applied sciences

MDPI

Article

Encrypted Network Traffic Analysis of Secure Instant Messaging Application: A Case Study of Signal Messenger App

Asmara Afzal [1], Mehdi Hussain [1,*], Shahzad Saleem [1,2], M. Khuram Shahzad [1], Anthony T. S. Ho [3,4] and Ki-Hyun Jung [5,*]

1 Department of Computing, School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad 44000, Pakistan; aafzal.msis17seecs@seecs.edu.pk (A.A.); shahzad.saleem@seecs.edu.pk (S.S.); mkhurram.shahzad@seecs.edu.pk (M.K.S.)
2 Department of Cybersecurity, College of Computer Science and Engineering, University of Jeddah, Jeddah 23218, Saudi Arabia
3 Department of Computer Science, University of Surrey, Guildford GU27XH, UK; a.ho@surrey.ac.uk
4 Department of Computer Science and Technology, Tianjin University of Science and Technology, Tianjin 300222, China
5 Department of Cyber Security, Kyungil University, Gyeongbuk 38424, Korea
* Correspondence: mehdi.hussain@seecs.edu.pk (M.H.); kingjung@kiu.kr (K.-H.J.)

Abstract: Instant messaging applications (apps) have played a vital role in online interaction, especially under COVID-19 lockdown protocols. Apps with security provisions are able to provide confidentiality through end-to-end encryption. Ill-intentioned individuals and groups use these security services to their advantage by using the apps for criminal, illicit, or fraudulent activities. During an investigation, the provision of end-to-end encryption in apps increases the complexity for digital forensics investigators. This study aims to provide a network forensic strategy to identify the potential artifacts from the encrypted network traffic of the prominent social messenger app Signal (on Android version 9). The analysis of the installed app was conducted over fully encrypted network traffic. By adopting the proposed strategy, the forensic investigator can easily detect encrypted traffic activities such as chatting, media messages, audio, and video calls by looking at the payload patterns. Furthermore, a detailed analysis of the trace files can help to create a list of chat servers and IP addresses of involved parties in the events. As a result, the proposed strategy significantly facilitates extraction of the app's behavior from encrypted network traffic which can then be used as supportive evidence for forensic investigation.

Keywords: network forensics; encrypted traffic forensic; signal application; Wireshark; trace files

Citation: Afzal, A.; Hussain, M.; Saleem, S.; Shahzad, M.K.; Ho, A.T.S.; Jung, K.-H. Encrypted Network Traffic Analysis of Secure Instant Messaging Application: A Case Study of Signal Messenger App. *Appl. Sci.* **2021**, *11*, 7789. https://doi.org/10.3390/app11177789

Academic Editor: Cheonshik Kim

Received: 14 July 2021
Accepted: 18 August 2021
Published: 24 August 2021

1. Introduction

End-to-end encryption based instant messaging (IM) apps have captured the attention of users due to increasing security and privacy concerns. Owing to this, organizations and individuals have adopted various types of IM platforms for their regular communication. Further, the salient features of IM apps such as convenience, speed, immediate receipt, acknowledgment, and reply to messages attract people of all ages. Similarly, organizations enjoy inexpensive and effortless marketing/advertising opportunities by using these platforms [1]. According to Juniper research [2], IM users substantially reach 4.3 billion in 2020 which is a yearly increase of approximately 9%. The COVID-19 pandemic has also increased the use of IM applications due to work from home policies in most organizations [3]. Hence, it is noteworthy that IM-based apps are performing a vital role in today's communications.

Common IM apps such as Signal, WhatsApp, Facebook Messenger, WeChat, and many more, employ end-to-end encryption techniques during transmission to provide security and privacy to user data. The Signal app claims to use the secure protocol during

communication, i.e., Signal Protocol. Malicious actors are also taking advantage of the end-to-end encryption feature of IM apps. Thus, the provision of security features (end-to-end encryption) in these apps provide an attractive communication medium for digital crime or fraudulent activities. Therefore, forensic investigation of digital crimes may increase effort and high possibility of failure scenarios due to end-to-end encryption services in IM apps [4].

For investigation purposes, device and network forensics strategies need to be followed. Device forensics helps to determine the file structure and extraction of useful information stored on the devices. In literature, numerous strategies of device forensics based on social media apps have already been proposed and are also useful, and have their merits and demerits [5–11]. Some of them also investigated encrypted databases and file structures to identify the evidence [12,13]. However, device forensic strategies cannot be fully utilized in the case of end-to-end encryption on the user's or the app's data [5,7,10,11,13–15]. It was also observed during the literature review phase that IM applications are extensively analyzed but limited to the device's internal structure of databases.

On the other hand, network forensics is a type of forensic analysis where the reconstruction of events is performed after an incident or cybercrime. In other words, a major concern is tracing communications through the network (live/backup) packets, i.e., to identify the traces of the apps' or users' behavior, data breaches, and other illicit activities. Thus, encrypted network traffic analysis uncovers the network-related artifacts which can be beneficial to the forensic investigator and are shown great interest [16,17]. However, the provision of end-to-end encryption protects the data during transmission. This becomes a challenge for the forensic investigator while identifying the behaviors of apps' or user's data from encrypted traffic. In literature, there are limited encrypted network forensics strategies proposed, especially on an IM-based app [8,18–21] that can be used to find some important events such as connection establishment, an encryption protocol, and payload sizes for analysis [21]. Moreover, these can also identify the basic user's or app's behavior, i.e., calling, texting, etc. [8,18,19]. However, there is still room to explore the characteristics of encrypted IM app-based packets to identify the apps' or user's behavior and employed clients' and servers' IPs to support forensic investigation.

In this research, we studied the popular IM Signal app. In the case of the Signal app, the communication data are (end-to-end) encrypted, owing to which it becomes difficult to investigate or often impossible to obtain the corresponding plain text as the keys are unknown. Therefore, in the proposed strategy we successfully demonstrated how artifacts of potential value can be extracted from an encrypted network traffic analysis of the Signal app.

The motivation behind choosing the Signal app is that it is a popular cross-platform encrypted messaging service developed by the Signal Foundation (previously known as Open Whisper Systems) powered by the open-source Signal Protocol (2013) and is available for android, IOS, and desktop. It consists of common features such as audio calls, video calls, voice messages, text, media messages, stickers, typing indicators, and many more. Each conversation has a safety number that can be verified among the communicating parties. Signal application encrypts all the messages using the Signal Protocol that is based on secure cryptographic algorithms. Recently, well-known IM apps, such as WhatsApp, Facebook Messenger, and Skype (in private conversation mode) employed the Signal Protocol to achieve secrecy and privacy. The protocol consists of a Double Ratchet algorithm, XEdDSA, and VXEdDSA, X3DH, and Sesame. XEdDSA creates a single key pair that is used in the elliptic curve Diffie–Hellman and signatures. VXEdDSA is an extension of XEdDSA to make it a verifiable random function (VRF) [22]. X3DH (extended Diffie–Hellman) is used to mutually authenticate the users and provides forward secrecy [23]. After sharing a common secret key, the messages are communicated between users by using the Double Ratchet encryption algorithm [24]. For session management of messages, Sesame (session management for asynchronous) is used with the Double Ratchet

algorithm and X3DH [25]. Owing to the provision of security services, the app has been endorsed by technologists and cryptographers [26].

We performed an extensive traffic analysis of the Signal app while incorporating the firewall approach for the investigation. The firewall helps to understand the pattern of connectivity and communication activities. We thus forced the Signal client to connect to its server in a controlled environment and this arrangement revealed the obscured design of Signal app. Further, we monitored the live traffic to dissect the bytes, and payload-based patterns to identify the different activities of the Signal app, i.e., calls, text, typing indicator, etc. The major steps of the proposed strategy that are employed to analyze the encrypted network traffic on IM-based apps are shown in Figure 1.

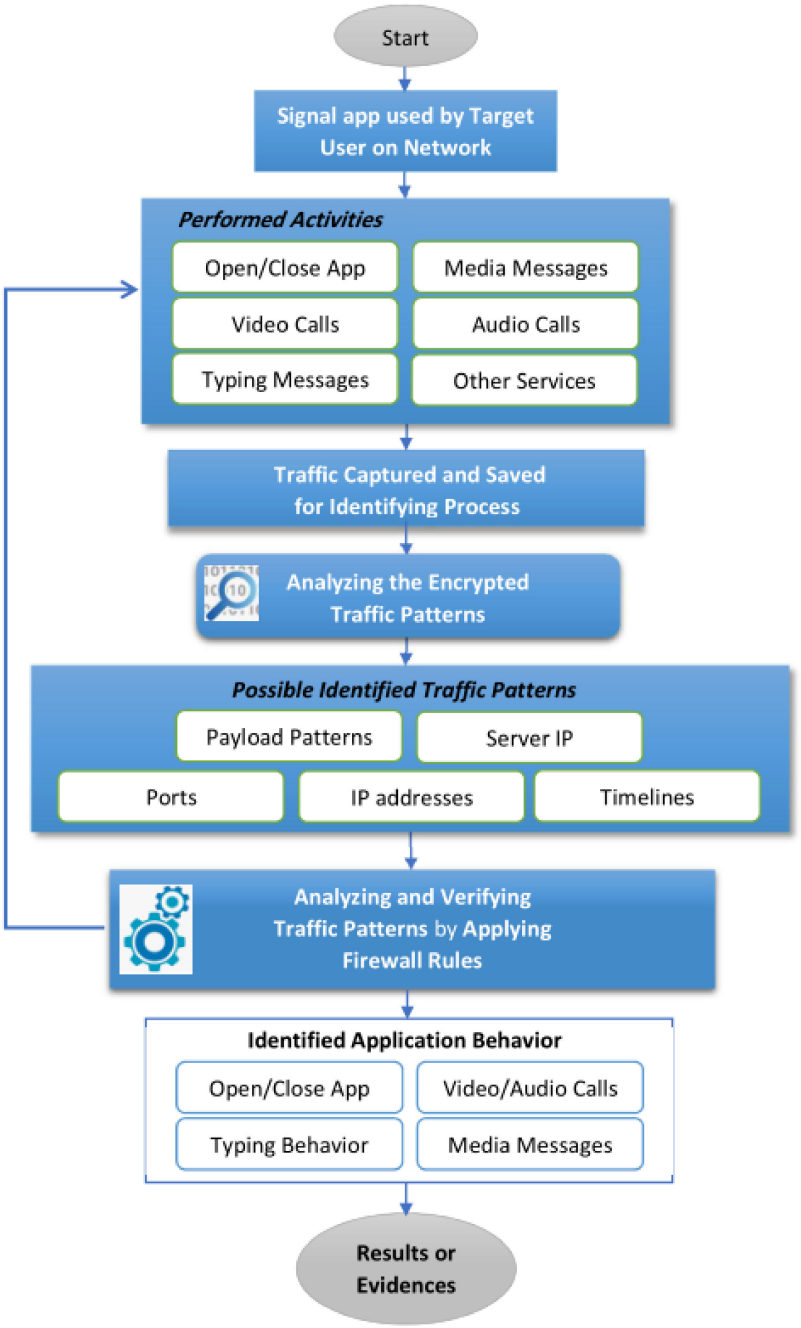


Figure 1. The flow diagram of proposed strategy.

1.1. Contributions

The motivation behind this research is to demonstrate how artifacts of potential value can be extracted from the Signal app. To this end, this study makes the following novel contributions:

- The identification of artifacts generated by the Signal app from the encrypted network traffic;
- Extensive experimental analysis is performed to validate the traffic patterns of respective activities;
- The proposed strategy is applicable to block/intercept the real-time apps' activities from the live network traffic by tuning a dynamic set of firewall rules. Moreover, the proposed strategy is 100% applicable on Android devices.

