Analysis of End-to-End Encryption Implementation Across Different Meta's Applications

Salsabiila - 13522062[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522062@std.stei.itb.ac.id

***Abstract*— In today's digital communication landscape, the preservation of user privacy stands as a pressing concern, prompting the need for robust security protocols. Meta, a prominent tech entity, has undertaken the challenge of enhancing user security through the recent integration of end-to-end encryption (E2EE) as the default in its Messenger application across Facebook and Instagram. This paper delves into a comprehensive analysis, aiming to unveil the similarities and differences between Messenger's E2EE implementation and the more established E2EE model in Meta's WhatsApp. By examining these encryption mechanisms, the study provides insights into Meta's ongoing commitment to user privacy, unraveling the distinctive security features within each application and contributing valuable perspectives to the broader dialogue on digital privacy.**

***Keywords*—cryptography, encryption, Messenger, WhatsApp**

I. Introduction

In the age of interconnected digital communication, the feature of digital messaging can be commonly found across every social media application out there. This shows the integral role that digital messaging plays in human socialization, providing individuals with an internet connection the means to establish digital connections with people worldwide. Despite its convenience, concerns regarding users' privacy have surfaced over the years, prompting increased recognition of this issue within the public sphere. Consequently, the implementation of a security protocol, such as end-to-end encryption, is imperative and serves as an integral component of every digital messaging platform.

As one of the most prominent tech company, Meta, formerly known as Facebook, provides digital messaging services to 3.14 billion of its daily active users across all their products[1], such as Facebook, Instagram, WhatsApp. According to Facebook's 2023 global advertising audience reach statistics, Facebook Messenger maintains 931 million monthly active users, while Instagram's 2023 key statistics indicate that 375 million users engage in Instagram Direct Message (DM) each month. Furthermore, WhatsApp's 2023 user statistics point to the platform's substantial scale with one hundred billion messages sent daily. Given the magnitude of this company, there is an expectation for Meta to maintain a robust and advanced safety protocol, prioritizing the security and privacy of its user base.

One of the safety protocols implemented to safeguard user data privacy is end-to-end encryption (E2EE). This protocol ensures the security of communication from one endpoint to another by utilizing an encryption key to transform transmitted data into an unreadable, scrambled format and only authorized users possess the requisite decryption key are able to access and decipher the information. Notably, the key difference between end-to-end encryption and other encryption methods in transit lies in its comprehensive coverage. Unlike the latter, which secures data solely during transmission over the network, end-to-end encryption extends its protective scope until the data reaches the recipient's device. This approach restricts access to the transmitted data, preventing servers facilitating the transmission from deciphering its contents[2].

The implementation of end-to-end encryption serves as a crucial safeguard against data leaks and breaches, events that, on a global average, result in a substantial cost of 3.86 million USD[3]. This figure includes the expenses associated with addressing the violation and compensating for lost revenue. Beyond financial considerations, the repercussions may extend to the decline of consumer trust and potential regulatory fines or legal actions against the company. In doing so, implementing strong security measures, including end-to-end encryption, not only addresses financial risks but also enhances the protection of user privacy and ensures compliance with regulatory standards in the digital realm.

II. Cryptography

Cryptography, defined as the application of mathematical principles to encode a readable message into an unintelligible form, derives its etymology from the Greek words "*kryptos*," meaning hidden, and "*graphein*," meaning to write. Its historical application in communication dates back to around 1900 B.C., with evidence of secret hieroglyphics on stone tablets in ancient Egypt. This practice continued through history, including its use by the Spartan military in ancient Greece with tools like the "Scytale" of Sparta and the adoption of the well-known Caesar shift cipher in ancient Rome. The integration of mathematical concepts like permutations and combinations into cryptography emerged later, coinciding with advances in science and technology, as seen in Al-Khalil's "Book of Cryptographic Messages"[4].

In modern times, several cipher devices were developed until the early 20th century. The Enigma machine, invented by German engineer Arthur Scherbius after World War I, gained prominence as the most well-known among them. German soldiers in World War II utilized the Enigma machine to

exchange vital information among themselves. With the advent of computers, cryptographic methods began incorporating more mathematical concepts like information theory and computational complexity, organizing sequences of binary bits.

Within the field of cryptography, the unaltered and intelligible text is identified as plaintext, while the transformed and encoded counterpart is referred to as ciphertext[5]. The procedural conversion of plaintext to ciphertext is designated as encryption or enciphering, as shown in Figure 1 below. In contrast, the reverse of this process, involving the restoration of ciphertext to its original plaintext form, is termed decryption or deciphering.

