A Security Analysis of Instant Messaging platforms.

This research paper analyses the security features offered by various instant messaging platforms and attempts to identify the most cryptographically secure platform.

AAYUSH CHOPRA

Graduate Student, Simon Fraser University, BC, Canada

SAMEER AHMED

Graduate Student, Simon Fraser University, BC, Canada

A comprehensive research paper that focuses on cryptographic protocols and security features used by various instant messaging platforms. Latest security features in the instant messaging domain are described to set the criterion for the rest of the paper. Based on our research, Signal Messenger is currently the most secure instant messenger platform in the industry. Furthermore, Signal protocol used in Signal Messenger offers some of the latest security features such as perfect forward secrecy and post compromise secrecy. The research paper also discusses some of the competitors to the Signal protocol in the industry and some poorly implemented instant messaging platforms.

CCS CONCEPTS • Perfect Forward Secrecy • Post Compromise Security • Double Ratchet Algorithm • Extended Triple Diffie-Hellman

Additional Keywords and Phrases: End-to-end encryption, Signal Protocol, Edward Curves, Ephemeral Keys, Key Continuity

1 INTRODUCTION

Instant messaging services have been around for several decades starting from the 1980s which saw the launch of Internet Relay Chat Protocol take place (IRC) [1]. Fast forward to the 2000s which saw the enormous popularity of platforms such as Yahoo Messenger, Windows Live Messenger (MSN Messenger), AIM (AOL Instant Messenger), Skype and others take place. Right at the onset of these platforms being launched there was very little attention paid to the security of messages being exchanged on these platforms [2]. As the popularity of these instant messaging platforms grew, subsequently the need for securing data being exchanged over the internet also grew. Security protocols have thereby been devised and implemented to safeguard the enormous amount of data being transferred over these instant messaging platforms. SecureIM used by Trillian was one of the first security protocols designed exclusively for instant messaging. Ever since there has been an

advent of such protocols such as eXtensible Message and Presence Protocol (XMPP) [3], Off-the-Record (OTR) [4], Silent Circle Instant Message Protocol (SCIMP) [5], ZRTP [6] and more recently Signal [7] to name a few. On the other hand email services started using cryptographic protocols such as Privacy Enhanced Mail (PEM) [8] Secure Multipurpose Internet Mail Extensions (S/MIME) [9] and Pretty Good Privacy (PGP) [10] [11]. Alternatively, AIM, MSN, Yahoo Messenger had their own security protocols in the initial years to implement whatever security they deemed appropriate at the time [12]. As can be noticed that there has been a constant need to improve security features in instant messaging platforms and thereby security protocols in the recent past have tried to overcome the failures of the past protocols. This paper is aimed at identifying and analyzing the most secure instant messaging platform available with respect to cryptographic security and briefly mentioning other non-cryptographic security factors as well.

With over two billion users in WhatsApp [13], nearly 1 billion users in WeChat and about 1 billion users in Facebook Messenger it is fair to establish the wide outreach of instant messaging platforms in the modern world. These numbers are only likely to increase as these platforms reach out to more people who gain access to the internet. Therefore, it becomes more important than ever to ensure that secure practices and only the most secure cryptographic protocols are used in these increasingly popular instant messaging platforms.

As noted earlier, there have been several popular instant messaging platforms in the past that have historically been extremely poor with respect to cryptographic security. Extremely popular messengers earlier in the 21st century such as Yahoo Messenger and Windows Live Messenger did not even provide basic encryption to its users [14] [15]. Although nearly all modern instant messaging platforms provide such basic security there are still certain platforms that do not use what seem like basic security features. Table 1 later in this paper makes note of a few such platforms. Section 4 also briefly highlights a couple of such platforms.

As mentioned briefly earlier, the aim of this research paper is to make an attempt in identifying the most secure instant messaging platform currently in the industry primarily in terms of cryptographic security (Section 6). In doing so we have also clearly defined what factors based on our research make an instant messaging platform secure (Section 2). Consequently, we have also listed out what instant messaging platforms fit our criterion (Section 3). Along the way we also discuss a few poorly implemented instant messaging platforms in terms of security to understand the potential pitfalls in cryptographic protocols (Section 4). We have explained in depth the security features implemented by the Signal protocol that we identify to be the most cryptographically secure instant messaging protocol currently in the industry based on our criteria (Section 5). Lastly, we briefly discuss some of the competitors to the Signal protocol (Section 7). Finally, we have concluded our findings from the research in a nutshell (Section 8). Based on our research we identify Signal Messenger to be the most secure instant messaging platform with WhatsApp and Viber almost being at par with Signal Messenger.

2 SECURTIY FEATURES

This section of the paper lists down the factors of what makes an instant messaging service secure based on our research. This section does not include essential security features: confidentiality, integrity and availability as we see these factors as being bare minimum requirements in an instant messaging platform given the current standards of cryptography in the IT industry.

2.1 Post Compromise Secrecy

Post compromise secrecy (PCS) briefly states that secrecy in future sessions should be maintained between the parties involved in a conversation despite a compromise of a key from a current session. One of the ways PCS is commonly achieved is by a mechanism called 'key continuity' wherein session keys are constantly updated as messages are being exchanged. Key continuity provides PCS and other security features and is used in protocols such as Signal protocol, Off-the-Record protocol (OTR) and MTProto 2.0 [47][4][67]. However, as is the case with many cryptographic constructions, the algorithm and the careful implementation of the algorithms is crucial for the desired security features. For instance, platforms such as TextSecure made use of a key continuity algorithm that had flaws as proven by Cohn-Gordon et. al [19]. Signal protocol that claims to provide PCS through key continuity has been described in detail later in this paper.

PCS is crucial in an instant messaging platform as it improves the security of the platform. Additionally, it acts as a secondary line of defense in case a user's long-term key is revealed or extracted by an adversary. This does not mean that keys should be leaked but if they do get leaked then there is a security system in place to ensure that secure communication can continue between the respective parties. However, this security feature by itself does not prevent the adversary from finding out the previous keys that have been exchanged given access to a current key at a certain time during a protocol [16]. To solve this issue there is a very closely related feature to PCS called perfect forward secrecy which is explained in the following section.

2.2 Perfect Forward Secrecy

Perfect forward secrecy is a security feature that advocates for independency of all keys exchanged prior to a certain time in a protocol. This security feature is meant for protocols that rely on key continuity. More precisely, it states that given that a key of a user is somehow extracted by an adversary then this should not give the adversary the ability to decipher previously used keys of the same user. Perfect forward secrecy thereby ensures that messages exchanged in an instant messaging protocol over the internet are safely locked into the past. They will be safely locked into the past because this security feature advocates that there should be no computationally efficient algorithm to extract the keys exchanged once the communication has occurred between parties. This mechanism of constantly exchanging keys during an ongoing communication is carried out by

various instant messaging protocols that very often tend to use Diffie-Hellman algorithm or some variant of it such as extended triple Diffie-Hellman that is covered in detail later in this paper [17] [18].