1.2. Organization

The remaining study is structured as follows: Section 2 sheds light on previous work performed related to the forensics analysis of social networking apps. The experimental setup is discussed in Section 3. Section 4 showcases the results of the experiments. This section also discusses the performed activities and their corresponding network communication patterns. Section 5 shows the discussion and use cases of the proposed encrypted network forensic strategy based on various use cases, i.e., crime scene reconstruction with a case study. Section 6 describes the benefits of the proposed strategy. Section 7 describes the limitation of the proposed strategy with future work. Lastly, Section 8 sheds light on the conclusion.

2. Literature Review

Owing to their extensive use and popularity, social media or instant messaging apps have previously been a subject of study from a device and network forensic perspective. Regarding the device forensic aspect, a study carried out by Zhang et al. [5] showed that database artifacts of popular social media apps such as Messenger, Hangouts and Line were unencrypted, unlike WhatsApp which keeps it encrypted. After rooting the android device, the chat messages, timestamps, and contact lists were found in files. However, network forensics was not performed on these apps. Similarly, Awan [27] performed forensic analysis of Facebook, Twitter, and LinkedIn on four different platforms, i.e., IOS, Android, Windows, and Blackberry. The study revealed many important artifacts that can be recovered from device memory except the Blackberry device. The extracted artifacts include important locations and databases with information related to the visited profiles, names, tweets, contacts, profile ID, etc. [27].

Gregorio et al. [11] investigated the device forensics of various messaging applications such as Telegram Messenger, on the Windows Phone. This study successfully analyzed the information structure such as user data, chat, conversation, etc. However, the study is limited to only device forensics aspects. Similarly, Al-Mutawa et al. studied applications such as Facebook, Twitter, and MySpace on android, IOS, and Blackberry devices. Through forensic device analysis of these apps, it was shown that valuable data reside in the device memory which can be extracted after logical acquisition except from Blackberry-based devices [6]. He et. al. presented a study in which they identified mobile applications from encrypted network traffic. In this study they correlated multiple factors such as IP addresses, and DNS queries to reach the conclusion [28].

Recently, Knox et al. [29] targeted the Happn social dating app for forensic analysis and successfully identified various artifacts for investigation purposes. The extraction of artifacts from Happn apps poses a personal security risk and can also result in privacy violations of various natures. Recently, Choi et al. [13] analyzed the locations and file formats of personal data files in (KakaoTalk, NateOn, and QQ) instant messaging applications. They examined the encryption and decryption procedures for internal databases. This study found that some encrypted database files can successfully be recovered without requiring a user password. Meanwhile, this study also identified that QQ messenger

utilized the external server for storing its encryption key for the database files. In a similar study, Anglano et al. [7] targeted the Telegram Messenger app for analysis. The decoding and correlating information helped to reconstruct the contact list, contents of messages, and logs of voice calls. Most experiments performed in the research were performed on virtualized android smartphones. Subsets of these experiments were also performed on a physical device to validate the results. These results can help in solving the cases during forensics investigations. However, this study did not perform any network forensic analysis for end-to-end encryption-based traffic analysis.

Walnycky et al. [8] analyzed the device and network traffic of 20 applications. They were able to reconstruct the messages and collected evidence traces such as passwords, screenshots, pictures, etc. Similarly, a network forensics analysis was targeted by Yusoff et al. [30] on social networking apps, i.e., Facebook, Twitter, and Telegram on Firefox Mobile OS simulator. In this study, simulator-based generated traffic was sniffed by the Wireshark network monitoring tool. The authors successfully showed the IP addresses, ports, domains, and subdomains. However, their proposed strategy was unable to decode the different artifacts/patterns, events, and activities, etc. Recently, a device and network forensic analysis of the IMO app was reported in [18]. In this work, the authors studied the IMO app file structure for Android and IOS platforms. The proposed strategy successfully identified the pattern of activities in encrypted network traffic; however, the proposed strategy was limited to only the IMO app. In other work, Conti et al. [15] used a machine learning technique to decode the user actions based on domain filtering, packet filtering, packet intervals, and timeout. The results showed that Facebook, Gmail, and Twitter apps can be used to infer user activities. This strategy can help in learning the behavior of one or more users thereby facilitating intelligent decision making by both an adversary and security expert. The authors discuss that some countermeasures can also be taken to frustrate the attackers such as adding some padding during communication etc. which can be used to avoid the attacks [15]. Anglano et al. [14] proposed an automated tool named AnForA for Android forensic analysis. The targeted Android applications are installed on a virtualized Android device and followed by a set of activities performed to monitor the device storage artifacts. In another study, authors analyzed WeChat, Viber, WhatsApp, and Telegram apps. The results show that there are possible ways to retrieve the encrypted database files of various apps such as WeChat and Viber from a rooted phone [12]. Similarly, Wu et al. analyzed the WeChat app and explored different scenarios such as acquiring the user data and decrypting the encoded database [9]. Similarly, Anglano et al. [10] analyzed the artifacts of the WhatsApp messenger app. This study managed to decode and interpret the artifacts from WhatsApp and successfully showed the correlation of artifacts among activities. Similarly, in another study, the authors performed a network analysis of WhatsApp. This strategy managed to decrypt the network traffic and is fully capable of obtaining the forensic artifacts related to WhatsApp calling activity [19]. Similarly, a dedicated study on the Signal app to evaluate the security of its key exchange service was presented [31]. A man-in-the-middle (MITM) attack was employed to compromise the key-exchange service. In experiments, authors launched the attack on a rooted device with the installation of the Cydia Substrate [32] and SSL Trustkiller [33]. Further, a PC was designated to intercept the traffic using an MIMT proxy and provides a WLAN hotspot for smartphones. For traffic interception, a dedicated script was written while the modified/customized version of the Signal app was installed on the attacker's device to launch MITM attacks. As a result, the authors demonstrated that 21 out of 28 users failed to verify the identity of the other users by comparing encryption keys. However, the whole experiment was conducted and only applicable to an obsolete version of the Signal app.

From the aforementioned studies, most of the work focused on device forensics of social media applications with interesting artifacts. However, limited work was found on network forensic analysis especially on encrypted network traffic for instant messaging apps. A comparison of existing work can be seen in Table 1.

Table 1. A comparison of device and network forensic analysis for android apps.

Article	Device Forensic		Network Forensic	
	Location of Artifacts	Database Decoding	Packet Analysis	Extraction of IP Addresses
Anglano et al. [10]	X	✓	X	X
Walnycky et al. [8]	✓	X	X	X
Karpisek et al. [19]	X	X	✓	✓
Sudozai et al. [18]	✓	✓	✓	✓
Wu et al. [9]	✓	✓	X	X
Rathi et al. [12]	✓	✓	X	X
Shawn et al. [29]	✓	✓	X	X
Gregorio et al. [11]	✓	✓	X	X
Anglano et al. [14]	✓	✓	X	X
Choi et al. [13]	✓	✓	X	X

There is still room to study the encrypted network traffic to identify various activities of user's and app's behaviors that further can be utilized for the evidence, either in direct/indirect or supportive manners in investigation purposes. Therefore, the motivation of this study is to propose a strategy to investigate the encrypted network traffic to uncover the various patterns/artifacts. Research is demonstrated by using the Signal app as a case study.

3. Experimental Setup

To capture network traffic, network infrastructure was designed as depicted in Figure 2 inspired by [18]. The proposed experimental setup includes a firewall, an access point, and a PC to display the ongoing traffic of mobile devices passing through the network. The Signal app-based mobile device was connected to the internet through a wireless access point, and all the internet traffic of the wireless access point was routed through a PC-based pfSense firewall [34]. The role of the firewall was to monitor, sniff, and capture the entire network traffic. Thus, the firewall-generated trace files were saved for analysis. The configured firewall filtered out the Signal app-based internet traffic in a controlled environment. The access point aided to connect other devices such as mobiles and PCs with firewalls.

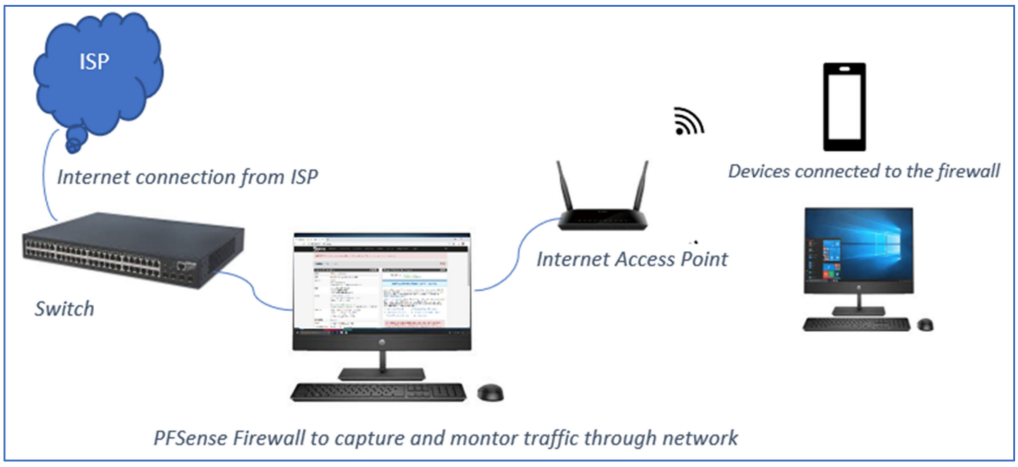


Figure 2. Experimental setup for capturing the network encrypted traffic of Signal app.

For network traffic analysis, the Wireshark software [35] was installed to monitor the encrypted traffic by analyzing the trace files. It is noteworthy that the IP addresses and ports were in plaintext while the payload was encrypted for the sake of secrecy and privacy services. The IP addresses and ports created an opportunity to determine the behaviors of apps and its various activities performed during the communication. During experiments, we collected a large no. of trace files for analysis purposes, some of them are available on [36] for better understanding. Table 2 shows the detail of the devices and tools used during experiments.

Table 2. Devices and tools specifications for experimental setup.

Device/Tool	Purpose	Company	Software/OS
Firewall	Capturing and monitoring network packets in the traffic. Firewall rules	PfSense	2.4.4-RELEASE-p3 and FreeBSD version 11.2-RELEASE-p3
CISCO switch	WAN connection	CISCO	SG300-52 port Gigabit Cisco Layer-3)
Wireless Access Point (WAP)	Used as bridge	PTCL	PTCL, DSL-2750U
Smartphone	To perform user activities on Signal app	Nokia 2.1	Android OS version 9
Desktop Computer	To deploy firewall and Wireshark	Dell	Windows 10, 64- bit Operating system that has an x64-based processor, 4GB RAM, Intel (R) Core (TM) i5-3470 CPU @3.20 GHz.
Wireshark	To read the network trace		Version 3.0.6
Signal application	To perform major activities	Signal Foundation	Version 4.58.5

To analyze the Signal app's behavior, e.g., how client and server communication works, the proposed strategy sniffed the encrypted traffic between the client and its servers, where it showed a list of different servers with IP addresses and ports. Identification of ports and IP addresses aid in imposing new rules in the network. For example, during experiments we observed that different app servers provide separate services such as text, media, and calls (Section 5). The detail of user activities such as opening the Signal app, text indicators, text messages, and audio calls are discussed in the result and analysis section.