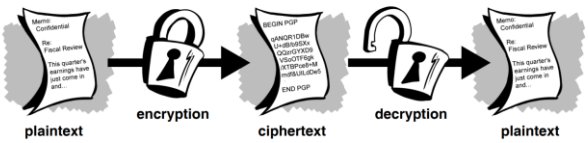

Figure 1. *Procedural sequence of encryption and decryption*

Source: https://www.cs.unibo.it/babaoglu/courses/security/resources/documents/intro-to-crypto.pdf

Encryption and decryption involve the application of a mathematical function known as a cryptographic algorithm, which operates in conjunction with a key. This key can take the form of a word, number, or phrase and is utilized for the encryption or decryption process. The combination of a cryptographic algorithm with a specific set of keys and protocols constitutes what is known as a cryptosystem[6]. The security of encrypted data relies on the strength of the cryptographic algorithms employed and the confidentiality of the key. The effectiveness of an algorithm is measure based on the time and resources required to uncover the original plaintext.

In general, cryptographic algorithms can be categorized in various ways, but a commonly employed classification is based on the number of keys used in encryption and decryption. One approach involves using a single key for both encryption and decryption, known as Secret Key Cryptography (SKC) or symmetric cryptography. Another method utilizes one key for encryption and a different key for decryption, referred to as Public Key Cryptography (PKC) or asymmetric cryptography. Additionally, there are algorithms that forego the use of keys entirely, relying instead on mathematical equations to achieve irreversible encryption; these are known as hash functions[7].

A. Secret Key Cryptography (SKC)

Secret-key cryptography utilizes the same key for both encryption and decryption, which is why it is also referred to as symmetric cryptography, as illustrated in Figure 2 below. Consequently, both the sender and the recipient must have access to the key, presenting a potential challenge as securely distributing the key can be a complex task. Several widely employed secret-key cryptographic algorithms in contemporary applications include the Data Encryption Standard (DES), acknowledged as one of the most widely used, the Advanced Encryption Standard (AES), CAST-128/256, and the International Data Encryption Algorithm (IDEA).

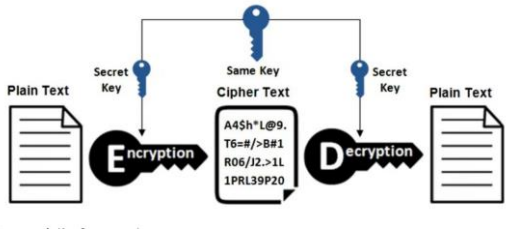

Figure 2. *The cryptosystem of secret key cryptography*

Source: https://www.shiksha.com/online-courses/articles/types-of-cryptography/

Secret-key cryptography is categorized into two types: stream ciphers and block ciphers. Stream ciphers operate on a single-bit basis, employing a feedback mechanism to generate a changing key. Among the various stream ciphers, two noteworthy types are self-synchronizing stream ciphers and synchronous stream ciphers. Self-synchronizing stream ciphers calculate each bit in the keystream based on the preceding n bits. As the name implies, they are self-synchronized, allowing the decryption process to synchronize with encryption by tracking its position in the n-bit keystream. However, a drawback is the potential for error propagation, where a distorted bit during transmission may impact the receiving end. In contrast, synchronous stream ciphers generate a keystream independently of the message stream, using the same keystream generation at both ends. While this approach avoids error propagation, the periodic nature of the keystream in synchronous stream ciphers means it will eventually repeat.

The alternative form of secret-key cryptography, known as block cipher, encrypts one block of data at a time, utilizing the same key for each block. The key distinction between block cipher and stream cipher is that, in the former, a block of text consistently encrypts to the same ciphertext, while the latter produces different ciphertext for each operation. Block ciphers operate in various modes, and four of the most pivotal modes include Electronic Codebook (ECB) mode, the most straightforward; Cipher Block Chaining (CBC) mode, implementing a feedback mechanism; Cipher Feedback (CFB) mode, incorporating the self-synchronizing stream cipher principle within block cipher methodology; and Output Feedback (OFB) mode, integrating synchronous stream cipher attributes within the framework of a block cipher.