Perfect Forward Secrecy (PFS) is important in instant messaging platforms as it adds a stronger layer of security over instant messaging security protocols which generally involve some sort of key continuity to conduct the various security primitives. By adding PFS to a protocol there is additional robustness included in a given protocol because past chat sessions cannot be decrypted just based on the compromise of the current key. Therefore, even if an adversary extracts a key at some point during a conversation, PFS ensure the conversations that took place earlier are kept secure.

Primary difference between PCS and PFS is the time period at which the compromise occurs with respect to the session that is going to be targeted by the adversary. In case of PFS upon compromise of the long-term key, the adversary looks to extract information from a chat session that has already occurred prior to the compromise. In case of PCS, the compromise of the key is followed by an attempt by the adversary to extract information from a future session after the compromise occurred. Hence, the name “post-compromise” secrecy and it is also considered to be a stronger security feature than PFS [20].

2.3 End-to-End Encryption

End-to-End encryption is a security feature that ensures messages being transmitted from one party to another are securely encrypted throughout the transmission between the parties. Moreover, it advocates that plaintexts of messages only become visible once they enter the application layer of only the respective parties involved in communication. Thereby, no third party including the instant messaging platform itself may have access to the message being transmitted between the parties [21]. This security feature ensures that confidentiality of messages is always maintained as messages travel across the internet from the sender to the receiver. The flip side of end-to-end encryption is ‘encryption in transit’ where the service provider could potentially decrypt the encrypted messages being transferred amongst its users. The fact that the instant messaging platform itself cannot decrypt the messages being encrypted across its own platform is very powerful in terms of security. Therefore, users can be completely assured that communication taking place on an instant messaging platform is as secure as communicating in a private setting with no third-party intervention whatsoever.

Encryption by itself is perhaps the most basic and essential security feature that an instant messaging platform should have because it ensures that no one in the public internet can simply sniff the internet packets and learn about the communication taking place. In simpler words encryption provides confidentiality of messages. Having encryption alone is not enough, it is crucial for an instant messaging service to be making use of standardized encryption protocols to ensure

encryption is not just happening for the sake of it. Updating encryption protocols as there is more research published with regards to encryption protocols becomes important for such platforms. For instance, the fairly recent birthday attack published on 64-bit block ciphers such as 3DES and Blowfish illustrate the need to be adaptive with regards to encryption protocols [22]. Adding end-to-end to encryption only enhances the basic notion of encryption by itself as there is added secrecy in communication to an extent that even the service provider is unable to decipher the communication. Therefore, it is fair to establish that end-to-end encryption is a critical security feature in an instant messaging platform.

An interesting caveat in end-to-end encryption is that although some instant messaging services provide this feature, but it might not be a feature that is fully supported or an optional feature in their service that is not provided by default. A breakdown of services that have such drawbacks in their end-to-end encryption along with other features is provided later in this paper in Table 1. For instance, Facebook Messenger provides end-to-end encryption but it is not supported in their website platform and is left as an optional feature wherein users have to willingly turn on this feature when communicating [23].

2.4 Data Storage

As text messages, voice notes, photos, videos and other such media is exchanged between users in instant messaging platforms, there is a large accumulation of data that needs to be stored somewhere. Perhaps the three most obvious places to store such data would be on the user's local storage, server of the instant messaging platform or some other third-party storage. In terms of security the most secure storage space of these three would be local storage because it mimics a real-world communication in a private setting scenario wherein only the parties involved in the conversation register the conversation. Therefore, it makes perfect security sense to have message data from instant messaging platform be stored in the local storage of the parties involved in communication and no other storage space. With the advent of cloud storage, a user may choose to save their data in the cloud which comes at the risk of security. This has become even more prevalent as quite a few instant messaging platforms offer their users to store their back up in the cloud. Platforms that offer users cloud back up are LINE, WhatsApp, WeChat and more. We briefly talk about cloud storage in instant messaging platforms later in this paper.

However, there is a caveat in the real-world scenario mentioned above because its infeasible in case of instant messaging platforms to completely mimic a real-world setting. Primarily because in case of instant messaging platforms, messages are exchanged via the internet and the given parties may not all be connected to the internet at the same time. This is known as an 'asynchronous setting' in the instant messaging domain. This scenario is analogous to the real world setting where all parties are not in the same room where the conversation is taking place. Thus, this problem of having a storage space apart from the local storage of the user introduces the need to have a secondary storage space. Put simply, messages sent from Alice to Bob who let's say is not connected to the

internet (perfectly practical scenario). Messages cannot be stored in Bob's storage because he is not connected to the internet and so cannot fetch the messages sent from Alice. Hence, the only other realistic secure option is to store such messages in the platform's server that is always connected to the internet. Therefore, there is always bound to be some third-party storage space that would be required which often is the instant messaging platform's servers. In an ideal scenario, instant messaging platforms should have minimal data storage of user's metadata and their messages exchanged through the particular instant messaging platform. This is ideal because it ensures utmost security of data as only the parties involved in communication are in hold of their data, identical to a secure real-world communication.

Data storage matters in security of instant messaging platforms because it determines who owns or is accountable for the messages being exchanged. As mentioned earlier a platform that pays attention to security would advocate for local storage of data on the respective devices of the users. This local storage of messages does not necessarily guarantee full security as data can still potentially be retrieved by adversaries from a device's local storage [24]. Nonetheless, local storage at least transfers the accountability of data privacy to the users themselves and not the instant messaging platform provider. Therefore, an instant messaging platform to ensure maximum security with regards to data storage should be storing messages in the local devices of users. Hence, a security centric instant messaging platform would make minimal use of their own servers for data storage of their users.

It is important to note that by storing data locally on the user's devices there is a tradeoff between user's ease of accessibility and security. Storing data locally implies that messages are stored in the user's device that uses the platform and not the platform's server. Therefore, users will not be able to carry on their conversations from multiple devices as their messages are not stored in a server from where another device can fetch messages. Although having data stored in the local devices adds a layer of security it comes at the cost of users ease of accessibility. There are mechanisms to navigate this issue such as mirroring the primary device's data to another device which is a mechanism adopted by WhatsApp's desktop/web applications and many other instant messaging platforms.

.

2.5 Open-Source Code

Providing source code access of instant messaging protocols implemented to the public is considered to be a preferred practice in terms of security. This is because it allows for public peer review that can potentially identify a security flaw in a protocol or its implementation. Absence of public peer review leaves the entire responsibility of a protocol's security on a handful of employees of the instant messaging platform. Moreover, this practice also abides by Kerchoff's principle which is widely accepted and advocates for not providing security through obscurity but instead transparency [25][26]. Open-source code is important in instant messaging services as being able

to provide optimum security despite having an open-source code, further validates the security measures taken by a particular platform. Although this feature by itself does not mitigate any potential security flaws in a platform, it does however make the security of the platform more robust as it is open to public review.