3.1. Justification of Firewall

Placing a firewall in the investigation network allows the control to detect the app behavior effectively. Firewall rules are used to confirm the default app behavior. Furthermore, restrictions can be applied, and ulterior app behaviors can be brought to light. Default and alternate traffic patterns can assist the forensics examiners in solving the cases involving Signal or any other app. Moreover, it also helps in observing the client-server connectivity design pattern such as TCP/UDP ports and server ranges, etc. In this scenario, firewall configuration requires defining rules according to network requirements for both WAN and LAN. By default, both inbound and outbound traffic are blocked through the firewall. The LAN rules are defined to configure access to internet resources. As a standard practice, there are no rules defined for WAN NIC which means all inbound traffic are blocked unless configured. If there is any web server in a network, a WAN rule can be defined to access that server.

The *packet capture* option in the firewall interface is used to capture the live traffic of the target device as shown in Figure 3. Explicit capturing of smartphone traffic of the Signal app through a firewall or any other device in the network is possible with the help of IP addresses as shown in Figure 4. This helps to reduce the reliance on port mirroring deployed in the setup explained in the paper [18]. The captured live network traffic is saved from LAN/WAN as trace files which are then analyzed using Wireshark.

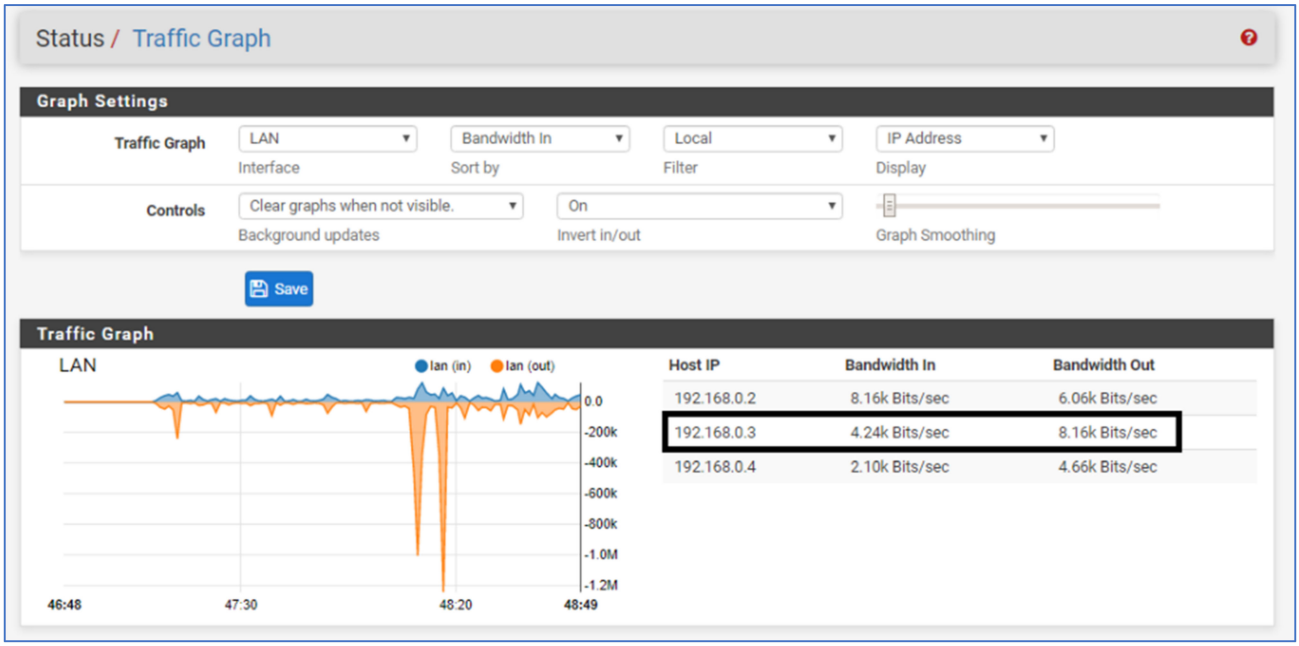


Figure 3. Screenshot of firewall (pfSense) traffic passing of smartphone with IP address.

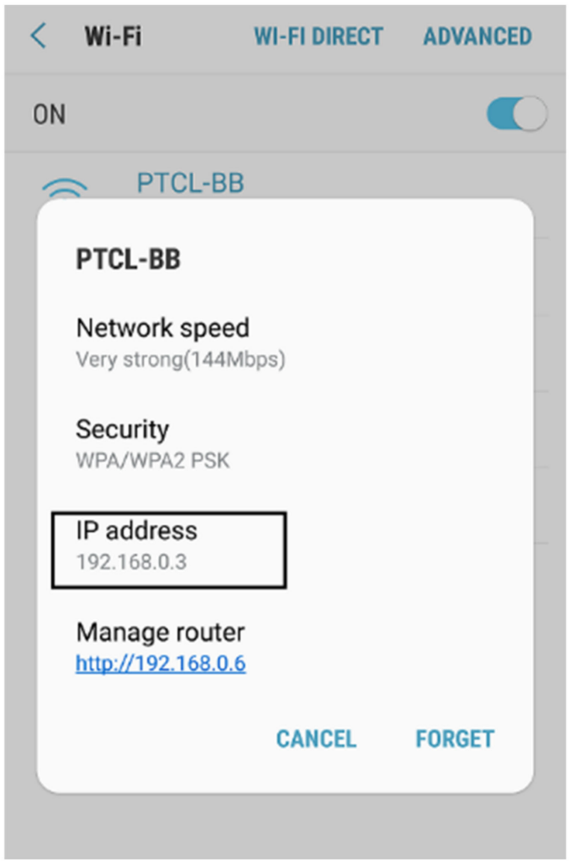


Figure 4. A screenshot of a smartphone IP address.

3.2. Configuration

The IP address for WAN is provided as 10.2.31.180/24 whereas the IP address of LAN interface is 192.168.0.6/24, thus static IP having a pool of DHCP LAN IPs range from 192.168.0.1 to 192.168.0.40.

4. Results and Analysis

The Signal app uses encryption algorithms to secure data for communication. Due to the encrypted payload, only packet sizes, frequency of the packets, and repetitive patterns of packets can be exploited for analysis. Therefore, certain payload sizes may help to determine the specific activity types. Although the privacy and secrecy are not compromised, extensive inspection of packet sizes (patterns, frequencies) reveals the prominent activities of an app. In this study, an extensive analysis of traffic dumps was performed, which revealed bytes and payload patterns of different activities. Although patterns emerged, as expected they did not help in the reconstruction of actual data. However, through the experimental setup explained in the earlier section, regulatory authorities and law enforcement agencies can determine the behaviors of the Signal app by observing consistent connectivity patterns. The complete summary can be seen at the end of the Results and Analysis section.

For better understanding, we created the following users with descriptions to conduct the app activities:

User A: target user, who's entire set of activities are monitored;
User B: to whom User A communicates.

4.1. Identifying of Ports and Servers' Range

After extensive monitoring of the encrypted traffic of the Signal app passing through the firewall, it was revealed that a common 443 TCP port is used throughout the communication process. In the case of blocking the 443 port on the firewall, it will also block other services on the smartphone that may be linked with the same port. Hence, blocking of the 443 port is not a suitable approach to analyze the other connectivity patterns of the Signal app. Therefore, the firewall rules are updated to evaluate new network patterns against different activities performed by the target user. Hence, network patterns against activities such as calling, texting, and typing are monitored and analyzed repeatedly to firmly reach the results.

Unlike many other XMPP (eXtensible Messaging Presence Protocol)-based applications that utilized specific TCP ports such as 5222, 5223, and 5228, the Signal app uses port 443 for communication. For voice and video calls, random UDP open ports are employed by the Signal app. However, through an extensive performing of various Signal app activities, we found a common set of specific chat server IP addresses with respect to our geographic location as shown in Table 3 (detail in Section 5). Similarly, for call services, the Signal client app uses Google's servers in combination with the observed IP addresses of servers as shown in Table 3. This may have been performed for better connectivity or load balancing of traffic.

Table 3. Group of chat servers identified during Signal app activities.

No.	Servers List 1	Servers List 2	Servers List 3	Servers List 4	Servers List 5
1	52.207.41.59	34.225.196.214	34.196.69.69	100.24.0.111	35.169.3.40
2	100.24.0.111	3.228.254.81	3.222.249.138	52.207.41.59	34.196.194.172
3	54.175.47.110	34.196.69.69	100.24.0.111	54.175.47.110	34.225.240.173
4	34.196.69.69	54.175.130.206	3.228.254.81	54.175.149.136	34.225.240.173
5	3.228.254.81	54.175.149.136	54.175.149.136	3.222.249.138	107.23.71.89
6	3.222.249.138	3.222.249.138	52.207.41.59	54.175.130.206	35.169.3.40
7	54.175.130.206	100.24.0.111	34.225.196.214	34.225.196.214	34.196.194.172

4.2. Identification of Signal App Traffic

To identify the activities associated with the Signal app, we dumped the network traffic from the targeted Android device. As the captured traffic was encrypted it could not be decrypted without the cryptographic key. Therefore, we extensively performed and captured the various activities to uncover traffic patterns against the respective activities. User A is a target device on which multiple activities are performed. For better understanding, we analyzed and classified target User A (device) in the capacity of the sender, caller, and receiver, and its respective activities on the app are as follows:

1. Accessing/opening the Signal app;
2. Target device (User A) typing patterns;
3. User B typing patterns;
4. Call initiated by User A (caller);
5. Call received by User A (callee);
6. Media messages;
7. Video calls.

4.2.1. Accessing/Opening the Signal App

The opening of the Signal app activity represents the start of app traces on the Android device. Initially, it was difficult to classify the Signal-only traffic from the shared network traffic packets. Therefore, an extensive number of trace files were captured and monitored during the opening of the Signal app. The process of the above activity is as follows.

- Open the Signal client app on User A device;
- Wait for 3 to 4 s without interacting with the app;
- Close the app.

We observed that a standard DNS query was sent to access the Signal server from the client end, i.e., textsecure-service.whispersystems.org. In response, a list of eight servers with IP addresses was sent back to the client to establish a connection. In most cases, only the first two servers in the list were employed to establish communications. Further, it was noticed that the Signal app employed the two random ports with the server 443 port for authentication. An authentication process is shown in Figure 5. A TCP connection was established in three steps, SYN, SYN ACK, and ACK between client and server. After that, a TLS handshake occurred by using TLSv1.2. Certificates, and session keys were exchanged between the client and server. Next, the client exchanged the key change cipher specs and sent an encrypted handshake message to the server. In response, the server sent a new session ticket, changed the cipher specifications, and sent an encrypted handshake message to the client device using TLS.

After the negotiation of session keys between the client and the server, the following packet patterns with payloads were observed between the server and client ends.

- Signal client app (IP: 192.168.0.5) sent packets of (449, 372, 127) bytes with a payload of size (383, 306, 61), to the Signal servers;
- In response, the server (IP: 54.175.47.110) sent packets of (261, 261, 137) bytes to the Signal client with a payload of (195, 195, 71).

The above activity was performed multiple times to confirm the patterns of opening the Signal app as shown in Figure 6. Hence, we concluded that the above patterns of bytes transmission indicate the opening or accessing of the Signal application.