B. Public Key Cryptography (PKC)

Public-key cryptography stands as a monumental leap in cryptographic advancements over the last three to four centuries. Introduced in 1976 by Professor Martin Hellman and graduate student Whitfield Diffie from Stanford University, it revolutionized secure communication by proposing a two-key crypto system. This innovative approach enables two parties to engage in secure communication over a non-secure channel without the necessity of sharing a secret-key, allowing for the public disclosure of the key, as depicted in Figure 3 below. The features inherent in public key cryptography include simplified initial deployment and maintenance, as key distribution is public and avoids the need for storing numerous secret keys. This system is particularly well-suited for open environments, marking a significant shift in cryptographic methodologies and

a substantial enhancement to the security landscape.

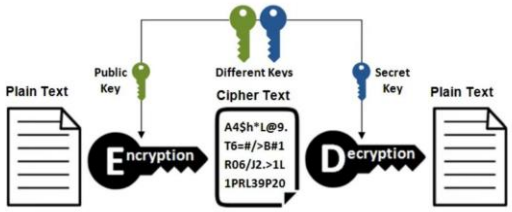

Figure 3. *The cryptosystem of public key cryptography*

Source: https://www.shiksha.com/online-courses/articles/types-of-cryptography/

Public-key cryptography relies on a "one-way" mathematical function, which is easy to compute but presents a challenging-to-compute inverse function without the necessary information. In a generic public-key cryptography system, two mathematically related keys are employed to ensure that possessing information about one key does not facilitate the determination of the other key. This design enables the widespread sharing of the designated public key, while safeguarding the secrecy of the designated secret key from unauthorized parties. With one key dedicated to encryption and another to decryption, the order of application is irrelevant as both keys are essential, leading to the term asymmetric cryptography. Among the public-key cryptography algorithms in current use, RSA takes the lead as the first and extensively utilized algorithm, alongside Diffie-Hellman, Digital Signature Algorithm (DSA), and Elliptic Curve Cryptography (ECC).

C. Hash Function

Hash functions, also referred to as message digests or one-way encryption, and formerly known as pseudo-random functions (PRF), are algorithms that operate without a key. They compute a fixed-length hash value from plaintext, rendering it impossible to determine the original plaintext length from a hash-created ciphertext. Hash algorithms are typically employed to ensure the integrity of a file, safeguarding it against unauthorized alterations by intruders or viruses. Despite a common misconception that two files cannot share the same hash value, such an occurrence is still possible. Therefore, a hash function must possess two essential properties: it must be one-way, as mentioned earlier, and it must exhibit resistance to collisions.

Despite the potential for collisions, discovering two files with the same hash value remains a challenging task. This emphasizes the prevalent use of hash functions in information security and computer forensic applications, often enhanced by specific extensions such as hash libraries, which consist of sets of hash values associated with known files, rolling hashes computed through a fixed-length sliding-window-like approach, and fuzzy hashes representing hash values indicative of similar inputs.

III. DISCUSSION

A. End-to-End Encryption (E2EE)

End-to-end encryption is a system where data is encrypted from one endpoint to another, ensuring that even unauthorized third parties, including those responsible for relaying transmitted data, cannot decipher the content[8]. A prevalent implementation in this domain is The Signal Protocol, recognized for its robust security and widely adopted by many companies. Notably, being an open-source solution, The Signal Protocol permits independent audits to identify and address potential security vulnerabilities, enhancing its credibility and reliability in safeguarding sensitive information.

The Signal Protocol employs a combination of symmetric and asymmetric encryption for both messages and calls. It utilizes the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol, specifically utilizing Curve25519, for the establishment of a shared secret key. This involves creating a private key for generating a public key, which is then shared with the other device. The shared secret key is employed to derive three session keys: one for message encryption using AES-256, one for message authentication, and one for call encryption using HMAC-SHA256.

Additionally, the Signal Protocol incorporates a ratchet mechanism to address situations where a device's keys are compromised, preventing the decryption of past messages by potential attackers. During the initial key exchange using ECDH, a master secret key is created, and from it, a Root Key and a Chain Key are derived for each device. When a message is sent, a Message Key is generated from the current Chain Key to encrypt the message. Subsequently, the used Chain Key is "ratcheted" forward, and a new one is generated, ensuring that the currently used Message Key cannot be used to access previous Chain Keys.

Furthermore, both Root Keys and Chain Keys are periodically regenerated by the Signal Protocol through new ECDH agreements, preventing a leaked Root Key from being utilized to access Chain Keys for current and future messages. All key pairs, except for the Identity Key, are temporary and frequently regenerated, generating a new master secret key. This approach aims to limit the potential compromise of data in case a key is leaked.