3. SECURITY FEATURES OFFERED BY VARIOUS INSTANT MESSAGING PLATFORMS

Table 1 in the following page contains a list of popular instant messaging platforms as per Hootsuite's Digital Global report of 2020 and some other prevalent instant messaging platforms in North America based on our experiences (iMessage, Google Hangout, Signal and Slack) [45]. We choose to focus on the most popular platforms globally to narrow down our research and to be able to address security concerns of platforms used by a wide range of people all over the world. The following table is by no means intended to be an exhaustive list of all the instant messaging platforms in the industry. Similarly, the table does not list down all the security features of an instant messaging platform but just the features we have set as criterion for purpose of this research paper. Security is not necessarily a top priority for all instant messaging platforms and there are numerous other objectives that a platform seeks to provide its users. Table 1 is a good indication of how various instant messaging platforms all around the world are keeping up with the developments in cryptographic security.

Moreover, Table 1 clearly illustrates how various instant messaging platforms offer different security features that we consider to be essential in terms of the security of an instant messaging platform. Certain platforms such as WeChat, Slack and Google Hangout do not even provide end-to-end encryption which is the most well-established feature in the industry out of the 6 features listed in the table. However, it is important to note that platforms such as Slack whose target audience are teams within middle to large scale organizations probably would want to give users the ability to connect to their teams via multiple devices. Thereby, local data storage as a security measure comes at the cost of the platform's intended usage which is meant for easy connectivity among team members. Hence, this table must not be taken in face value as not all platforms in table 1 are designed to target the same audience. Nonetheless, Slack's design decision of not advocating for local storage and other platforms that don't provide local storage of messages to its users does make them less secure relatively.

***Table 1**: Instant Messaging Platforms and Security Features offered by them [** Unknown, ~ Partial, * Some features]*

Instant Messaging Platforms	Forward Secrecy	Post Compromise Security	End-to-end Encryption	Default End-to-End encryption	Local Data Storage	Open-Source Code	Security protocol used
Facebook Messenger [27]	~	~	✓	×	×	×	Partial Signal Protocol*
Google Hangout [28]	**	**	×	×	×	×	Undisclosed
iMessage [29]	**	**	✓	✓	✓	×	Undisclosed
IMO [30]	×	×	✓	×	×	×	Undisclosed
Line [31][32][42]	✓	×	✓	×	×	×	SPDY 2.0 [x]
Signal Messenger [33]	✓	✓	✓	✓	✓	✓	Signal Protocol
Skype [34]	~	~	✓	×	×	×	Partial Signal Protocol*
Slack [35]	×	×	×	×	×	×	Undisclosed
Telegram [36] [37]	✓	×	✓	×	✓	✓	MTProto
Viber [38]	✓	✓	✓	✓	✓	×	Based on Signal Protocol
WeChat [39][40]	×	×	×	×	✓	×	Undisclosed **
WhatsApp [41]	✓	✓	✓	✓	✓	×	Signal Protocol

4. POORLY IMPLEMENTED PROTOCOLS

Before delving deep into secure instant messaging platforms, it is important to highlight some instant messaging platforms that have history of not being secure but are still popular for various other reasons. On top of this list is Apple's iMessage that is popular particularly in North America. iMessage's messaging protocol was among the first few messaging protocols in the industry designed specifically keeping end-to-end encryption in mind. Apple's iMessage protocol is not publicly accessible and thereby publicly available information regarding the protocol is very limited. Using a protocol that is not extensively peer reviewed is not the safest practice in cryptography given there is always going to be scope of a security flaw. Apple claims that iMessage is end-to-end encrypted which consequently implies that even Apple cannot access the messages being exchanged by users in its own service. There have been successful attacks against Apple's iMessage protocol in the past such as the chosen ciphertext attack demonstrated by Garman et. al [43]. What makes iMessage even more susceptible is that it also integrates SMS services under the same iMessage application on iOS devices. SMS are a completely different category of instant messages that operate under a different protocol and mechanism to that of Apple's iMessage. Thereby, SMS on iMessages do not even necessarily provide end-to-end encryption despite Apple's claims of iMessage providing end-to-end encryption. While the integration of SMS in iMessage is perhaps done keeping user friendliness in mind. However, it keeps an average user oblivious to whether their messages are being end-to-end encrypted or not. Since iMessage does not explicitly mention anything about end-to-end encryption in their service at the time of use. The only visible difference between SMS and messages sent over iMessage is the color of the message box. SMS are green in color and iMessages are blue in color, but an average user is not made aware of how the end-to-end encryption differs in the two categories.

When discussing instant messaging platforms, it is important to take a note of SMS. Short Message Service commonly referred to as SMS is a popular form of communication between mobile phones as it is practically available to everyone that has a mobile phone connection. However, SMS are completely different to the instant messaging platforms mentioned so far in this paper. The primary difference is that SMS messages are exchanged over cellular networks and not through internet protocols as is the case with all the platforms we have spoken about earlier in this paper. Additionally, SMS are designed in a way such that providing basic security primitives such as encryption and authentication is kept optional [44]. Which implies that SMS are by no means end-to-end encrypted and can potentially be decrypted by telecommunication services offering SMS service over their network.

5. SIGNAL PROTOCOL

Based on our research and the criterion mentioned in section 2, Signal Messenger followed by WhatsApp and Viber seem to be the most secure instant messaging platforms. These three services

provide most of the security criteria we have listed in section 2. Interestingly, these three platforms make use of the same cryptographic messaging protocol developed by Open Whisper Systems, called: Signal Protocol. However, Viber doesn't necessarily use the exact signal protocol, but a custom protocol largely inspired by Signal protocol [38]. There are certain differences between these three platforms such as WhatsApp's source code is not open to the public while Signal's source code public. Viber as mentioned earlier has built its own cryptographic messaging protocol inspired by Signal while both WhatsApp and Signal use the Signal protocol without customization. The top three platforms identified based on our research derive their cryptographic security entirely from the protocol they use in their platforms. Therefore, instead of focusing on just one of these platforms we have decided to focus on the Signal protocol by itself.

Signal Protocol developed by Open Whisper System and was released in 2013 with the intention of providing a cryptographic messaging protocol that provides premium security features such as end-to-end encryption, forward secrecy as well as post compromise secrecy [21]. What makes Signal more secure than previous protocols based on Pretty Good Privacy (PGP) that provided perpetual key exchanges is that Signal provides ephemeral key exchanges for each session instead [46]. This is the 'key continuity' mechanism we mentioned earlier in section 2. The following passages of this paper describe in detail some of the noble features within the Signal protocol.

5.1 Double Ratchet Algorithm

Is an algorithm in the Signal protocol whose objective is to carry out an exchange of encrypted messages between parties based on a shared key acquired and generated by the Diffie-Hellman algorithm [47]. 'Ratcheting' by itself as a concept was first introduced by Off-the-Record messaging protocol (OTR). In simple words, ratcheting refers to an exchange of a shared secret that is constantly updated as messages are exchanged between two parties [20]. In the case of the Signal protocol this shared secret is a pair of keys produced by the Diffie-Helman algorithm.