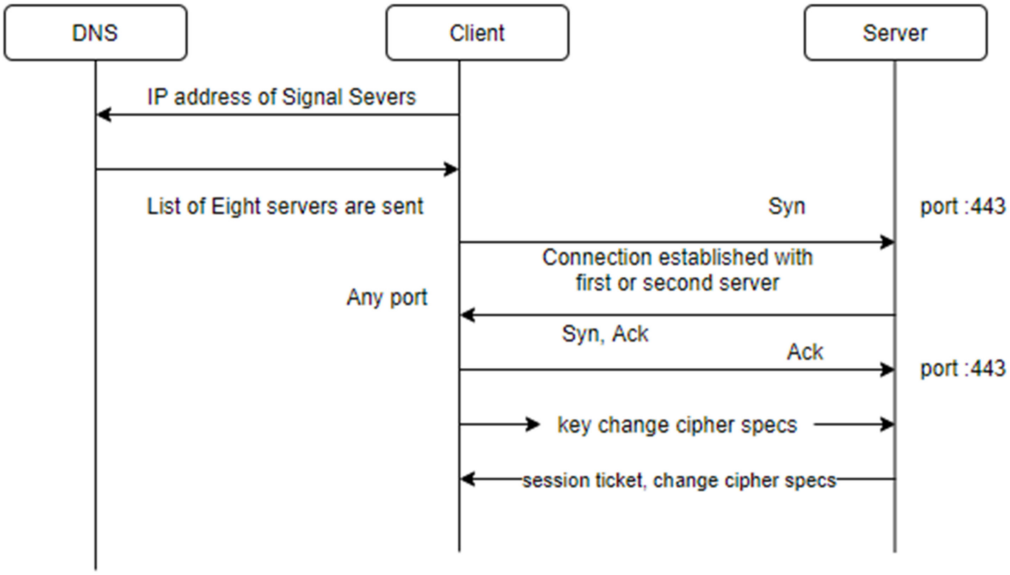


Figure 5. The authentication process of Signal app for connection establishment.

No.	Time	Source	Destination	Protocol	Length	Info
33	09:44:46	192.168.0.5	54.175.47.110	TCP	66	47880 → 443 [ACK]
34	09:44:46	54.175.47.110	192.168.0.5	TLSv1.2	1134	Certificate, Serv
35	09:44:47	192.168.0.5	54.175.47.110	TCP	66	47880 → 443 [ACK]
36	09:44:47	192.168.0.5	54.175.47.110	TCP	66	47879 → 443 [ACK]
37	09:44:47	192.168.0.5	54.175.47.110	TCP	66	47879 → 443 [ACK]
38	09:44:47	192.168.0.5	54.175.47.110	TLSv1.2	192	Client Key Exchan
39	09:44:47	192.168.0.5	54.175.47.110	TLSv1.2	192	Client Key Exchan
40	09:44:47	54.175.47.110	192.168.0.5	TLSv1.2	356	New Session Ticke
41	09:44:47	54.175.47.110	192.168.0.5	TLSv1.2	356	New Session Ticke
42	09:44:47	192.168.0.5	54.175.47.110	TLSv1.2	449	Application Data
43	09:44:47	192.168.0.5	54.175.47.110	TLSv1.2	372	Application Data
44	09:44:47	54.175.47.110	192.168.0.5	TLSv1.2	261	Application Data
45	09:44:47	54.175.47.110	192.168.0.5	TLSv1.2	261	Application Data
46	09:44:47	54.175.47.110	192.168.0.5	TLSv1.2	137	Application Data
47	09:44:47	192.168.0.5	54.175.47.110	TCP	66	47880 → 443 [ACK]
48	09:44:47	192.168.0.5	54.175.47.110	TLSv1.2	127	Application Data
49	09:44:47	192.168.0.5	54.175.47.110	TCP	66	47879 → 443 [ACK]

Figure 6. Opening/accessing the Signal app.

4.2.2. Target Device (User A) Typing Pattern

To notify the patterns of the Signal app when target User A starts typing a message in a chat window, certain flow patterns with fixed payload sizes were noticed. To be assured, these patterns were observed several times in trace files to deduce results as shown in Figure 7. Target User A is highlighted with IP 192.168.137.69. It was noticed that when User A with IP 192.168.137.69 started typing in a chat window, 1181, 1182 and 1183 bytes of data packets with 1115 and 1116 payload size were sent to the Signal server. In response, the server sent 139 or 140 bytes of data packets with payload sizes of 73, 74 to the Signal client. The Signal client in response sent an empty packet of 66 bytes with payload size 0 to the server as an acknowledgment of the previous packets. Meanwhile, it was noticed that the Signal app indicated the typing status to the respective Signal servers, even though the message was not being sent to another party.

No.	Time	Source	Destination	Protocol	Length	Info
1	09:58:25	192.168.137.69	3.228.254.81	TLSv1.2	1181	Application Data
2	09:58:25	3.228.254.81	192.168.137.69	TLSv1.2	139	Application Data
3	09:58:26	192.168.137.69	3.228.254.81	TCP	66	44378 → 443 [ACK]
4	09:58:33	192.168.137.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1
5	09:58:35	192.168.137.69	3.228.254.81	TLSv1.2	1182	Application Data
6	09:58:36	3.228.254.81	192.168.137.69	TLSv1.2	140	Application Data
7	09:58:36	192.168.137.69	3.228.254.81	TCP	66	44378 → 443 [ACK]
8	09:58:36	192.168.137.1	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1
9	09:58:39	192.168.137.69	3.228.254.81	TLSv1.2	1181	Application Data
10	09:58:39	3.228.254.81	192.168.137.69	TLSv1.2	139	Application Data
11	09:58:39	192.168.137.69	3.228.254.81	TCP	66	44378 → 443 [ACK]

Figure 7. User A message typing and sending.

4.2.3. User B Typing Patterns

In this section, we will analyze the traffic patterns of User B. Once User B started typing a message in his/her chat window, which would be sent to the target User A, it was always noticed that 729 bytes of packets with 663 bytes of payload were sent from the server to the client at the device of target User A. The client at target User A also sent 122 bytes of the packet in response with 56 bytes of payload. The target client device A also sent a response with packets of 105 bytes. In sequence, the server kept sending alerts to the client of 101, 97 bytes data packet with 35 and 31 payload sizes as shown in Figure 8. This pattern was observed and confirmed repeatedly through multiple iterations.

No.	Time	Source	Destination	Protocol	Length	TCP Segı	Info
1	11:39:57	13.248.212.111	192.168.137.43	TLSv1.2	729	663	Application Data
2	11:39:57	13.248.212.111	192.168.137.43	TCP	729	663	[TCP Retransmissic
3	11:39:58	192.168.137.43	13.248.212.111	TCP	66	0	41415 → 443 [ACK]
4	11:39:58	192.168.137.43	13.248.212.111	TCP	78	0	[TCP Dup ACK 3#1]
5	11:39:58	192.168.137.43	13.248.212.111	TLSv1.2	122	56	Application Data
6	11:39:58	192.168.137.43	13.248.212.111	TLSv1.2	105	39	Application Data
7	11:39:58	13.248.212.111	192.168.137.43	TCP	66	0	443 → 41415 [ACK]
8	11:39:58	192.168.137.43	13.248.212.111	TLSv1.2	105	39	Application Data
9	11:39:58	13.248.212.111	192.168.137.43	TCP	66	0	443 → 41414 [ACK]
10	11:39:58	13.248.212.111	192.168.137.43	TCP	66	0	443 → 41415 [ACK]
11	11:39:58	13.248.212.111	192.168.137.43	TLSv1.2	101	35	Application Data
12	11:39:58	13.248.212.111	192.168.137.43	TLSv1.2	97	31	Encrypted Alert
13	11:39:58	13.248.212.111	192.168.137.43	TCP	66	0	443 → 41414 [FIN,
14	11:39:58	192.168.137.43	13.248.212.111	TCP	66	0	41414 → 443 [ACK]
15	11:39:58	192.168.137.43	13.248.212.111	TCP	66	0	41414 → 443 [ACK]
16	11:39:58	192.168.137.43	13.248.212.111	TCP	66	0	41414 → 443 [ACK]
17	11:39:58	13.248.212.111	192.168.137.43	TLSv1.2	101	35	Application Data
18	11:39:58	13.248.212.111	192.168.137.43	TLSv1.2	97	31	Encrypted Alert
19	11:39:58	13.248.212.111	192.168.137.43	TCP	66	0	443 → 41415 [FIN,
20	11:39:58	192.168.137.43	13.248.212.111	TCP	66	0	41415 → 443 [ACK]

Figure 8. User B message typing and sending.

4.2.4. Call Initiated by User A (Caller)

Once User A dialed a call, the IP address of the receiver was also captured. While this scenario differs in the case of sending text messages, the IP address of the receiver would never be known even if the users were on the same network. Hence, binding between both users is explicitly seen. It was observed that the Signal app connected to at least three servers to finally connect with the receiver or callee. When a user opens the app, it is connected to a Signal server. Next, the user opens a chat window of the receiver to place a voice call. Further, when a user dials a call, the client app makes a DNS query to the Simple/Session Traversal of UDP through NAT (STUN) server of Google (stun1.l.google.com) and the Traversal Using Relays around NAT (TURN) server (turn1.whispersystems.org). In response to these DNS queries, the STUN server provides a Google server such as (64.233.161.127) and the DNS turn query provides a server IP of whispersystems.org/signal server such as (52.47.185.9) as shown in Figure 9.

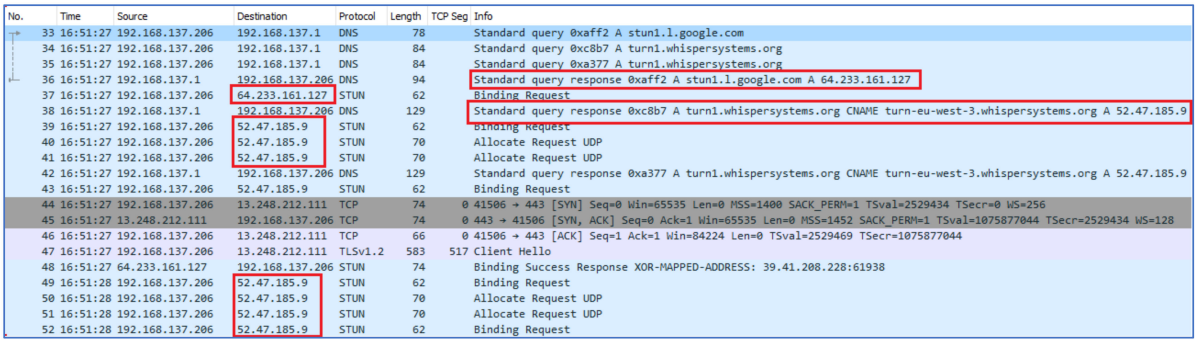


Figure 9. STUN and TURN servers during call initiated by User A.

Stun protocol is used for UDP calls to connect the electronic devices behind the NAT. This resolves the issues related to devices that are deployed between different NAT. Whereas TURN protocol is an extension of STUN. It is used to relay the messages between two devices after a connection is established. It can also be noted that binding is performed by the STUN server, while further communication is relayed between two devices and performed through the TURN server. These patterns were observed while the audio call was established, hence 1388 bytes of reassembled PDUs were sent to the Signal server to indicate audio call activity. When the binding request between the users was successful, it was observed that the private IP address of the other user was also visible. In Figure 10, the IP address of the receiver was 192.168.10.10, and the IP address of the initiating user was 192.168.137.206.

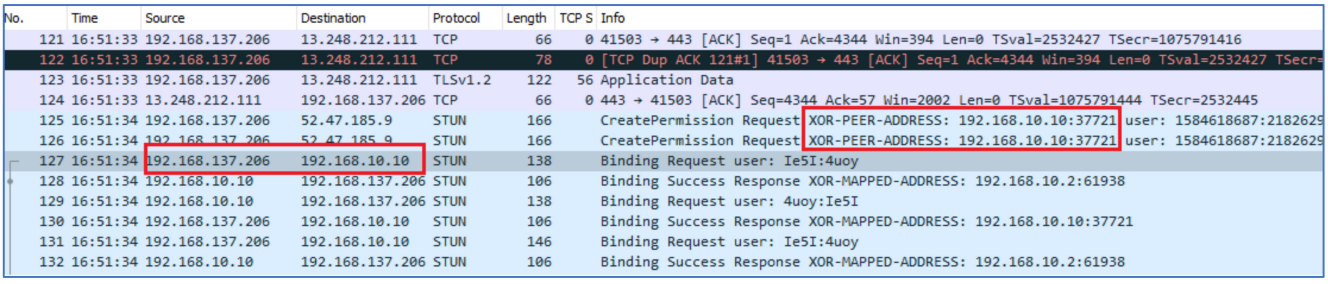


Figure 10. Receiver's IP address during call initiated by User A.