While Signal Protocol has numerous strengths and advantages, one notable challenge from a user's standpoint is the cross-device transfer of messages. Signal Protocol requires manual transfer of messages between devices, presenting a potential hurdle to a seamless user experience. However, ongoing efforts to enhance Signal Protocol include the introduction of "Sealed Sender," a feature designed to facilitate cross-device messaging and synchronizing.

This aspect becomes particularly intriguing when comparing Messenger’s implementation of end-to-end encryption to WhatsApp’s since it has had end-to-end encryption in place for an extended period, offering a longer-established foundation than the recently implemented default end-to-end encryption for Messenger, now used across Facebook and Instagram apps. Notably, in contrast to WhatsApp, Facebook and Instagram support cross-device messaging and syncing, posing a notable technical challenge due to the default end-to-end encryption, which limits server access to facilitate message transmission. A detailed analysis of the end-to-end encryption mechanisms employed by both WhatsApp and Messenger (utilized by

Facebook and Instagram) will be further discussed, focusing specifically on general and message encryption, to explore the similarities and differences between the two.

B. Similarities Between WhatsApp's and Messenger's End-to-End Encryption

As both WhatsApp and Messenger utilize the Signal Protocol, many encryption protocols employed in these applications are similar, although certain protocols exclusive to WhatsApp are absent in Messenger due to inherent limitations. The cryptographic operations utilized by both applications encompass ECDH, X3DH, AES, CBC, HMAC-SHA256, HKDF, and incorporate a double-ratchet system. Additionally, both applications feature identical public key and session key types. The public key types or pre-key utilized are enumerated as follows:

1. **Identity Key Pair (IK):** A long-term Curve25519 key pair generated during the registration process.
2. **Signed Pre-Key (SPK):** A medium-term Curve25519 key pair, also generated at registration time, signed by the Identity Key, and subject to periodic rotation.
3. **One-Time Pre-Keys (OPK):** A queue of Curve25519 key pairs for single use, generated at registration time, and replenished as required. A singular key is restricted to use in a single X3DH protocol operation.

Simultaneously, the session key types employed are as follows:

1. **Root Key:** A 32-byte value employed in the creation of Chain Keys.
2. **Chain Key:** A 32-byte value utilized in the creation of Message Keys.
3. **Message Key:** An 80-byte value employed in the encryption of message contents. This consists of 32 bytes for an AES-256 key, 32 bytes for an HMAC-SHA256 key, and 16 bytes for an IV.

The Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol, specifically Curve25519, stands out as the fastest option available. It enables two parties to establish a shared secret over an insecure channel by using one party's private key and the other's public key. This shared secret key is then employed for encrypting and authenticating messages exchanged between the two parties. Notably, the shared secret key created through elliptic curve cryptography possesses a considerably smaller size while maintaining a comparable level of strength when compared with a standard key. For instance, a key generated using elliptic curve with a length of 512 bits is roughly equivalent in strength to one created using standard asymmetric cryptography with a length of 15,360 bits. This efficiency facilitates a rapid and secure key agreement mechanism, as the smaller key size contributes to an expedited key exchange[9].

X3DH, or Extended Triple Diffie-Hellman Key Agreement, is an extension of the Diffie-Hellman protocols designed to address the limitations of asynchronous settings. An example of such a scenario is when the recipient of a message is offline while the sender remains online. The X3DH protocol comprises three significant phases, outlined as follows:

1. **Publishing keys:** A collection of public keys from the recipient is transmitted to the server prior to the start of communication. The initial key sent is the identity key, transmitted only once to the server. Subsequent keys include the signed pre-key and the pre-key signature, which are regularly updated at set intervals. When the server does not have a sufficient quantity of the recipient's one-time pre-keys, a new one is transmitted, accompanied by the deletion of the old key.
2. **Sending the initial message:** For the initiation of the X3DH key agreement, the sender is required to retrieve the recipient's identity key, signed pre-key, pre-key signature, and one of their one-time pre-keys if available; otherwise, the one-time pre-key retrieval is optional since sending a message is not mandatory. Initially, the sender verifies the pre-key signature, and if unsuccessful, the entire process is terminated. On successful verification, a pair of temporary keys is generated, which are then utilized to calculate the secret key. An associated data byte is derived, including both the sender's and recipient's identity keys. Subsequently, a message is sent to the recipient, containing the sender's identity key, temporary public key, an identifier specifying which of the recipient's pre-keys has been utilized, and an initial ciphertext.
3. **Receiving the initial message:** Upon receiving the initial message, the recipient employs their own identity private key, along with the sender's identity key, temporary public key, and the private counterparts of the signed pre-key and one-time pre-key. Only after this step can the deciphering process start. In the event of the deciphering process failing, the secret key is promptly deleted, and the process is terminated. Alternatively, if the decoding process succeeds, the private one-time pre-key is erased, while the secret key may be employed for post-X3DH processing.