Signal protocol combines two ratchets to create what is known as a 'double ratchet', one of these ratchets is created by Diffie-Helman ratchet algorithm and the other ratchet is created by the symmetric key ratchet algorithm. Before we go into finer details of how both these algorithms combine to produce this double ratchet notion in the following paragraphs it is important to understand KDF chains. KDF chains are used a fair amount in the Signal protocol. KDF chains are cryptographic functions that output some data given an input and a KDF key. This output generated by KDF function is indistinguishable from a random bit string of the same length, KDF functions are cryptographically speaking Pseudo-Random Functions. Both parties involved in a certain conversation maintain KDF chains such as the root chain, sending chain and receiving chain. We attempt to describe how the Symmetric key ratchet and Diffie-Hellman ratchet below.

Symmetric key ratchet algorithm as previously alluded to is one part of this double ratchet algorithm. Symmetric key algorithm is where the encryption and decryption of messages in this

protocol takes place. All messages exchanged are encrypted with a unique message key. These unique message keys are derived from the sending and receiving KDF chains' outputs. In a conversation between Alice and Bob, the sending chain of Alice corresponds to the receiving chain of Bob and similarly for Bob. This is where the 'symmetric' notion of this algorithm originates from as both Alice and Bob share the same KDF chains that are responsible for creating the same unique message keys for both parties to facilitate encryption and decryption. It is important to note that chain keys are not message keys but instead chain keys are what help in creating 'unique' message keys. Additionally, this symmetric key algorithm also derives the next KDF chain keys for the sending and receiving KDF chains respectively as the conversation takes place.

The Diffie-Hellman ratchet algorithm is the second part of this Double Ratchet algorithm and its primary role is to update the KDF keys used in the three KDF chains: root, sending and receiving every time message is exchanged. This is done so that a potential compromise of the KDF chain keys does not lead to a compromise of the messages being exchanged. The KDF keys are updated based on the standard Diffie-Hellman algorithm. All messages exchanged consist of the sender's current public key and upon receiving a message the receiver computes and updates the Diffie Hellman key pair for itself. Then subsequently when this receiver responds to the sender, it follows the same procedure as the original sender performed and this back and forth updating of Diffie Hellman key pair continues between the two parties as long as they exchange messages.

Diffie-Hellman outputs are not directly used in the sending and receiving KDF chains but instead they are used as the key to the root KDF chain that then sequentially creates the KDF keys for the sending and receiving chains. Therefore, this is the chain like notion enabled by KDF chains that was previously alluded to in this section. To better explain this Diffie-Hellman exchange let's take for example a conversation that Alice initiates with Bob. Alice would share her public key by sending a message to Bob and upon receiving this public key, Bob would calculate it with his private key and then update this public key he received. After updating this pair, he would then send a new public key along with his message when he responds to Alice at a later stage. Alice would then perform the same steps as Bob did when he received the first message from Alice. This back and forth continues to happen between the two parties as they exchange messages. In simpler words, the shared secret generated by Diffie-Hellman become inputs for the root KDF chain whose output then sequentially becomes the KDF keys for the subsequent chains (sending and receiving) outputs respectively [47].

Thereby, Diffie-Hellman ratchet and Symmetric key ratchet combine to create the Double Ratchet algorithm that creates a constantly updating key exchange mechanism between the sender and receiver. Entropy in this algorithm largely originates from updating the Diffie-Hellman key pairs that are generated in a "ping-pong" manner as messages are exchanged. Both parties continuously create new key pairs as messages are exchanged and this also ensures that perfect future secrecy and post compromise security are achieved. This happens because compromise of a certain key pair at some time period will not have an effect on previously generated keys nor keys generated in the future as they are completely independent of each other [20].

5.2 Extended Triple Diffie-Hellman

This is a key exchange algorithm in the Signal protocol that is based upon the principles of Diffie-Hellman as the name suggests. X3DH uses either curve X25519 or curve X448 for its implementation [48][49]. Moreover, this algorithm must make use of a server because of how it is structured and because of the asynchronous nature of instant messaging i.e., a user should be able to message another user regardless of whether they are online or offline. This notion of having a server has been briefly alluded to earlier in section 2 of this paper.

There are several public and private key pairs in the Signal protocol such as identity key, ephemeral key, identity key, signed prekey and one-time prekey. The X3DH algorithm can be segregated into three parts, firstly, all registered users have to store an identity key, signed prekey and a set of one-time prekey into the server. All these keys are required to initiate a conversation. Identity keys are stored in the server just once but signed prekey and the set of one-time prekeys are updated periodically depending on various factors. Along with these prekeys there is also an ephemeral key that each user makes use of to compute the 'superkey' as is explained below.

Secondly, when initiating a conversation, the sender must be able to access the stored identity key, one-time prekey and signed prekey of the user it wishes to contact. These keys can be accessed by a user as they are stored in the platform's server. A 'superkey' is generated based on the set of keys of both the sender and receiver. Eventually, the sender sends its initial key and its ephemeral key to the receiver along with an encrypted message termed as 'associate data' to serve as the initial data sent from the sender. 'Superkey' is calculated by concatenating the results of applying the Diffie-Hellman algorithm to 4 different pairs of keys between the sender and receiver (3 pairs if the one-time pre key is not available). The equations below state the 4 equations used to calculate the 'superkey' from the sender's end.

Equation. *To calculate the superkey shared between the sender and receiver in X3DH. 'DH' refers to Diffie-Hellman*

DH1 = DH (Initial Key of sender, Signed Prekey of receiver)

DH2 = DH (Ephemeral Key of sender, Initial Key of receiver)

DH3 = DH (Ephemeral Key of sender, Signed Prekey of receiver)

DH4 = DH (Ephemeral Key of sender, One Time Prekey of receiver)

SuperKey = KDF (DH1 || DH2 || DH3 || DH4)

Lastly, the receiver upon receiving the initial message also computes the 'superkey' at its own end. 'Superkey' is derived based on the initial key, ephemeral key, signed prekey and one-time prekey as illustrated in the equations above. 'Superkey' can be derived at the receiver's end because the sender sends its own identity and ephemeral key along with a ciphertext and the first message. Therefore, both the sender and receiver have the same set of keys and can thereby compute the symmetrical 'superkey'. The 'associated data' is decrypted using the 'superkey' that is calculated. If the 'associate data' ciphertext fails to decrypt at the receiver's end, then the communication is aborted. On the contrary, if the ciphertext is decrypted successfully then the X3DH protocol between the sender and receiver has been completed and both parties now share a secret key, namely: 'superkey' [18].

5.3 Signing & Verifying algorithms

The following few paragraphs attempt to give a brief introduction to the data integrity algorithms used by Signal protocol. We refrain from going into the granular details of them as it is out of the scope of this paper. XEdDSA and VXEdDSA are the two data integrity algorithms that Signal protocol uses and they are primarily based on EdDSA signatures which are by themselves based on elliptic curves [50][51]. Let us start with XEdDSA, it comprises of three inputs: a Montgomery private key, message and a random 64-byte sequence. The Montgomery keys in this algorithm have compatible geometrical properties to Edwards curves as proven by Daniel et. al [52]. XEdDSA algorithms also makes use of a hashing algorithm and for this purpose they use SHA-512 which is proven to be secure by the FIPS publication released in 2012 [53]. XEdDSA does a fair amount of calculations and ends up by outputting the concatenation of two derived values from the computations within the algorithm. The verification algorithm takes as input the signature, message and a Montgomery public key and makes use of elliptic curve and Montgomery key properties to verify the signature.