The port numbers used by the TURN servers were 80 and 3478. In most cases, it was noticed that the public IP of the user was mapped with the TURN server. When the call was successfully established between the users, multiple UDP packets of length 22 were found that were transmitted between both users as shown in Figure 11. When the call ended, ICMP packets were sent from the caller to the server that showed that the destination host

was unreachable. The traffic patterns related to the indication of ending a call are shown in Figure 12.

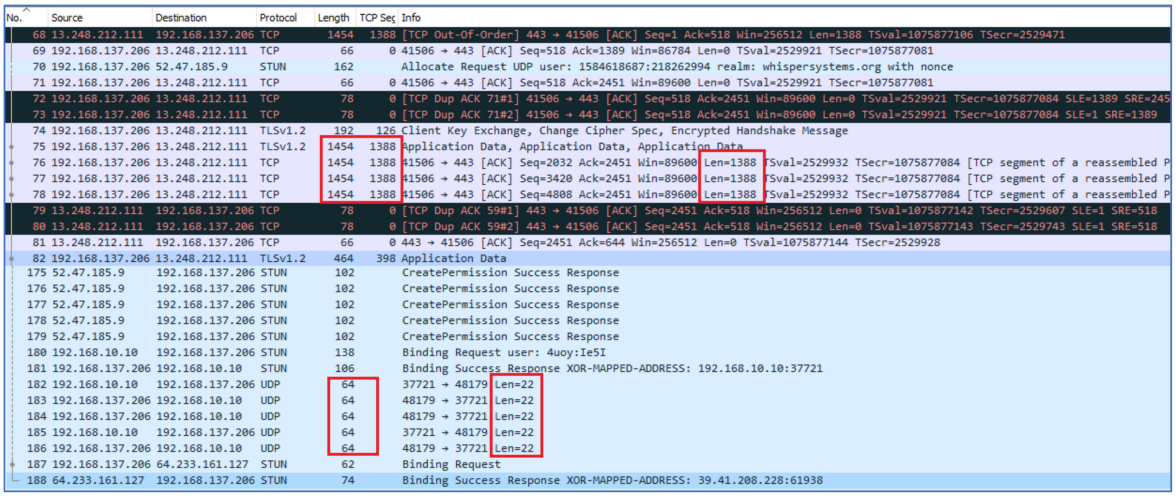


Figure 11. The traffic pattern of established calls from User A to User B.

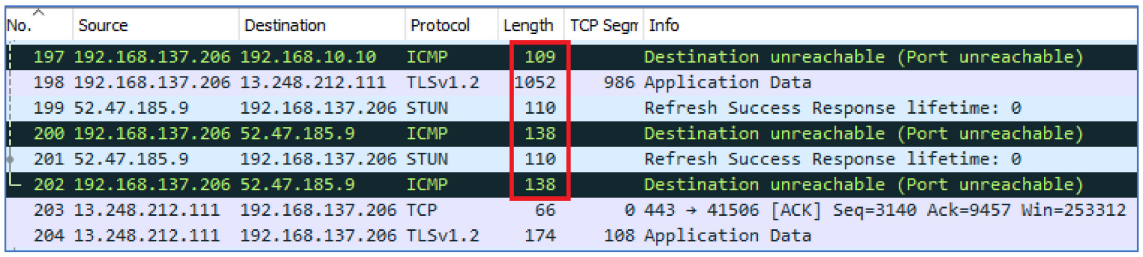


Figure 12. The traffic pattern of ending calls from User A to User B.

4.2.5. Call Received by User A (Callee)

When another user dialed a call to target device A, certain patterns were observed. Initially, the server sent a TCP segment of reassembled PDU to the client with packets of length 1454 bytes and a payload size of 1388 bytes. In response, the client sent 66 bytes of packets to the server. Later UDP packets were observed flowing between the sender and the receiver. It was also noticed that the IP address of the caller was also captured during this activity. The UDP packets of 64 bytes with a payload size of 22 bytes were observed between the receiver and the caller. As shown in Figures 13 and 14, the server's IP was 76.223.92.165, the IP of User A was 192.168.137.252, and the IP of User B was 192.168.10.3. The traffic patterns of incoming calls to User A is shown in Figure 13. The flow of UDP packets during a call is shown in Figure 14.

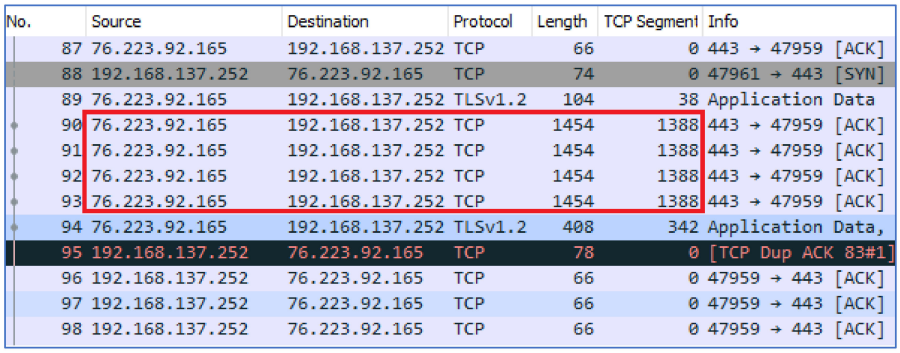


Figure 13. Traffic patterns of incoming calls to User A.

No.	Source	Destination	Protocol	Length	TCP Segment	Info
400	192.168.137.252	192.168.10.3	UDP	64		45475 → 39725 Len=22
401	192.168.10.3	192.168.137.252	UDP	64		39725 → 45475 Len=22
402	192.168.10.3	192.168.137.252	UDP	64		39725 → 45475 Len=22
403	192.168.137.252	192.168.10.3	UDP	64		45475 → 39725 Len=22
404	192.168.137.252	157.175.72.151	STUN	62		Binding Request
405	192.168.137.252	192.168.10.3	STUN	138		Binding Request user:
406	192.168.10.3	192.168.137.252	STUN	106		Binding Success Respo
407	157.175.72.151	192.168.137.252	STUN	126		Binding Success Respo
408	192.168.10.3	192.168.137.252	UDP	64		39725 → 45475 Len=22

Figure 14. The flow of UDP packets during a call.

4.2.6. Media Messages

Media messages include sending pictures, videos, and stickers. These messages are sent from the target User A device to another user. Traced files of media messages were captured and saved for detailed inspection. We noticed that the bytes patterns were the same for all (picture, videos, and stickers) activities. Hence, distinguishing among these messages was difficult. The patterns included multiple groups of reassembled PDU that were seen delivered from client to server. The size of each PDU was 1454 with a payload size of 1388 bytes. It was also noticed that the client used only one port to send messages to the server port 443. After sending the messages, the server sent an equal acknowledgment to the PDU list of size 108 bytes with a payload size of 42 bytes as shown in Figure 15. Due to similar pattern of all the above media messages, it was difficult to distinguish the types, (i.e., image, sticker) of messages. Moreover, the IP address the of receiver was also difficult to identify.

Source	Destination	Protocol	Length	TCP Segment	Info
192.168.137.45	13.35.183.48	TLSv1.2	184	118	Application Data
192.168.137.45	13.35.183.48	TCP	1454	1388	40451 → 443 [ACK] Seq=119 Ack=1 Win=341 Len=1388 TSval=317050981 TSecr=313665589 [TCP segment of a reassembled PDU]
192.168.137.45	13.35.183.48	TCP	1454	1388	40451 → 443 [ACK] Seq=1507 Ack=1 Win=341 Len=1388 TSval=317050981 TSecr=313665589 [TCP segment of a reassembled PDU]
192.168.137.45	13.35.183.48	TCP	1454	1388	40451 → 443 [ACK] Seq=2895 Ack=1 Win=341 Len=1388 TSval=317050981 TSecr=313665589 [TCP segment of a reassembled PDU]
192.168.137.45	13.35.183.48	TCP	1454	1388	40451 → 443 [ACK] Seq=4283 Ack=1 Win=341 Len=1388 TSval=317050981 TSecr=313665589 [TCP segment of a reassembled PDU]
192.168.137.45	13.35.183.48	TCP	1454	1388	40451 → 443 [ACK] Seq=5671 Ack=1 Win=341 Len=1388 TSval=317050981 TSecr=313665589 [TCP segment of a reassembled PDU]
192.168.137.45	13.35.183.48	TLSv1.2	1454	1388	Application Data [TCP segment of a reassembled PDU]
192.168.137.45	13.35.183.48	TCP	1454	1388	40451 → 443 [ACK] Seq=8447 Ack=1 Win=341 Len=1388 TSval=317050981 TSecr=313665589 [TCP segment of a reassembled PDU]
192.168.137.45	13.35.183.48	TCP	1454	1388	40451 → 443 [ACK] Seq=9835 Ack=1 Win=341 Len=1388 TSval=317050981 TSecr=313665589 [TCP segment of a reassembled PDU]
192.168.137.45	13.35.183.48	TCP	1454	1388	40451 → 443 [ACK] Seq=11223 Ack=1 Win=341 Len=1388 TSval=317050981 TSecr=313665589 [TCP segment of a reassembled PDU]

Figure 15. Traffic patterns for all type of media messages.

4.2.7. Video Calls

During the video call, similar patterns of audio calls were observed as shown in Figure 16. The only difference was found in the length of UDP packets that varies from 800 bytes up to 1250 bytes.

No.	Time	Source	Destination	Protocol	Length	TCP Seg	Info
481	21:51:02	192.168.10.3	192.168.137.45	UDP	152		38492 → 46852 Len=110
482	21:51:02	192.168.137.45	192.168.10.3	UDP	152		46852 → 38492 Len=110
483	21:51:02	192.168.137.45	192.168.10.3	UDP	1214		46852 → 38492 Len=1172
484	21:51:02	192.168.10.3	192.168.137.45	UDP	152		38492 → 46852 Len=110
485	21:51:02	192.168.137.45	192.168.10.3	UDP	1214		46852 → 38492 Len=1172
486	21:51:02	192.168.10.3	192.168.137.45	UDP	152		38492 → 46852 Len=110
487	21:51:02	192.168.137.45	192.168.10.3	UDP	152		46852 → 38492 Len=110
488	21:51:02	192.168.137.45	192.168.10.3	UDP	1214		46852 → 38492 Len=1172
489	21:51:02	192.168.10.3	192.168.137.45	UDP	80		38492 → 46852 Len=38
490	21:51:02	192.168.137.45	192.168.10.3	UDP	152		46852 → 38492 Len=110
491	21:51:02	192.168.137.45	192.168.10.3	UDP	1214		46852 → 38492 Len=1172
492	21:51:02	192.168.10.3	192.168.137.45	UDP	152		38492 → 46852 Len=110
493	21:51:02	192.168.137.45	192.168.10.3	UDP	1215		46852 → 38492 Len=1173
494	21:51:03	192.168.137.45	192.168.10.3	UDP	152		46852 → 38492 Len=110

Figure 16. UDP traffic patterns for video calls.