The Advanced Encryption Standard (AES) is a widely employed symmetric encryption standard, representing a set of block ciphers distinguished by their key sizes. For instance, the commonly used AES-256, employed by Meta, features a 256-bit key. AES encryption involves permutating, substituting, and shifting data in a series of rounds determined by the key size; for instance, AES-256 requires 14 rounds for encryption. This encryption method is utilized to secure messages transmitted between parties. Both WhatsApp and Messenger specifically employ AES-256-CBC[10]. CBC, or Cipher Block Chaining, is a mode of operation well-suited for certain potential attacks but lacks message authentication. Therefore, both applications incorporate HMAC-SHA256 alongside it[11].

As mentioned earlier, HMAC-SHA256 serves the crucial role of authenticating and ensuring the integrity of a message. HMAC, an acronym for Hash-based Message Authentication Code, represents a cryptographic technique grounded in the principles of the SHA-256 hash function. In this process, the encrypted message and the secret key undergo hashing; the resulting hash value, combined with the secret key, undergoes a second hashing iteration, producing a final output of 256 bits. HKDF, or Hierarchical Key Derivation Function, builds upon HMAC-SHA256 to specifically generate the Root Key and

Chain Key. This function, true to its name, is designed to derive keys from a shared secret, enhancing the security architecture.

A double-ratchet system is created by integrating two distinct ratchet systems: the symmetric-key ratchet illustrated in Figure 4 and the Diffie-Hellman ratchet depicted in Figure 5[12].

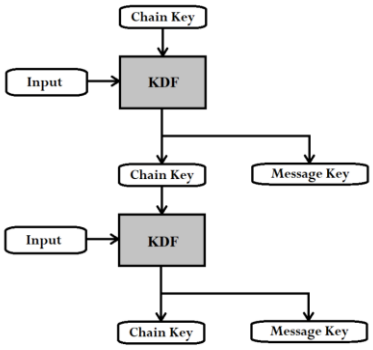

Figure 4. *Symmetric ratchet system*

Source: https://arxiv.org/pdf/2209.11198.pdf

The KDF function serves as the core component of the symmetric ratchet system. It computes an output by taking input data and a random KDF key or chain keys, as illustrated in Figure 4 above, which are utilized for sending and receiving chains. A segment of this output becomes the new KDF Chain Key, replacing the old one, while the remaining portion functions as a message key for encrypting transmitted messages using AES. However, a drawback of the symmetric ratchet system is its vulnerability to an attacker obtaining both the sending and receiving chain keys. In such a scenario, the attacker could calculate all future keys and decrypt upcoming messages. To enhance security, the symmetric ratchet system is employed in conjunction with the DH ratchet system, ensuring that the chain key output is influenced by the DH outputs, thus minimizing the risk of an attacker obtaining both the sending and receiving chain keys.

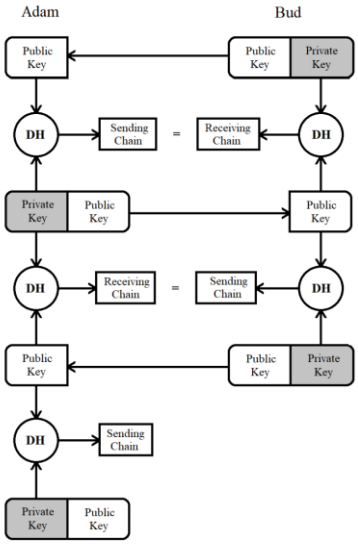

Figure 5. *Diffie-Hellman ratchet system*

Source: https://arxiv.org/pdf/2209.11198.pdf

The Diffie-Hellman (DH) ratchet operates by utilizing the user's private key and the other party's public key. This information is processed through a DH function, generating an output that establishes a sending/receiving chain matching the other party's. When a message is sent or received, the respective sending or receiving chain undergoes the symmetric ratchet system to derive the message key. Following this, the DH ratchet system is applied, incorporating the private and public keys, with the output serving as input for the symmetric ratchet system. This process results in a new message key for encrypting the message. The term "double ratchet" stems from the fact that the output of the symmetric ratchet system is used as the input for the DH ratchet system, and vice versa.