VXEdDSA is simply an extension and thereby very similar to XEdDSA. It also has the same inputs as the XEdDSA but it outputs a more complex signature to XEdDSA. It uses the same hash function, but it computes the hash of more values within the algorithm and makes relatively more extensive use of the geometrical properties of Elliptic curves to derive its signature. The verification algorithm takes the same inputs as XEdDSA's verification algorithm. However, the algorithm itself has certain modifications as it needs to account for a relatively more complex signature to XEdDSA. If successful the verification algorithm outputs a byte sequence instead of a standard boolean value of true, upon failure it simply outputs false. The value outputted by the verification algorithm upon success is termed as the 'verifiable random function' which as the name suggests is a random function that is proven to be indistinguishable to random [54]. The exact specifications of the algorithms can be found in Signal's documentation [55].

5.4 Noble Features of the Signal Protocol

X3DH and Double Ratchet algorithm together provide some of the noble features in the Signal Protocol such as perfect forward secrecy (PFS) and post compromise secrecy (PCS). This happens primarily because of the 'ping pong' like behavior which is created by the Double Ratchet algorithm facilitated by X3DH. The ratcheting mechanism in the Signal protocol ensures that earlier computed keys that are not required for future correspondence are deleted [19]. Double Ratchet protocol describes the exchange of encrypted messages between the sender and receiver while X3DH protocol describes the calculation of a shared secret: 'superkey'. However, there is one other algorithm within the Signal protocol: Sesame. Details about the Sesame protocol have been left out of this paper because it does not directly concern the cryptographic security measures within the Signal protocol. The Sesame algorithm deals with the management of maintaining a solitary end-to-end encrypted session between two parties communicating across various devices that they might use their Signal enabled instant messaging platform on. Hence, Sesame protocol is designed to handle the logistics of the sessions and not necessarily facilitate the security component of the Signal protocol. Although the XEdDSA and VXEdDSA algorithms do have an impact on the cryptographic security of the protocol to an extent, these algorithms are not the main factors in providing the Signal protocol with the noble security features highlighted in this paper.

6. MOST SECURE INSTANT MESSAGING PLATFORM

The three platforms from table 1 that base their cryptographic security on the Signal protocol are: WhatsApp, Viber, and Signal Messenger. Other services such as Skype and Facebook messenger also make use of the Signal protocol, but it is not currently their default option. More specifically, in Skype a feature called 'private conversation' and in Facebook messenger a feature called 'secret conversation' need to be turned on for Signal's protocol to be used in these services [34]. WhatsApp clearly mentions in its technical white paper that it makes use of the Signal protocol directly into its instant messaging platform and the documentation very closely resembles all the components of the Signal protocol [41]. Viber on the other hand explicitly mentions that its protocol is largely inspired by Signal but has been created by Viber themselves from scratch, but it has very negligible differences to the original Signal protocol. For instance, the calculation of the 'superkey' in Viber's protocol makes use of only 3 Diffie Hellman calculations while the standard Signal protocol and also WhatsApp's signal protocol make use of 3 or 4 depending on if the one-time prekey of the user is available. There isn't a big scope to modify the Signal protocol by these platforms because they want to equip their services with noble features such as post-compromise secrecy and perfect forward secrecy. Massively modifying the algorithms creates an unwanted risk of a potential security flaw. Therefore, there is very little modification done to the cryptographic security aspects (Double ratchet and X3DH) of the Signal protocol used by WhatsApp and Viber. Furthermore, Cohn et. al's formal security analysis of the Signal protocol where they prove the security of the protocol further validates the security principles of the Signal protocol [20]. More formally, they state that the Signal protocol

is a secure protocol assuming the KDF primitive is a random oracle and under the GDH assumption [56][20].

Therefore, based on our research and our criterion laid out in section 2, we identify Signal Messenger to be the most secure instant messaging platform currently in the industry. Signal Messenger is developed by the same organization that developed the Signal protocol that has been covered in this research paper. However, it would be naive to ignore platforms that have a large user database as that provides greater accessibility to a user. With this in regard, WhatsApp is a clear front runner because of its wide popularity in the current global market. WhatsApp provides nearly all the security features that Signal Messenger does apart from being open source and enabling backup of messages in cloud services. Amongst the most popular instant messaging platforms, WhatsApp is the most cryptographically secure service that offers default end-to-end encryption, post-compromise security, perfect forward secrecy and local data storage. Having said that, considering the scope of this paper we do not factor in non-security features such as popularity or user interface and thereby we claim that WhatsApp is still inferior to Signal Messenger in terms of security. The following passage contains the reasons behind our claim.

The couple of security aspects going against WhatsApp is the fact that it is not open source and thereby suffers from genuine credibility on whether they truly implement the security features that they claim to implement. The other security flaw in WhatsApp is that it allows its users to back up their data on cloud services such as Apple's iCloud or Google Drive [57][58]. Data backed up in these cloud services is not end-to-end encrypted like how it is within the WhatsApp platform. Thereby, these cloud service vendors can potentially extract personal conversations that occurred in WhatsApp that provides end-to-end encryption. It is important to note that backing up conversations to the cloud is not a default configuration but an optional feature in WhatsApp. On the contrary, Signal Messenger does not allow for cloud backup of messages exchanged through its platform. Signal Messenger strictly saves all the message history local to the device and does not offer a cloud backup option to its users [59].

7. COMPETITORS TO SIGNAL MESSENGER

We have explained in detail about the security features offered by the Signal protocol but that does not imply that there are no other alternatives to this protocol in instant messaging platforms. This section of the paper will briefly outline some of these other competitors in the industry and how they differ from the Signal protocol.

Wire Messenger is one such instant messaging platform that primarily focuses on providing the most secure communication between parties. Its entire business model and platform seems to revolve around their attention to detail regarding the security of their users. They use a protocol called 'Proteus protocol' that is an implementation of the Double Ratchet algorithm from Open

Whisper System described in the previous section [60]. It makes a few minor modifications to the original Double Ratchet algorithm but nonetheless it provides perfect future secrecy along with post-compromise security. One notable feature of the Wire messenger is that it provides each user in a conversation the ability to personally verify the identity of the other party. The user interface of a conversation is equipped with a blue shield that denotes if the conversation taking place is verified with respect to the identities of each of the parties [61]. Moreover, this messaging service is a 100% open source and is thereby open to peer review which as we identified earlier in this paper is a noteworthy security feature by itself [62].