A summary of observed encrypted traffic patterns is shown in Table 4. As we learned the behavior of all major activities, now we can filter out the application traffic by looking at these patterns. From a generic network traffic dump, we can identify the parties without knowing anything about the parties beforehand. By looking at the patterns of protocol, forensic experts, network administrators, and security engineers can filter the Signal packets first and then further inspect them to identify the information about the user activities, type of communication, etc. to examine many cases depicting various scenarios.

Table 4. Summary of observed traffic characteristics of Signal app on android device.

Activities/Events	Client/ Sever	Observed Bytes Patterns	Payload Size
Opening the Signal Application	Client	Client sends encrypted data packets of 449, 372 and 127 bytes to the server.	Actual payload size of these packets are 338, 306, 61 bytes, respectively.
	Server	Server acknowledges the client data with 261, 261 and 137 bytes.	Payload size of these packets are 195, 195, 71 bytes, respectively.
Target User A typing	Client	Client sends encrypted data packets of 1181, 1182 and 1183 bytes to the server.	Payload size of these packets are 1115, 1116 and 1117 bytes, respectively.
	Server	Server responds to the client requests with 139 and 140 bytes.	Payload size of these packets are 73 and 74 bytes, respectively.
User B typing	Server	Server sends encrypted data packets of 729, 101 and 97 bytes to the client at the target device.	Payload size of these packets are 663, 35 and 31 bytes, respectively.
	Client	Client at the target device responds to the server's requests with 122, 105 and 105 bytes.	Payload size of these packets are 56, 39 and 39 bytes, respectively.
User A as Caller	Client	Client sends TCP segment of a reassembled PDU of size 1454 bytes to server. UDP packets length 64.	Payload size of TCP packets is 1388 bytes and UDP payload size is 22 bytes.
	Server	Server responds the above TCP segments with 66 bytes.	Payload size is 0 bytes.
Target User A as Receiver	Server	Server sends TCP segment of reassembled PDU to the client with packets length 1454 bytes. UDP packets length 64 bytes.	TCP payload size is 1388 bytes and UDP payload size is 22 bytes.
	Client	Client responds to the above TCP segments sent by the server with 66 bytes.	Payload size is 0 bytes.
Media Messages by Target User A	Client	Client device sends media messages such as pictures, video, and stickers. TCP segment of assembled PDU 1454 bytes is noticed.	Payload size of the packets is 1388 bytes.
	Server	After Ack with 66 bytes, the server sends 108 bytes of data to client.	Payload size of 108 bytes is 42 bytes.
Video Call	Client-Server	UDP packet lengths varying from 800 bytes up to 1250 bytes are exchanged between client and server.	Payload size varies.
Call Termination	Client	Client sends 109 and 138 bytes of ICMP packets to the receiver and server, respectively.	

4.3. Performance Comparision

In this section, we discuss and compare the identified potential artifacts from the proposed and existing strategies as shown in Table 5. It is depicted that [7,9,12,29] strategies have not targeted the network traffic analysis. Meanwhile, Karpisek et al. [19] aimed the network forensic of WhatsApp and successfully identified the video/audio calls (initiating, duration, termination) activity with an involved IP address. However, the messaging, typing, accessing, or closings of apps artifacts were not explored. Furthermore, the above results were taken from WhatsApp version 2.11.x.x that was released on 5 March 2015 almost 6 years ago. Similarly, Walnycky et al. [8] successfully identified the call send/receive

messages, etc. on various apps except for the Signal app. Recently, Sudozai et al. [18] successfully identified the artifacts only on the IMO app. However, the proposed strategy successfully identified the app opening/closing, typing, media messages, calling with involved IP addresses from the encrypted network traffic for the Signal app.

Table 5. A comparison of identified artifacts from encrypted network packets of the proposed and existing strategies.

Year	Authors	Title	Apps	Network Forensics					
				Detecting User Activities from the Packets				IP Addresses of Involved Parties	Identified No. Server and IPs
				Accessing the App	Audio Video Calling	Media Messages	Typing		
2015	Anglano et al. [10]	*Forensic Analysis of WhatsApp Messenger on Android Smartphone*	WhatsApp	No	No	No	No	No	No
2015	Walnycky et al. [8]	*Network and device forensic analysis of Android social messaging applications*	SnapChat, Wickr, BBM, Tinder, and 16 other apps	Yes	Yes	Yes	Yes	No	No
2015	Karpisek et al. [19]	*WhatsApp Network Forensics: Decrypting and Understanding the WhatsApp Call Signaling Messages*	WhatsApp	No	Yes	Yes	No	Yes	Yes
2016	Wu et al. [9]	*Forensic analysis of WeChat on Android smartphones*	WeChat	No	No	No	No	No	No
2018	Sudozai et al. [18]	*Forensics study of IMO call and chat app*	IMO	Yes	Yes	Yes	Yes	Yes	Yes
2018	Rathi et al. [12]	*Forensic Analysis of Encrypted Instant Messaging Applications on Android*	WeChat, WhatsApp, Telegram, Viber	No	No	No	No	No	No
2020	Shawn et al. [29]	*What is really "Happning"? A forensic analysis of Android and iOS Happn dating apps*	Happn	No	No	No	No	No	No
2021	**Proposed Strategy**	***Encrypted Network Traffic Analysis of Secure Instant Messaging Application: A Case Study of Signal Messenger App***	**Signal App**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

5. Discussion and Use Cases

In this section, we will discuss various use cases to validate and verify the identified evidence from the proposed forensic strategy regarding the Signal app analysis.

5.1. Blocking List of Servers

While performing all the above-mentioned activities, certain results are deduced. For example, during multiple activities, we observed a group of Signal servers that are found in the response of the DNS query. While monitoring the network packets during the Signal app opening and connection times, it was noted that the app always connects to one of the servers mentioned in the DNS query, i.e., textsecureservice.whispersystems.org. During the experimentation, a group of eight type A servers were discovered in most DNS queries as shown in Figure 17. It was also observed that an application establishes a connection

with one of these servers using two random ports. The details of all those servers can be found in packets details in the Wireshark as shown in Figure 18.

No.	Source	Destination	Protocol	Length	Info
1	192.168.137.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
2	192.168.137.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
3	192.168.137.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
4	192.168.137.196	192.168.137.72	DNS	97	Standard query 0x5c72 A textsecure-service.whispersystems.org
5	192.168.137.196	192.168.137.72	DNS	97	Standard query 0x4e74 A textsecure-service.whispersystems.org
6	192.168.137.72	192.168.137.196	DNS	225	Standard query response 0x5c72 A textsecure-service.whispersystems.org A 52.207.41.59 A 100.24.
7	192.168.137.196	52.207.41.59	TCP	74	48641 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1400 SACK_PERM=1 TSval=229071609 TSecr=0 WS=256
8	192.168.137.72	192.168.137.196	DNS	225	Standard query response 0x4e74 A textsecure-service.whispersystems.org A 54.175.47.110 A 34.225

Figure 17. A sample screenshot of identified server using DNS query.

```
Authority RRs: 0
Additional RRs: 0
Queries
  textsecure-service.whispersystems.org: type A, class IN
      Name: textsecure-service.whispersystems.org
      [Name Length: 37]
      [Label Count: 3]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
Answers
  textsecure-service.whispersystems.org: type A, class IN, addr 52.207.41.59
  textsecure-service.whispersystems.org: type A, class IN, addr 100.24.0.111
  textsecure-service.whispersystems.org: type A, class IN, addr 54.175.47.110
  textsecure-service.whispersystems.org: type A, class IN, addr 34.196.69.69
  textsecure-service.whispersystems.org: type A, class IN, addr 3.228.254.81
  textsecure-service.whispersystems.org: type A, class IN, addr 3.222.249.138
  textsecure-service.whispersystems.org: type A, class IN, addr 54.175.130.206
  textsecure-service.whispersystems.org: type A, class IN, addr 34.225.196.214
```

Figure 18. Type A Server Addresses.

The Signal app attempts to connect either with the first server using one of the two ports or with the first two servers using any of the two ports mentioned in the list of the servers. It was also noticed that one port was used to establish a connection for indicating the typing patterns while the other port was used to send the actual data in the context of text messages. There are almost 10 servers that are used to connect with clients. Chat servers found during the study are already shown in Table 3. It is also worth mentioning that server's addresses are common in each list. However, the order of these addresses may be changed. To validate the server IP addresses of different services, firewall rules are applied to check the response of the app on the client end, because the firewall always provides an edge to control and monitor the ongoing activities within the network.

5.2. Blocking Chat Servers

In this section, we verify the chat servers' IPs by blocking them using firewall rules on the pfSense software [34]. Blocking these servers would let us know the connectivity pattern of the client app. For example, once the chat servers are blocked, the chat messages should not be delivered to another end. For experiments, we blocked the traffic of IP addresses of chat servers (on Table 3) for LAN and WAN through firewall rules. As a result, the client app of Signal was not able to connect to these servers and the messages were not sending as depicted in Figure 19. Signal servers use port 443 to connect to the client which cannot be blocked as other apps used 443 for communication purposes, (443 is an SSL communication port and highly resistant to eavesdropping and interception). Remote servers providing services using this port are trustworthy and verified without any doubt. Similarly, the web servers listen on this port that accepts and establishes secure connections for web browsers that desire strong communication security. Hence, blocking these servers would help to stop the Signal app services.

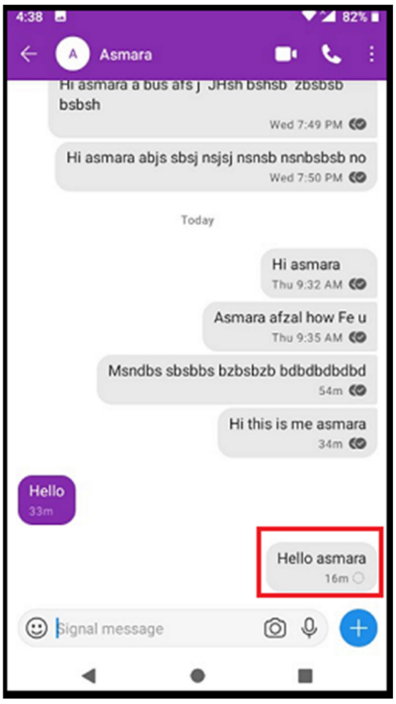
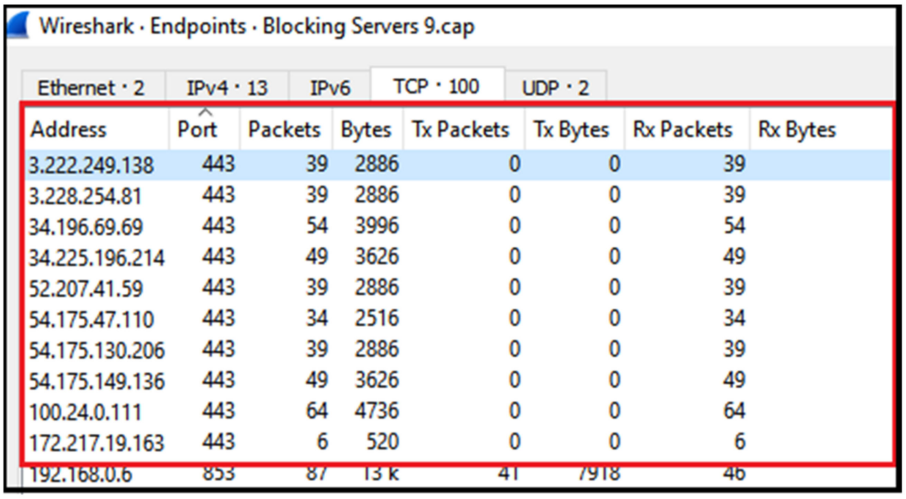


Figure 19. Screenshot of message sending failed.