C. Differences Between WhatsApp's and Messenger's End-to-End Encryption

WhatsApp and Messenger, both utilizing the Signal protocol for end-to-end encryption (E2EE), exhibit variations in their implementation. While sharing a common encryption framework, their distinct features contribute to differences. WhatsApp follows a one-account-one-primary-device model, emphasizing security within a singular device, whereas Messenger accommodates cross-device messaging. Despite their shared protocol, these platform-specific features result in nuanced distinctions in their E2EE approaches.

The initial difference between WhatsApp and Messenger's end-to-end encryption lies in their respective registration processes. In WhatsApp, client registration begins with primary device registration, involving the storage of the identity key, signed pre-key, along with its signature, and a set of one-time pre-keys associated with the user's identifier on the server, as initiated by the WhatsApp client[13]. In contrast, Messenger follows a device registration approach, similar in content to WhatsApp but specifically associated with the user's device, facilitating offline session establishment between two devices, especially when one is offline[14]. Notably, WhatsApp introduces companion device registration after primary device registration, where the user's primary device creates an Account signature using the new device's identity key. The companion device responds by generating a Device Signature, signing the primary's public key identity key, allowing the establishment of end-to-end encrypted sessions on the companion device. WhatsApp's implementation introduces an additional key type, the Linking Secret Key, which is a 32-byte value generated on a companion device and securely transmitted to the primary device. This key verifies an HMAC of the linking payload received from the primary device, and the transmission is facilitated through scanning a QR code. Furthermore, WhatsApp incorporates companion linking data, including:

1. **Linking Metadata:** An encoded data blob containing metadata associated with a linked companion device. This information, combined with the companion device's Identity Key, serves to uniquely identify the linked companion across WhatsApp clients.
2. **Signed Device List Data:** An encoded list detailing the companion devices currently linked at the time of signing. This list is signed by the primary device's Identity Key, utilizing the 0x0602 prefix.
3. **Account Signature:** A Curve25519 signature computed

over the 0x0600 prefix, Linking Metadata, and the public Identity Key of the companion device. This computation is performed using the primary device's Identity Key.

4. **Device Signature:** A Curve25519 signature computed over the 0x0601 prefix, Linking Metadata, the public Identity Key of the companion device, and the public Identity Key of the primary device. This calculation is executed using the companion device's Identity Key.

WhatsApp features a strong end-to-end encryption framework, offering additional security measures not found in Messenger. One noteworthy inclusion is the encryption of message add-ons within community announcement groups. In these groups, regular members are restricted from sending messages and can solely interact with the content shared by group administrators. WhatsApp employs add-on sender keys, distinct from conventional group sender keys, to encrypt these add-ons. When an administrator sends a message in a community announcement group, the message is encrypted similarly to a standard group message. However, the end-to-end encrypted message payload includes an additional element—a randomly generated key known as the message secret.

For the end-to-end encryption of add-ons in WhatsApp Community Announcement Groups, established pairwise encrypted sessions play a crucial role in distributing a dedicated add-on sender key component, following the Signal Messaging Protocol. Upon sending an add-on to a community announcement group for the first time, an add-on sender key is generated and distributed to each member's device within the group, utilizing the pairwise encrypted sessions. The add-on content undergoes encryption using a key derived from the target message's "Message Secret" and is then encrypted once more using the add-on sender key. This ensures an efficient and secure fan-out for the add-ons sent to Community Announcement Groups.

While it might appear that Messenger falls short in comparison to WhatsApp, Meta has made significant strides in addressing the primary challenge of the Signal Protocol, particularly in the realm of cross-device messaging. The Signal Protocol traditionally stores message data locally, requiring manual transfer via Bluetooth. This limitation is a key factor in restricting a WhatsApp account to a single primary device. In response, Meta's engineers have introduced their proprietary encrypted storage protocol named Labyrinth. This innovative protocol allows users' data to be securely stored on the company's servers, ensuring that the data remains encrypted and inaccessible to the company itself[15].