Another instant messaging platform that is renowned for its security is Telegram Messenger. Two issues in terms of cryptographic security in this platform are: default end-to-end encryption and lack of external validation of their security protocol. Users must use a feature called ‘secret chats’ to be able to communicate over an end-to-end encryption channel. End-to-end encryption in ‘secret chats’ is provided by a custom protocol created by Telegram Messenger Inc. called: MTProto. There have been 2 versions of this protocol released since its inception, we will be focusing on the latest version currently: 2.0 [67]. The key sharing algorithm used in MTProto 2.0 is a more standard version of the Diffie-Hellman algorithm. It does not modify the algorithm a great deal in comparison to the X3DH (Triple Extended Diffie-Hellman) protocol explained earlier in this paper. The MTProto 2.0 protocol also provides perfect forward secrecy although not in a similar manner to the Double Ratchet algorithm that has been explained earlier. MTProto exchanges what they call ‘special messages’ to perform forward secrecy, using these so called ‘special messages’ both the sender and receiver generate new keys using their previous keys. They do this periodically which eventually leads to perfect forward secrecy [36]. One of the primary differences between the Signal protocol and MTProto is that Signal integrates the exchange of message keys within the exchange of messages between parties. Signal protocol generates new pairs of keys for both the sender and receiver during each Diffie-Hellman ratchet step as explained earlier. This ratchet occurs every time a party sends a new message. On the contrary MTProto shares message keys with respect to time or a certain number of messages exchanged. More specifically MTProto automatically generates new keys and destroys previous keys after more than 100 messages or after one week of use of a key pair as long as the key pair has at least been used once.

MTProto is a protocol whose previous version in the past has been proven to not be secure against certain cryptographic standards such as indistinguishability under a chosen ciphertext attack and authenticated encryption as proven by Jakob et. al [68]. MTProto 1.0 suffered from the fact that during decryption the padding was not checked for integrity. Hence, a potential attacker could come up with new ciphertexts that decrypt to the same message as the original ciphertext. This security flaw was rectified by Telegram in their current MTProto version 2.0. There were a few other security flaws in version 1.0 such as using SHA-1 which was already known to be broken even at the time of launch of MTProto version 1.0 [69]. The current version 2.0 of MTProto is relatively new to the other competing protocols in the industry. Therefore, there needs to be more concrete external research

done on the cryptographic aspects of MTProto 2.0 before it can be established in the industry and give a stronger competition to the Signal protocol.

Another competitor in the industry is the Element Messenger (previously called Riot) that is associated with the Matrix ecosystem. Matrix is primarily an open communication protocol developed in the United Kingdom and France. Matrix protocol by itself is designed with the intention of providing a central HTTP messaging system for various devices on the internet [63]. It can consequently be used in instant messaging platforms such as Element. Element messenger is a product developed by the same team behind the Matrix protocol. It also just like Wire and Telegram Messengers runs under the objective of creating a secure platform for communication over the internet. It makes use of the Matrix protocol for key sharing and is based upon the 'Olm' and 'Megaolm' protocols for end to end encryption of messages [64]. Olm is once again another implementation of the Open Whisper System's Double Ratchet algorithm that we have already described earlier in this paper [65]. Therefore, the OLM protocol used in Element Messenger provides perfect forward secrecy as well as post-compromise security. Megaolm on the contrary is not an implementation of the Double Ratchet algorithm instead it is a protocol designed for group communication. While it facilitates a better design for group messages it comes at the cost of security. Megaolm also uses the ratchet mechanism as part of its protocol to achieve 'key continuity'. It only manages to provide partial forward secrecy and lacks post-compromise security because of how the algorithm is designed to facilitate group communication efficiently [66].

Very little academic external research has been done on these different protocols and implementations mentioned above. Therefore, it's hard to evaluate them against each other. Given how things stand in the instant messaging domain currently, Signal protocol for the time being seems to be the preferred protocol for the various noble features it provides as mentioned earlier. Moreover, the protocol is being continuously recognized as popular instant messaging platforms such as WhatsApp, Facebook Messenger, Skype and Viber are starting to make use of it in their own services. The Signal protocol implements end-to-end encryption and the more recent security measures such as post-compromise security and perfect forward secrecy. Furthermore, given the external academic research done on the protocol and the fact that it can be publicly peer reviewed, make it a protocol that offers modern day security measures for instant messaging platforms to use.

8. CONCLUSION

As can be seen by the analysis of the instant messaging platforms in this research paper that most platforms concerned about security tend to use the Signal Protocol or an implementation of it. Such as WhatsApp, Wire, Viber and Element Messengers and some platforms use it to an extent such as Facebook Messenger and Skype. Signal protocol is publicly available and provides modern day security features such as perfect forward secrecy and post-compromise security. As the scope of the internet grows further, consequently the need for security is likely to become more of a concern

in the near future. Therefore, secure messaging platforms might well be in high demand as they provide an essential necessity for human beings: communication. Going by historic trends the need for secure instant messaging platforms would grow exponentially if a major breach were to be disclosed in instant messaging platforms as was the case after Snowden's discoveries in 2013-2014 [70]. Snowden's discoveries led to a rapid acceleration in cyber surveillance methods and something similar could happen in the case of instant messaging platforms more specifically. Therefore, the scope of such instant messaging platforms is massive and potential security breaches could potentially change popularity of certain instant messaging platforms as customers become more aware of security. However, as things stand, Signal protocol is the frontrunner with regards to cryptographic security as it provides the latest noble security features in the instant messaging domain. One can't help but imagine that there could be a constant need for such developments to be made in this domain to maintain top notch security. Subsequently, it becomes essential for external academic researchers to further evaluate the security of the new security protocols in the industry.

In conclusion, we have analysed some of the security features offered by popular instant messaging platforms in the globe. We have also defined based on our research the security features that are necessary for these modern-day instant messaging platforms. In doing so we have been able to identify Signal Messenger to be the most secure Messenger service in the industry. Viber and WhatsApp come very close to Signal but due to their implementation details and other security features they fail to outperform Signal Messenger in terms of security. Additionally, in this paper we described essential details about the Signal protocol which is a frontrunner in terms of cryptographic security of instant messaging platforms. Hence, we present a comprehensive security analysis of several instant messaging platforms with a special focus on cryptographic security, we also describe the competing protocols and platforms that are trying to provide the best security in the industry.

REFERENCES

[1] J. Oikarinen, D. Reed. 1993. RFC 1459 – Internet Relay Chat Protocol. Retrieved November 11, 2020 from https://www.hjp.at/doc/rfc/rfc1459.html

[2] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk and T. Holz. 2016. How Secure is TextSecure? 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrucken, pp. 457-472, doi: https://doi.org/10.1109/EuroSP.2016.41.