If we look at the Signal app chat behavior on a mobile device in Figure 19, it tried almost 16 min to deliver the text message but failed. Meanwhile, the app tried to connect to all the above-mentioned chat servers. The Wireshark-based analysis shows the details of these servers in Figure 20. When the client app tried to connect to these servers to deliver a message to another user, the mentioned servers did not respond or a 0 was sent in response to the client requests. It was noted that all services were working correctly without the restriction of IP addresses on the firewall. However, firewall rules used to block/delay the chat activities are mentioned in Table 6. With the help of the rules and list of servers, the chat service was disrupted. The device of target User A tried to send messages to other servers but failed to do so. Similarly, Figure 21 shows that the client application tried to connect the main servers to send messages. However, the connection was denied with each of them as highlighted in black.



Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
3.222.249.138	443	39	2886	0	0	39	
3.228.254.81	443	39	2886	0	0	39	
34.196.69.69	443	54	3996	0	0	54	
34.225.196.214	443	49	3626	0	0	49	
52.207.41.59	443	39	2886	0	0	39	
54.175.47.110	443	34	2516	0	0	34	
54.175.130.206	443	39	2886	0	0	39	
54.175.149.136	443	49	3626	0	0	49	
100.24.0.111	443	64	4736	0	0	64	
172.217.19.163	443	6	520	0	0	6	
192.168.0.6	853	87	13 k	41	7918	46	

Figure 20. Response from blocked servers.

Table 6. Firewall rules for blocking the chat servers.

Step	Protocol	Source Port	Destination Port	Action	Observation
1	TCP	Any	Any	Default allow LAN to any rule.	All services work smoothly
2	TCP	Any	Any	Block the entire list of IP addresses of chat servers.	Message sending failed

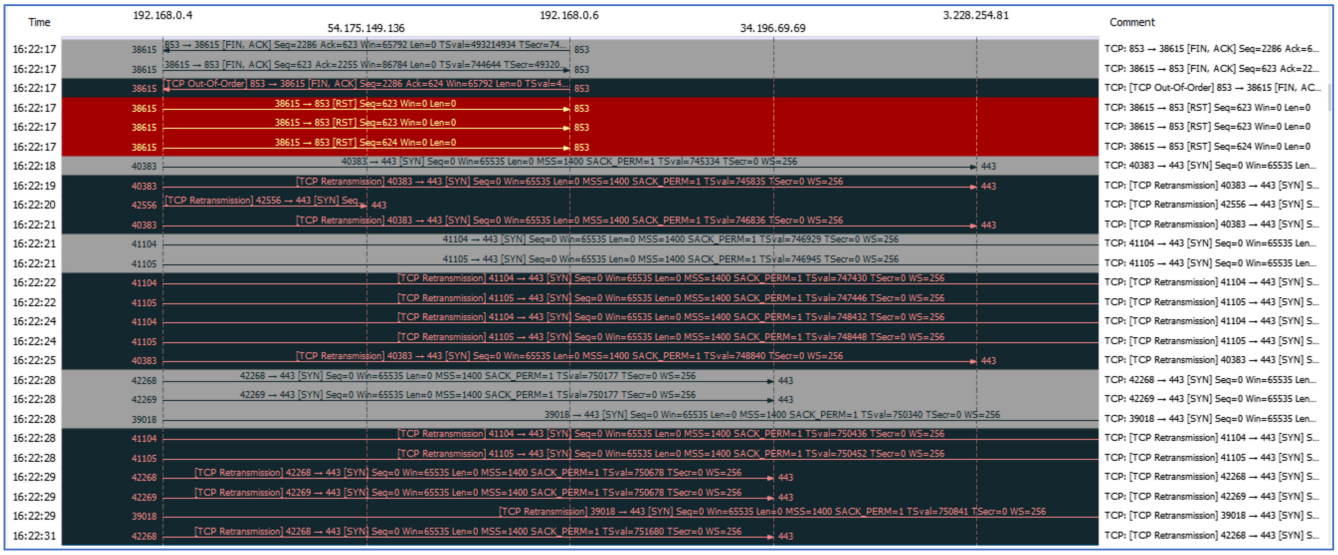


Figure 21. Traffic pattern responses from blocked servers.

5.3. List of Call Servers and Blocking

Through experiments, many trace files were captured and saved during call activity. As discussed earlier, the IP address of the call receiver can be captured during call activity, unlike messages activity. It is worth noting that at least three servers were involved. The initial server authenticates the client and the DNS query response, where two more IP addresses of the servers were sent to the client for further communication. The client connected with the first server (i.e., Google's) which helps to find the receiver's IP address. The second server helps to relay the communication between the caller and receiver. After collecting the above, the client was connected to the desired receiver as shown in Table 7.

Table 7. IP addresses found for call servers and receivers.

Trace Files	Client's IP	Initial Server	Google's Stun Server	Signal's TURN Server	Receiver's Private IP	Receiver's Public IP
1	192.168.137.34	13.248.212.111	64.233.161.127	52.47.185.9	192.168.0.5	
2	192.168.137.206	13.248.212.111	64.233.161.127	52.47.185.9	192.168.10.10	
3	192.168.137.206	13.248.212.111	64.233.161.127	52.47.185.9	192.168.10.10	
4	192.168.137.25	76.233.92.165	64.233.161.127	52.47.185.9	192.168.0.6	182.182.252.29
5	192.168.137.25	13.248.212.111	64.233.161.127	52.47.185.9	192.168.0.6	
		76.233.92.165				
6	192.168.137.25	13.248.212.111	64.233.161.127	52.47.185.9	192.168.0.3	
		76.233.92.165				
		74.125.133.188				
7	192.168.137.43	13.248.212.111	64.233.161.127	52.47.185.9	192.168.0.6	182.182.252.29

5.4. Crime Scene Reconstruction

The details of results found during this study can help in investigating the cases involving live traffic monitoring and capturing based on the Signal app. The forensics examiner can deduce the results of on-going calls, messages, and typing patterns. The communicating parties can also be identified with the IP addresses during the real-time call as well. The forensic examiner needs to obtain access to the target network with the help of the specific organization. Traffic dumps of the ongoing traffic are required to be captured at a certain time of the suspect's activities. With the help of the results deduced in the section Results and Analysis, specific traffic can be filtered out among the ongoing traffic. Specific activities with the help of predefined patterns can be distinguished. Further, the information of the users with captured IP addresses can also be verified from ISPs.

5.5. Use Case (Hypothesis)

As we noticed, encrypted network traffic analysis can help in proving evidence of device forensics. For example, an investigator may correlate the evidence collected from device and network packets to infer or validate the results. Therefore, the proposed strategy may help to analyze the real-time/offline application behavior (from encrypted network packets) which can be useful for the investigator.

Use Case: For example, if sensitive official digital media was leaked in the organization using Signal app, to trace the activity, firewall logs can help investigate the activities assuming that the logs collection/retention policy is implemented in a way that an investigator can obtain information about devices, source IP, destination IP, payload sizes, services, protocols, and timestamps. Investigators can use the study of the behavioral analysis of the application to interpret the logs. Therefore, the proposed strategy for analysis of encrypted traffic of apps (such as Signal) can support in deducing the activities and involved users. For example:

1. If documents were leaked and received by users on 5th April around some specific time window such as 21:20 PM to 23:00;
2. Traffic dumps for the duration of interest were taken. Figure 22 depicts that the Signal app communicated with its DNS for that specific duration;
3. Identify and analyze the traffic for particular activity as shown in Figure 23 that depicts media message patterns were observed in a particular timeline with a specific source IP;
4. Finally, the investigator identified the source IP address from DHCP list;
5. A temporal analysis of a particular incident can be seen in Table 8.

Table 8. Temporal analysis of a particular incident.

Date	Source IP	Destination IP	Timestamps	Session	Protocol	Packet Type	Packet Length
5 April 2020	192.269.137.45	192.269.137.1	21:23:59 PM	Client to Server	DNS	DNS	97
			21:23:59 PM	Server to Client	DNS	DNS (Response)	113
	13.35.183.42	192.269.137.45	21:24:14 PM	Client to Server	TCP	TCP Segments—1454 bytes payload size	1388
			21:24:14 PM	Server to Client	TCP	ACK	66

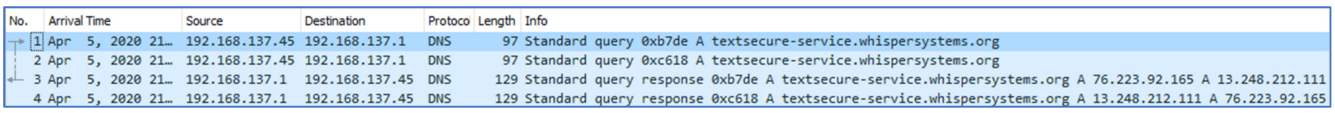
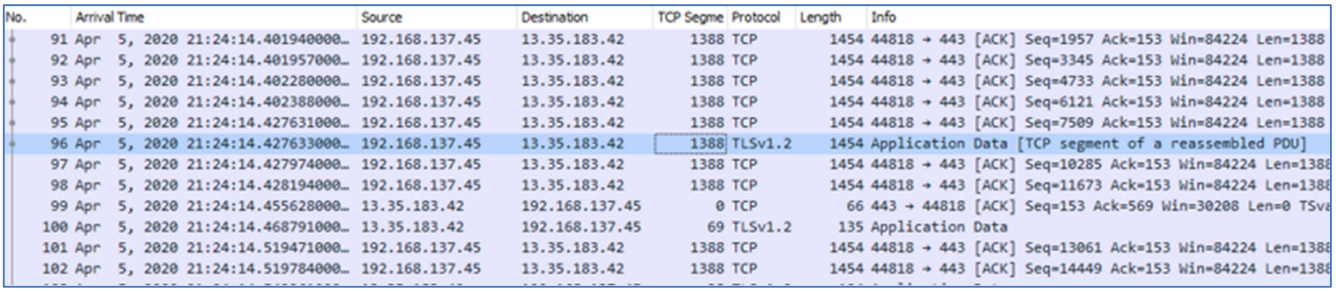


Figure 22. Signal DNS queries and responses.



No.	Arrival Time	Source	Destination	TCP Segme	Protocol	Length	Info
91	Apr 5, 2020 21:24:14.401940000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=1957 Ack=153 Win=84224 Len=1388
92	Apr 5, 2020 21:24:14.401957000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=3345 Ack=153 Win=84224 Len=1388
93	Apr 5, 2020 21:24:14.402280000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=4733 Ack=153 Win=84224 Len=1388
94	Apr 5, 2020 21:24:14.402388000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=6121 Ack=153 Win=84224 Len=1388
95	Apr 5, 2020 21:24:14.427631000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=7509 Ack=153 Win=84224 Len=1388
96	Apr 5, 2020 21:24:14.427633000…	192.168.137.45	13.35.183.42	1388	TLSv1.2	1454	Application Data [TCP segment of a reassembled PDU]
97	Apr 5, 2020 21:24:14.427974000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=10285 Ack=153 Win=84224 Len=1388
98	Apr 5, 2020 21:24:14.428194000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=11673 Ack=153 Win=84224 Len=1388
99	Apr 5, 2020 21:24:14.455628000…	13.35.183.42	192.168.137.45	0	TCP	66	443 → 44818 [ACK] Seq=153 Ack=569 Win=30208 Len=0 TSva
100	Apr 5, 2020 21:24:14.468791000…	13.35.183.42	192.168.137.45	69	TLSv1.2	135	Application Data
101	Apr 5, 2020 21:24:14.519471000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=13061 Ack=153 Win=84224 Len=1388
102	Apr 5, 2020 21:24:14.519784000…	192.168.137.45	13.35.183.42	1388	TCP	1454	44818 → 443 [ACK] Seq=14449 Ack=153 Win=84224 Len=1388

Figure 23. Media message patterns are detected.