Labyrinth not only securely stores encrypted user data but also manages post-revocation messages, accommodating both the addition and removal of devices from a user's account. This is crucial to ensure that devices removed from the account have no access to new messages. The system also adeptly handles sent attachments, storing them separately from mailboxes. The cryptographic primitives employed by Labyrinth include AES-GCM-Extended, Labyrinth HPKE, XEdDSA[2], HMAC-SHA256, HKDF-SHA256, and a distinctive construction termed The Oblivious Revocable Function (ORF)[16]. The ORF is built around the Ristretto 255 group and is designed to reduce linkability between attachments and their respective mailboxes. The ORF comprises two pseudo-random function (PRF) chains—one running on the client side and the other on the server side. Each entity possesses its secret scalar key, allowing these keys to be regenerated to new ones in a manner that maintains consistency between the output of the two PRFs for a given input. This distinctive design ensures that the client side is unaware of the overall output, and similarly, the server side remains unaware of the overall input. As a result, this architecture facilitates access to the system by different devices or clients with unique keys, allowing the server to revoke access from a client while preserving the integrity of inputs.

The backend of Labyrinth comprises two essential components: one containing operational protocol data, known as the mailbox metastore, and another housing message ciphertexts in a structured database, referred to as the mailbox. The delineation between these components, along with much of the protocol's intricacy, stems from the objective of treating revoked devices as potential threats. Consequently, the protocol is designed to facilitate key rotation while accommodating devices that may remain offline for extended durations.

Each instance of Labyrinth is characterized by global values, including a unique identifier labelled labyrinthID, assigned by the server and associated with the mailbox metastore, and mailboxRootSalt, a non-secret random value unique to the Labyrinth instance. In the Labyrinth context, any entity with access to a Labyrinth mailbox is termed a device. While commonly physical devices like smartphones, they may also be termed "virtual devices" – collections of cryptographic keys treated as devices by the protocol, not tied to a physical device. Each Labyrinth device possesses specific keys, and within the mailbox metastore, the server maintains, per-device, various pieces of information. To facilitate key rotation, Labyrinth incorporates the concept of an "epoch," representing a timeframe during which no device is revoked, although devices can be added. Each epoch is linked to specific values to support the seamless functioning of the protocol.

IV. Conclusion

In conclusion, end-to-end encryption (E2EE) is a critical security measure ensuring data remains encrypted from one endpoint to another, preventing unauthorized access. The Signal Protocol, widely adopted for its robust security, employs a combination of symmetric and asymmetric encryption, featuring Elliptic Curve Diffie-Hellman (ECDH) for key agreement. The protocol includes a ratchet mechanism to counter key compromise and incorporates periodic regeneration of Root and Chain Keys for enhanced security. While Signal Protocol exhibits strengths, the challenge of cross-device messaging persists, addressed by ongoing efforts such as the "Sealed Sender" feature.

Both WhatsApp and Messenger, utilizing the Signal Protocol, share encryption protocols, including ECDH, X3DH, AES, CBC, HMAC-SHA256, and HKDF, featuring a double-ratchet system. The public and session key types used align between the two applications. ECDH, particularly Curve25519, stands out for its efficiency in key exchange. X3DH addresses asynchronous settings, ensuring secure communication initiation even when the recipient is offline.

Differences arise in their implementation, with WhatsApp emphasizing one-account-one-primary-device security, while Messenger allows cross-device messaging. WhatsApp's registration involves primary device and companion device steps, introducing additional key types like the Linking Secret Key. WhatsApp further encrypts message add-ons in community announcement groups, enhancing security.

Messenger, with a more recent implementation of default end-to-end encryption, accommodates cross-device messaging. Meta's Labyrinth addresses cross-device limitations in Signal Protocol, introducing encrypted storage and handling post-revocation messages. Labyrinth uses innovative cryptographic primitives, including The Oblivious Revocable Function, ensuring secure access from different devices.

In summary, while both WhatsApp and Messenger share the foundational Signal Protocol, their nuanced implementations reflect platform-specific features and security priorities. Ongoing developments continue to refine end-to-end encryption mechanisms, emphasizing the importance of robust encryption in safeguarding user data.

V. ACKNOWLEDGMENT

The author expresses heartfelt appreciation to their parents, Dr. Fariska Zakhralativa Ruskanda, S.T., M.T., the discrete mathematics lecturer, and all friends who have consistently supported them throughout this semester.

REFERENCES

[1] "Meta global family DAU 2023," Statista. Accessed: Dec. 09, 2023. [Online]. https://www.statista.com/statistics/1092227/facebook-product-dau/#:~:text=During%20the%20third%20quarter%20of

[2] IBM, "What is end-to-end encryption? | IBM," www.ibm.com, 2023. Accessed: Dec. 09, 2023. [Online]. https://www.ibm.com/topics/end-to-end-encryption.