[3] P. Saint-Andre. 2011. RFC 6120 - Internet Engineering Task Force (IETF) - Extensible Messaging and Presence Protocol (XMPP). Retrieved October 25, 2020 from https://tools.ietf.org/html/rfc6120

[4] Ian Goldberg and the OTR Development Team. 2004. Off-the-Record Messaging Protocol version 2. Retrieved October 26, 2020 from https://otr.cypherpunks.ca/Protocol-v2-3.1.0.html

[5] Vinnie Moscaritolo, Gary Belvin, Phil Zimmermann. 2012. Silent Circle Instant Messaging Protocol Specification. Retrieved October 19, 2020 from https://netzpolitik.org/wp-upload/SCIMP-paper.pdf

[6] P. Zimmermann, A. Johnston, J. Callas. 2011. RFC 6189 - ZRTP: Media Path Key Agreement for Unicast Secure RTP. Retrieved October 19, 2020 from https://tools.ietf.org/html/rfc6189

[7] Signal Foundation. 2018. Signal Documentation. Retrieved October 20, 2020 from https://signal.org/docs/

[8] John Linn (BBNCC). 1987. RFC 989 - Privacy Enhancement for Internet Electronic Mail - Part I: Message Encipherment and Authentication Procedures. Retrieved October 18, 2020 from https://tools.ietf.org/html/rfc989

[9] B. Ramsdell, Ed. 1999. RFC 2633 - S/MIME Version 3 Message Specification. Retrieved October 29, 2020 from https://www.hjp.at/doc/rfc/rfc2633.html

[10] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer. 2007. RFC 4880 - OpenPGP Message Format https://tools.ietf.org/html/rfc4880

[11] Simson L. Garfinkel, David Margrave, Jeffrey I. Schiller, Erik Nordlander, and Robert C. Miller. 2005. How to make secure email easier to use. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05). Association for Computing Machinery, New York, NY, USA, 701–710. DOI:https://doi.org/10.1145/1054972.1055069

[12] Nikita Borisov, Ian Goldberg, and Eric Brewer. 2004. Off-the-record communication, or, why not to use PGP. In Proceedings of the 2004 ACM workshop on Privacy in the electronic society (WPES '04). Association for Computing Machinery, New York, NY, USA, 77–84. DOI:https://doi.org/10.1145/1029179.1029200

[13] Facebook, Inc. 2020. WhatsApp blog - Two Billion Users -- Connecting the World Privately. Retrieved November 30, 2020 from https://blog.whatsapp.com/two-billion-users-connecting-the-world-privately

[14] R. Movva, W. Lai. 1999. Instant Messaging and Presence Protocol - MSN Messenger Service 1.0 Protocol. Retrieved December 2, 2020 from https://tools.ietf.org/html/draft-movva-msn-messenger-protocol-00

[15] Alex Stamos. 2014. Yahoo Post - Status Update: Encryption at Yahoo. Retrieved December 1, 2020 from https://yahoo.tumblr.com/post/81529518520/status-update-encryption-at-yahoo

[16] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. 1996. Handbook of applied cryptography. CRC Press Inc.

[17] E. Rescorla. 1999. RFC 2631 - Diffie-Hellman Key Agreement Method. Retrieved December 5, 2020 from https://www.hjp.at/doc/rfc/rfc2631.html

[18] Moxie Marlinspike. 2016. Signal Documentation - The X3DH Key Agreement Protocol. Retrieved October 10, 2020 from https://signal.org/docs/specifications/x3dh/x3dh.pdf

[19] K. Cohn-Gordon, C. Cremers and L. Garratt. 2016. On Post-compromise Security. IEEE 29th Computer Security Foundations Symposium (CSF), Lisbon, 2016, pp. 164-178, https://doi.org/10.1109/CSF.2016.19.

[20] Cohn-Gordon, K., Cremers, C., Dowling, B. et al. 2020. A Formal Security Analysis of the Signal Messaging Protocol. J Cryptol 33. https://doi.org/10.1007/s00145-020-09360-1

[21] Ermoshina K., Musiani F., Halpin H. (2016) End-to-End Encrypted Messaging Protocols: An Overview. Bagnoli F. et al. (eds) Internet Science. INSCI 2016. Lecture Notes in Computer Science, vol 9934. Springer, Cham. doi: https://doi.org/10.1007/978-3-319-45982-0_22

[22] Karthikeyan Bhargavan and Gaëtan Leurent. 2016. On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA, 456–467. DOI:https://doi.org/10.1145/2976749.2978423

[23] Facebook, Inc. 2020. ‘Secret Conversations’ in Facebook Messenger. Retrieved December 11, 2020 from https://www.facebook.com/help/messenger-app/1084673321594605

[24] Rahim, Robbi, A. P. U. Siahaan, and R. Manikandan. 2018. Insecure Whatsapp Chat History, Data Storage and Proposed Security. International Journal of Pure and Applied Mathematics 119.16 (2018): 2481-2486.

[25] Auguste Kerckhoffs. 1883. ‘La cryptographie militaire’ - Journal des sciences militaires, vol. IX, pp. 5–38.

[26] S. Mrdovic and B. Perunicic. 2008. "Kerckhoffs' principle for intrusion detection" - The 13th International Telecommunications Network Strategy and Planning Symposium, Budapest, 2008, pp. 1-8, doi: https://doi.org/10.1109/NETWKS.2008.6231360.

[27] Facebook, Inc. 2016. Facebook News - Messenger Starts Testing End-to-End Encryption with Secret Conversations. Retrieved October 11, 2020 from https://about.fb.com/news/2016/07/messenger-starts-testing-end-to-end-encryption-with-secret-conversations/

[28] Google LLC. 2020. Google Hangout Help Center. Retrieved December 2, 2020 from https://support.google.com/hangouts/

[29] Apple Inc. 2019. iMessage and FaceTime & Privacy. Retrieved October 28, 2020 from https://support.apple.com/en-ca/HT209110

[30] Sudozai, K. & Saleem, Shahzad & Buchanan, William & Habib, Nisar & Zia, Haleemah. 2018. Forensics study of IMO call and chat app. Digital Investigation. doi: https://doi.org/10.1016/j.diin.2018.04.006.

[31] Line Corporation. 2020. ‘Letter Sealing’ help center. Retrieved October 28, 2020 from https://help.line.me/line/?contentId=50001520

[32] Line Corporation. 2016. Line Encryption – Technical White paper. Retrieved October 28, 2020 from https://scdn.line-apps.com/stf/linecorp/en/csr/line-encryption-whitepaper-ver1.0.pdf

[33] Signal Messenger LLC. 2020. GitHub repository - signalapp/Signal-Android. Retrieved October 28, 2020 from https://github.com/signalapp/Signal-Android

[34] Microsoft Corpration. 2020. Skype, ‘Private Conversations’ – Support Centre. Retrieved October 27, 2020 from https://support.skype.com/en/faq/fa34824/what-are-skype-private-conversations

[35] Slack. 2019. Security White Paper. Retrieved October 28, 2020 from https://a.slack-edge.com/80588/marketing/downloads/security/Security_White_Paper_2019.pdf

[36] Telegram Messenger Inc. 2020. Perfect Forward Secrecy. Retrieved October 28, 2020 from https://core.telegram.org/api/end-to-end/pfs

[37] Telegram Messenger Inc. 2020. GitHub repository – Telegram. Retrieved October 28, 2020 from https://github.com/DrKLO/Telegram

[38] Rakuten Viber. 2018. Viber Encryption Overview. Retrieved October 28, 2020 from https://www.viber.com/app/uploads/viber-encryption-overview.pdf

[39] Chen, Melanie, Lauren Clayberg, and Helen Li. 2018. Security in the Face of Censorship - Massachusetts Institute of Technology.