As in our lab, we had a controlled environment to study and analyze the behavior of the application ports and IP ranges. However, assumptions are as follows:

1. Controlled environment, admin has control over network and firewall;
2. Logs are being maintained according to the specific time duration;
3. After an incident, investigators can obtain the firewall logs to analyze the performed activities.

6. Benefits of Proposed Strategy

In this section, we highlight the possible benefits of the proposed strategy that can be utilized in the identified usage scenarios.

- If someone is already a victim or culprit, monitoring the network traffic, can help to investigate the incident (Section 5.5);
- If an organization is interested in disabling the services of this type of app without disrupting the other services, the proposed strategy can become helpful (Sections 5.1–5.3);
- The proposed strategy can help in identifying the traffic and ongoing activities without decrypting the traffic (Section 5.5);
- The proposed strategy can identify important artifacts such as IP addresses, timestamps, packets, and payload sizes (Table 4);
- The proposed strategy can help in reconstructing important events after obtaining the log files from the firewall (see Section 5.5);
- Live network monitoring can also help to make decisions during the investigations in many cases;
- Helps to identify the other existing application signatures;
- Verify the claims that the Signal application provides.

7. Limitations and Future Work

The major limitation of the proposed work is the lack of device forensic analysis even though we extensively studied the encrypted network traffic of the Signal app and demonstrated significant detection of the app's behavior as supportive evidence for forensic investigation. Device forensic analysis helps to provide a complete solution in terms of investigation; therefore, device forensic can be targeted in the future. The second limitation consists of version dependency. Forensics analysis of applications is dependent on the specific version of the application. Hence, the proposed work is also version dependent. If a new version of the application is launched with extensive modification of the structure or network behavior, then its network traffic patterns may change and may conflict with the results presented in this work. In the future, we will work to provide a GUI-based solution for existing IM-based apps with configurable attributes including but not limited to ports and associated server IPs, etc., to generalize the proposed strategy and possibly make it portable to new versions of the application.

8. Conclusions

In this paper, the encrypted network traffic of the Signal app was analyzed. The Signal app was used as a case study to show the effectiveness of the proposed strategy in analyzing

the encrypted network traffic. Without knowledge of communication protocol and security architecture, we observed possible behavior from encrypted network traffic of the Signal app which can be beneficial to investigate a criminal case involving the Signal app. The new idea of using a firewall that was supported by the proposed strategy was implemented to reveal obscured call connectivity scenarios of the Signal app in such studies.

Furthermore, the experimental results on Signal traffic patterns can facilitate and correctly identify the events of Signal chats, voice, video calls and revealed IP addresses of involved parties. It also helped to list the involved servers in the proposed strategy. It would be interesting to see the results of the proposed strategy if it is used for the analysis of some other social media apps by slight fine-tuning in the IP ranges and ports of associated devices.

Author Contributions: All authors discussed the contents of the manuscript and contributed equally to its preparation. Especially, A.A. performed experiments, M.H. designed and supervised the study, S.S. analyzed the experimental results, M.K.S. aided to interpret the results, A.T.S.H. reviewed the article and K.-H.J. drafted the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National University of Sciences and Technology (NUST) under the Department of Computing, School of Electrical Engineering and Computer Science, Islamabad, Pakistan. This research was supported by Brain Pool program funded by the Ministry of Science and ICT through the National Research Foundation of Korea (2019H1D3A1A01101687) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1A09081842, 2021R1I1A3049788).

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: The datasets used in this paper are publicly available and their links are provided in the References section.

Acknowledgments: We thank the anonymous reviewers for their valuable suggestions that improved the quality of this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Omolara, A.E.; Jantan, A.; Abiodun, O.I.; Dada, K.V.; Arshad, H.; Emmanuel, E. A Deception Model Robust to Eavesdropping Over Communication for Social Network Systems. *IEEE Access* **2019**, *7*, 100881–100898. [CrossRef]
2. Staff Writers. Instant Messaging Users to Reach 4.3 Billion in 2020: Juniper Research. Available online: https://which-50.com/instant-messaging-users-to-reach-4-3-billion-in-2020-juniper-research/ (accessed on 27 May 2020).
3. Enberg. Global Mobile Messaging 2020. Available online: https://www.emarketer.com/content/global-mobile-messaging-2020 (accessed on 21 October 2020).
4. Arshada, H.; Jantana, A.; Hoon, G.K.; Abiodun, I.O. Formal knowledge model for online social network forensics. *Comput. Secur.* **2019**, *89*, 101675. [CrossRef]
5. Zhang, H.; Chen, L.; Liu, Q. Digital Forensic Analysis of Instant Messaging Applications on Android Smartphones. In Proceedings of the 2018 International Conference on Computing, Networking and Communications (CNC), Maui, HI, USA, 5–8 March 2018; pp. 647–651. [CrossRef]
6. Al Mutawa, N.; Baggili, I.; Marrington, A. Forensic analysis of social networking applications on mobile devices. *Digit. Investig.* **2012**, *9*, S24–S33. [CrossRef]
7. Anglano, C.; Canonico, M.; Guazzone, M. Forensic analysis of Telegram Messenger on Android smartphones. *Digit. Investig.* **2017**, *23*, 31–49. [CrossRef]
8. Walnycky, D.; Baggili, I.; Marrington, A.; Moore, J.; Breitinger, F. Network and device forensic analysis of Android social-messaging applications. *Digit. Investig.* **2015**, *14*, S77–S84. [CrossRef]
9. Wu, S.; Zhang, Y.; Wang, X.; Xiong, X.; Du, L. Forensic analysis of WeChat on Android smartphones. *Digit. Investig.* **2017**, *21*, 3–10. [CrossRef]
10. Anglano, C. Forensic analysis of WhatsApp Messenger on Android smartphones. *Digit. Investig.* **2014**, *11*, 201–213. [CrossRef]
11. Gregorio, J.; Gardel, A.; Alarcos, B. Forensic analysis of Telegram Messenger for Windows Phone. *Digit. Investig.* **2017**, *22*, 88–106. [CrossRef]

12. Rathi, K.; Karabiyik, U.; Aderibigbe, T.; Chi, H. Forensic analysis of encrypted instant messaging applications on Android. In Proceedings of the 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 22–25 March 2018. [CrossRef]
13. Choi, J.; Yu, J.; Hyun, S.; Kim, H. Digital forensic analysis of encrypted database files in instant messaging applications on Windows operating systems: Case study with KakaoTalk, NateOn and QQ messenger. *Digit. Investig.* **2019**, *28*, S50–S59. [CrossRef]
14. Anglano, C.; Canonico, M.; Guazzone, M. The Android Forensics Automator (AnForA): A tool for the Automated Forensic Analysis of Android Applications. *Comput. Secur.* **2020**, *88*, 101650. [CrossRef]
15. Conti, M.; Mancini, L.V.; Spolaor, R.; Verde, N.V. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In Proceedings of the CODASPY 2015: 5th ACM Conference on Data and Application Security and Privacy, San Antonio, TX, USA, 2–4 March 2015; pp. 297–304. [CrossRef]
16. Sikos, L.F. Packet analysis for network forensics: A comprehensive survey. *Forensic Sci. Int. Digit. Investig.* **2020**, *32*, 200892. [CrossRef]
17. Montasari, R.; Hill, R.; Carpenter, V.; Montaseri, F. Digital Forensic Investigation of Social Media, Acquisition and Analysis of Digital Evidence. *Int. J. Strat. Eng.* **2019**, *2*, 52–60. [CrossRef]
18. Sudozai, M.; Saleem, S.; Buchanan, W.J.; Habib, N.; Zia, H. Forensics study of IMO call and chat app. *Digit. Investig.* **2018**, *25*, 5–23. [CrossRef]
19. Karpisek, F.; Baggili, I.; Breitinger, F. WhatsApp network forensics: Decrypting and understanding the WhatsApp call signaling messages. *Digit. Investig.* **2015**, *15*, 110–118. [CrossRef]
20. Rodrigues, G.A.P.; Albuquerque, R.D.O.; de Deus, F.E.G.; De Sousa, R.T., Jr.; Júnior, G.A.D.O.; Villalba, L.J.G.; Kim, T.-H. Cybersecurity and Network Forensics: Analysis of Malicious Traffic towards a Honeynet with Deep Packet Inspection. *Appl. Sci.* **2017**, *7*, 1082. [CrossRef]
21. Velan, P.; Cermak, M.; Celeda, P.; Drašar, M. A survey of methods for encrypted traffic classification and analysis. *Int. J. Netw. Manag.* **2015**, *25*, 355–374. [CrossRef]
22. Perrin, T. The XEdDSA and VXEdDSA Signature Schemes. 2016; p. 14. Available online: https://signal.org/docs/specifications/xeddsa/ (accessed on 18 April 2020).
23. The X3DH Key Agreement Protocol. Available online: https://signal.org/docs/specifications/x3dh/ (accessed on 18 April 2020).
24. The Double Ratchet Algorithm. Available online: https://signal.org/docs/specifications/doubleratchet/ (accessed on 18 April 2020).
25. Marlinspike, M.; Perrin, T. The Sesame Algorithm: Session Management for Asynchronous Message Encryption. 2017; p. 17. Available online: https://signal.org/docs/specifications/sesame/ (accessed on 18 April 2020).
26. Signal-Home. Available online: https://signal.org/ (accessed on 18 April 2020).
27. Awan, F.A. Forensic examination of social networking applications on smartphones. In Proceedings of the 2015 Conference on Information Assurance and Cyber Security (CIACS), Rawalpindi, Pakistan, 18 December 2015; pp. 36–43. [CrossRef]
28. He, G.; Xu, B.; Zhu, H. Identifying Mobile Applications for Encrypted Network Traffic. In Proceedings of the 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD), Shanghai, China, 13–16 August 2017; pp. 279–284. [CrossRef]
29. Knox, S.; Moghadam, S.; Patrick, K.; Phan, A.; Choo, K.-K.R. What's really 'Happning'? A forensic analysis of Android and iOS Happn dating apps. *Comput. Secur.* **2020**, *94*, 101833. [CrossRef] [PubMed]
30. Yusoff, M.N.; Dehghantanha, A.; Mahmod, R. Network Traffic Forensics on Firefox Mobile OS: Facebook, Twitter, and Telegram as Case Studies. In *Contemporary Digital Forensic Investigations of Cloudand Mobile Applications*; Elsevier Inc.: Amsterdam, The Netherlands, 2017; pp. 63–78.
31. Schröder, S.; Huber, M.; Wind, D.; Rottermanner, C. When SIGNAL hits the Fan: On the Usability and Security of State-of-the-Art Secure Mobile Messaging. In Proceedings of the 1st European Workshop on Usable Security 2016, Darmstadt, Germany, 18 July 2016. [CrossRef]
32. L. SaurikIT, "Cydia Substrate," 2016. Available online: http://www.cydiasubstrate.com (accessed on 14 August 2021).
33. M. Blanchou, "Android-ssl-Trustkiller," 2016. Available online: https://github.com/iSECPartners/Android-SSL-TrustKiller (accessed on 14 August 2021).
34. pfSense. Available online: https://www.pfsense.org/ (accessed on 14 August 2021).
35. Wireshark. Available online: https://www.wireshark.org/ (accessed on 14 August 2021).
36. Limited Version of Captured Trace Files Are Available. Available online: https://drive.google.com/drive/folders/1Ke64Ftd4K_jD4HFPp_lSnAAaVnz4grX4?usp=sharing (accessed on 14 August 2021).