  /file/464001880/2/VGTLBN3QuYM.10959/8940838e7dddc787d8
2...  200     HTTPS         telegram.org  /file/464001812/2/kLAK2TPyvUU.12545/f68c1caf735a2ea3db
2...  200     HTTPS         telegram.org  /file/464001453/2/eW_MzRhUGoM.10926/fe1f3bc3dd08367c0a
2...  200     HTTPS         telegram.org  /file/464001166/1/01aTJ2ISKeU.21801/24028c7b6d07639794
2...  302     HTTPS         telegram.org  /dl/desktop/win64
2...  200     HTTPS      td.telegram.org  /current2
```

```
2...  200  HTTP  Tunnel to  lycopinrekey.com:443
2...  200  HTTP  Tunnel to  lycopinrekey.com:443
2...  200  HTTP  Tunnel to  lycopinrekey.com:443
2...  200  HTTP  Tunnel to  js.genieessp.com:443
2...  200  HTTP  Tunnel to  js.genieessp.com:443
2...  200  HTTP  Tunnel to  checkappexec.microsoft.com:443
2...  200  HTTP  Tunnel to  smartscreen-prod.microsoft.com:443
2...  200  HTTP  Tunnel to  checkappexec.microsoft.com:443
2...  200  HTTP  Tunnel to  smartscreen-prod.microsoft.com:443
2...  200  HTTP  Tunnel to  www.youtube.com:443
2...  200  HTTP  Tunnel to  95.161.76.100:443
2...  200  HTTP  Tunnel to  149.154.167.51:443
2...  200  HTTP  Tunnel to  td.telegram.org:443
2...  200  HTTP  Tunnel to  149.154.167.51:443
2...  200  HTTP  Tunnel to  95.161.76.100:443
2...  502  HTTP  Tunnel to  149.154.167.91:443
2...  200  HTTP  Tunnel to  149.154.167.91:443
2...  200  HTTP  Tunnel to  149.154.171.5:443
2...  200  HTTP  Tunnel to  chifsr.lenovomm.com:443
2...  200  HTTP  Tunnel to  www.overleaf.com:443
2...  200  HTTP  Tunnel to  www.overleaf.com:443
2...  200  HTTP  Tunnel to  www.google.com:443
2...  200  HTTP  Tunnel to  www.google.com:443
2...  200  HTTP  Tunnel to  www.google.com:443
2...  200  HTTP  Tunnel to  www.google.com:443
```

```
A SSLv3-compatible ClientHello handshake was found. Fiddler extracted the parameters below.

Version: 3.3 (TLS/1.2)
Random: 4F 1F C0 F0 A8 45 9A 86 F6 82 1F 7D 32 B3 34 F7 39 19 AE E7 4F F7 47 BC 0B 4D 3D 09 64 78 6C 95
"Time": 29-Dec-97 04:32:47
SessionID: 77 8E 63 3E 46 60 B6 10 3D 64 D8 50 55 CB 67 21 00 38 7C E6 E7 9F E6 01 B3 23 1D 71 88 76 65 4B
Extensions:
        server_name         td.telegram.org
        ec_point_formats    uncompressed [0x0], ansiX962_compressed_prime [0x1], ansiX962_compressed_char2 [0x2]
        supported_groups    x25519 [0x1d], secp256r1 [0x17], x448 [0x1e], secp521r1 [0x19], secp384r1 [0x18]
        SessionTicket       empty
        encrypt_then_mac (RFC7366)empty
        extended_master_secret      empty
        signature_algs      ecdsa_secp256r1_sha256, ecdsa_secp384r1_sha384, ecdsa_secp521r1_sha512, ed25519, ed448, rsa_pss_pss_sha256,
rsa_pss_pss_sha384, rsa_pss_pss_sha512, rsa_pss_rsae_sha256, rsa_pss_rsae_sha384, rsa_pss_rsae_sha512, rsa_pkcs1_sha256, rsa_pkcs1_sha384, rsa_pkcs1
_sha512, ecdsa_sha224, ecdsa_sha1, rsa_sha224, rsa_pkcs1_sha1, dsa_sha224, dsa_sha1, dsa_sha256, dsa_sha384, dsa_sha512
        supported_versions Tls1.3, Tls1.2, Tls1.1, Tls1.0
        psk_key_exchange_modes      01 01
        key_share           00 24 00 1D 00 20 62 E5 B2 2B 9A 0C 56 8C 96 BD E2 36 16 FA B8 A4 F7 E0 6A 70 C1 12 90 91 02 6A 15 DC BC 40 F3 2E
        padding             144 null bytes
Ciphers:
        [1302]    TLS_AES_256_GCM_SHA384
        [1303]    TLS_CHACHA20_POLY1305_SHA256
        [1301]    TLS_AES_128_GCM_SHA256
        [C02C]    TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        [C030]    TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
        [009F]    TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
        [CCA9]    TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
        [CCA8]    TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
        [CCAA]    TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
        [C02B]    TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        [C02F]    TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
```

4 Conclusion

In summary, Signal is generally considered to have the strongest security features and encryption among the three, followed by WhatsApp, and Telegram. However, it's important to consider other factors such as user base, data collection and usage policies, and other feature when making a decision on which app to use.

References

[ACG17] Cosimo Anglano, Massimo Canonico, and Marco Guazzone. Forensic analysis of telegram messenger on android smartphones. *Digital Investigation*, 23:31–49, 2017.

[AHS+21] Asmara Afzal, Mehdi Hussain, Shahzad Saleem, M Khuram Shahzad, Anthony TS Ho, and Ki-Hyun Jung. Encrypted network traffic analysis of secure instant messaging application: A case study of signal messenger app. *Applied Sciences*, 11(17):7789, 2021.

[BSH+20] Alireza Bahramali, Ramin Soltani, Amir Houmansadr, Dennis Goeckel, and Don Towsley. Practical traffic analysis attacks on secure messaging applications. *arXiv preprint arXiv:2005.00508*, 2020.

[com16] Cometchat Blog. https://www.cometchat.com/blog/xmpp-extensible-messaging-presence-protocol, 2016.

[CVE22a] Signal Security vulnerabilities. https://www.cvedetails.com/vulnerability-list/vendor_id-17912/Signal.html/, 2022.

[CVE22b] Telegram Security vulnerabilities. https://www.cvedetails.com/vulnerability-list/vendor_id-16210/Telegram.html, 2022.

[CVE22c] Whatsapp Security vulnerabilities. https://www.cvedetails.com/product/54433/Whatsapp-Whatsapp.html?, 2022.

[cyb22] Cybernews. https://cybernews.com/news/whatsapp-data-leak/, 2022.

[Jud18] Samantha M Judge. *Mobile Forensics: Analysis of the Messaging Application Signal*. University of Central Oklahoma, 2018.

[mtp] MTProto. https://core.telegram.org/mtproto.

[nis21] NIST standards. http://www.nist.gov/,http://www.csrc.nist.gov/, 2021.

[Rin21] RingCentral. End-to-end Encryption. https://www.ringcentral.com/us/en/blog/dynamic-end-to-end-encryption/, 2021.

[sig16] Signal Documentation. https://signal.org/docs/specifications/doubleratchet/, 2016.

[SSA19] C Shubha, SA Sushma, and KH Asha. Traffic analysis of whatsapp calls. In *2019 1st International Conference on Advances in Information Technology (ICAIT)*, pages 256–260. IEEE, 2019.

[Tha13] Neha S Thakur. Forensic analysis of whatsapp on android smartphones. 2013.

View publication stats