[3] IBM, "Cost of a data breach 2023," IBM, 2023. Accessed: Dec. 09, 2023. [Online]. https://www.ibm.com/reports/data-breach

[4] S. Naser, "CRYPTOGRAPHY: FROM THE ANCIENT HISTORY TO NOW, IT'S APPLICATIONS AND A NEW COMPLETE NUMERICAL MODEL," International Journal of Mathematics and Statistics Studies, vol. 9, no. 3, pp. 11–30, 2021. Accessed: Dec. 10, 2023. [Online]. Available: https://www.eajournals.org/wp-content/uploads/Cryptography.pdf.

[5] K. Rosen, "Series Editor DISCRETE MATHEMATICS AND ITS APPLICATIONS An INTRODUCTION to CRYPTOGRAPHY Second Edition." Accessed: Dec. 10, 2023. [Online]. Available: https://mrajacse.files.wordpress.com/2012/01/an-introduction-to-cryptography.pdf

[6] N. Papanikolaou, "An introduction to cryptography," PGP Corporation, version 8.0, October 2002. Accessed: Dec. 11, 2023. [Online]. Available: https://www.cs.unibo.it/babaoglu/courses/security/resources/documents/intro-to-crypto.pdf

[7] G. Kessler, "An Overview of Cryptography An Overview of Cryptography," 1998. Accessed: Dec. 11, 2023. [Online]. Available: https://www.cs.princeton.edu/~chazelle/courses/BIB/overview-crypto.pdf

[8] A. Greenberg, "Hacker Lexicon: What Is End-to-End Encryption?," Wired, Nov. 25, 2014. Accessed: Dec. 11, 2023. [Online]. https://www.wired.com/2014/11/hacker-lexicon-end-to-end-encryption/

[9] M. Niasar, R. El Khatib, R. Azarderakhsh, and M. Mozaffari-Kermani, "Fast, Small, and Area-Time Efficient Architectures for Key-Exchange on Curve25519," in 2020 IEEE 27th Symposium on Computer Arithmetic (ARITH), 2020, pp. 72-79.

[10] V. Bhuse, "Review of End-to-End Encryption for Social Media," International Conference on Cyber Warfare and Security, vol. 18, no. 1, pp. 35–37, Feb. 2023. Accessed: Dec. 11, 2023. [Online]. doi: https://doi.org/10.34190/iccws.18.1.1017.

[11] N. Kishore and B. Kapoor, "Attacks on and Advances in Secure Hash Algorithms," IAENG International Journal of Computer Science, vol. 43, no. 3, pp. 25-34, 2016.

[12] P. Prasad and G. Neogi, "A Dive into WhatsApp's End-to-End Encryption," Sep. 2022. Accessed: Dec. 11, 2023. [Online]. Available: https://arxiv.org/pdf/2209.11198.pdf

[13] WhatsApp, "WhatsApp Encryption Overview: Technical White Paper," Sep. 2023. Accessed: Dec. 11, 2023. [Online]. Available: https://scontent.fbdo2-1.fna.fbcdn.net/v/t39.8562-6/384251896_820338303082371_8514785982310046047_n.pdf?_nc_cat=100&ccb=1-7&_nc_sid=e280be&_nc_ohc=SbnQ_wvTBFUAX_O_IE4&_nc_oc=AQnKpAFl9OxxHMBMt5KksMLVpvUq1SYJuXq9R2Y7WrQJrirzoXZbVf22Fw7kyq_2si8&_nc_ht=scontent.fbdo2-1.fna&oh=00_AfB8u3iyb-lHXDWquVZB6wSjJkosGhV9sfMwZm3uN5HlFw&oe=657B2111

[14] Meta, "Messenger End-to-End Encryption Overview," Dec. 2023. Accessed: Dec. 11, 2023. [Online]. Available: https://engineering.fb.com/wp-content/uploads/2023/12/MessengerEnd-to-EndEncryptionOverview_12-6-2023.pdf

[15] L. H. Newman, "Why It Took Meta 7 Years to Turn on End-to-End Encryption for All Chats," Wired, Dec. 07, 2023. Accessed: Dec. 11, 2023. [Online]. https://www.wired.com/story/meta-messenger-instagram-end-to-end-encryption/ (accessed Dec. 11, 2023).

[16] Meta, "The Labyrinth Encrypted Message Storage Protocol," Dec. 2023. Accessed: Dec. 12, 2023. [Online]. Available: https://engineering.fb.com/wp-content/uploads/2023/12/TheLabyrinthEncryptedMessageStorageProtocol_12-6-2023.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2023

Salsabiila, 13522062