[40] Tencent Holdings Limited, WeChat. 2020. WeChat Privacy Policy. Retrieved October 28, 2020 from https://www.wechat.com/en/privacy_policy.html#pp_period

[41] Facebook, Inc. 2020. WhatsApp Security Technical White Paper. Retrieved October 31, 2020 from https://www.whatsapp.com/security/?lang=en

[42] Mike Belshe et al. for Google. 2011. SPDY Protocol. Retrieved November 18, 2020 from https://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft2

[43] Christina Garman, Matthew Green, Gabriel Kaptchuk, Ian Miers, and Michael Rushanan. 2016. Dancing on the lip of the volcano: chosen ciphertext attacks on apple iMessage. Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16). USENIX Association, USA, 655–672.

[44] S. Ariffi, R. Mahmod, R. Rahmat and N. A. Idris, "SMS Encryption Using 3D-AES Block Cipher on Android Message Application," 2013 International Conference on Advanced Computer Science Applications and Technologies, Kuching, 2013, pp. 310-314, doi: https://doi.org/10.1109/ACSAT.2013.68

[45] Hootsuite. 2020. Digital 2020 Report. Retrieved November 24, 2020 from https://hootsuite.com/resources/digital-2020

[46] Moxie Marlinspike. 2013. Signal Blog - Advanced cryptographic ratcheting. Retrieved October 10, 2020 from https://signal.org/blog/advanced-ratcheting/

[47] Moxie Marlinspike. 2016. Signal Documentation - The Double Ratchet Algorithm. Retrieved October 10, 2020 from https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf

[48] Hamburg, Mike. 2015. Ed448-Goldilocks, a new elliptic curve. IACR Cryptol. ePrint Arch. (2015): 625.

[49] Bernstein D.J. 2006 Curve25519: New Diffie-Hellman Speed Records. In: Yung M., Dodis Y., Kiayias A., Malkin T. (eds) Public Key Cryptography - PKC 2006. Lecture Notes in Computer Science, vol 3958. https://doi.org/10.1007/11745853_14

[50] Bernstein, Daniel J., et al. 2015. EdDSA for more curves. Cryptology ePrint Archive 2015.

[51] A. Langley, M. Hamburg, S. Turner. 2016. RFC 7748 - Elliptic Curves for Security. Retrieved November 20, 2020 from https://www.ietf.org/rfc/rfc7748.txt

[52] Bernstein, Daniel J., et al. 2008. Twisted edwards curves. International Conference on Cryptology in Africa. Springer, Berlin, Heidelberg, 2008.

[53] The National Institute of Standards and Technology. 2015. Secure Hash Standard (SHS). Retrieved November 11, 2020 from https://csrc.nist.gov/csrc/media/publications/fips/180/4/final/documents/fips180-4-draft-aug2014.pdf

[54] Micali, Silvio, Michael Rabin, and Salil Vadhan. 1999. Verifiable random functions. 40th annual symposium on foundations of computer science (cat. No. 99CB37039). IEEE., 1999.

[55] Trevor Perrin. 2016. Signal Documentation - The XEdDSA and VXEdDSA Signature Schemes. Retrieved October 9, 2020 from https://signal.org/docs/specifications/xeddsa/xeddsa.pdf

[56] Eli Biham, Dan Boneh, Omer Reingold. 1999. Breaking generalized Diffie–Hellman modulo a composite is no easier than factoring. Information Processing Letters, Volume 70, Issue 2. Pages 83-87. doi: https://doi.org/10.1016/S0020-0190(99)00047-2

[57] Facebook, Inc. WhatsApp. 2020. WhatsApp frequently asked questions – How to back up to iCloud. Retrieved November 8, 2020 from https://faq.whatsapp.com/iphone/chats/how-to-back-up-to-icloud

[58] Facebook, Inc. WhatsApp. 2020. WhatsApp frequently asked questions – How to back save your chat history. Retrieved November 8, 2020 from https://faq.whatsapp.com/android/chats/how-to-save-your-chat-history/

[59] Signal Messenger LLC. 2020. Signal Messenger Features – Backup and Restore Messages. Retrieved November 22, 2020 from https://support.signal.org/hc/en-us/articles/360007059752-Backup-and-Restore-Messages

[60] Wire Swiss. 2020. GitHub repository – wireapp/proteus. Retrieved December 13, 2020 from https://github.com/wireapp/proteus

[61] Wire Swiss. 2020. Wire Security Whitepaper. Retrieved December 11, 2020 from https://wire-docs.wire.com/download/Wire+Security+Whitepaper.pdf

[62] Wire Swiss. 2020. GitHub repository – wireapp/wire. Retrieved December 13, 2020 from https://github.com/wireapp/wire

[63] The matrix.org Foundation. 2020. Matrix protocol's frequently asked questions. Retrieved December 12, 2020 from https://matrix.org/faq/

[64] The matrix.org Foundation. 2020. GitHub repository – Matrix official documentation. Retrieved December 12, 2020 from https://github.com/matrix-org/matrix-doc/blob/master/specification/modules/end_to_end_encryption.rst

[65] Floris Hendriks. 2020. Analysis of key management in Matrix. Retrieved December 12, 2020 from https://www.cs.ru.nl/bachelors-theses/2020/Floris_Hendriks___4749294___Analysis_of_key_management_in_Matrix.pdf

[66] The matrix.org Foundation. 2020. GitLab repository – Megaolm group ratchet official documentation. Retrieved December 12, 2020 from https://gitlab.matrix.org/matrix-org/olm/-/blob/master/docs/megolm.md#the-megolm-ratchet-algorithm

[67] Telegram Messenger Inc. 2020. End-to-end encryption, Secret Chats. Retrieved October 28, 2020 from https://core.telegram.org/api/end-to-end

[68] Jakob Jakobsen and Claudio Orlandi. 2016. On the CCA (in)Security of MTProto. In Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM '16). Association for Computing Machinery, New York, NY, USA, 113–116. DOI: https://doi.org/10.1145/2994459.2994468

[69] Wang X., Yin Y.L., Yu H. 2005. Finding Collisions in the Full SHA-1. In: Shoup V. (eds) Advances in Cryptology – CRYPTO 2005. CRYPTO 2005. Lecture Notes in Computer Science, vol 3621. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11535218_2

[70] Zygmunt Bauman, Didier Bigo, Paulo Esteves, Elspeth Guild, Vivienne Jabri, David Lyon, R. B. J. Walker. 2014. After Snowden: Rethinking the Impact of Surveillance, International Political Sociology, Volume 8, Issue 2, June 2014, Pages 121–144, https://doi.org/10.1111/ips.12048