Curious Administrator's Misbehaviors in Message Layer Security (MLS)

Hyoung-Kee Choi
College of Computing and Informatics
Sungkyunkwan University
Suwon, Korea
0000-0002-5342-5913

Songhui Kwon
Department of Electrical and Computer Engineering
Sungkyunkwan University
Suwon, Korea
0000-0002-2875-7502

***Abstract*—A key factor of the MLS group protocol is that it provides asynchronously efficient group key establishment with forward secrecy and post-compromise security for many groups in a standardized way. End-to-end encryption (E2EE) is essential to fulfilling the correctness of security in a messenger application. The correctness heavily depends on the trustworthiness of a middleman server in the center where a curious administrator may break E2EE. If this happened the administrator as a nonmember could read messages and impersonate a member by intervening with group-key management. This paper presents an in-depth explanation of the MLS protocol and demonstrates a proof-of-concept of curious administrator's misbehaviors by analyzing the protocol. Our research outputs suggest that unconditionally trustworthy relationships on the central server in designing network protocols for security needs to prepare for any incidents caused by misbehaving administrator of the central server.**

Keywords— Secure Messaging Service, Message Layer Security, End-to-End Encryption

I. INTRODUCTION

Messenger applications are gaining popularity as it is the preferred method of communication in a multitude of different situations from collaborating with a colleague to dropping a line to an acquaintance [1]. With new ways to communicate and connect via technology, security has been lagging during the increased modernization of these applications. Personal messages could potentially be read by third parties, the organization behind applications, and governments who collect private information on their citizens. Users may want extra layers of protection to help prevent any outside snooping from third parties.

Privacy, security, and proper implementation are primary concerns and requirements in messenger applications to ensure that only intended recipients can read messages. End-to-end encryption (E2EE) proposes as one of the solutions for the concern. E2EE uses cryptographic keys that the sender and recipient only own. Thus, only the entity can access the message if they have the key. When the coverage of E2EE for a pairwise connection extends to support group conversations, its security gets even more complex [2]. Users of the same group maintain a secret key used for encrypting and decrypting messages. The key needs to share among group members only in a secure manner. The group key needs to be updated every time the group membership change. This issue has implications for security and performance making it challenging to operate group conversations of over a few hundred members.

Most messenger applications are homogeneous and centralized such that users of each application can only communicate with one another within application boundaries. A user cannot choose the most trustworthy provider but needs to fully trust the chosen provider that develops both protocol and application. In online communications, an intermediary is almost always relaying messages between two parties involved in an exchange. That intermediary is usually a server belonging to an ISP, a telecommunications company, or other organizations. Messages must go to the server first and then be delivered to the designated recipient. Often servers cannot be trusted, as they can hacking by an adversary or may contact by law enforcement to give them the information sent by clients to the server. If a messenger application relies on a central server to function, an administrator who controls that server effectively controls the network.

A well-designed, secure E2EE ensures that intermediaries cannot eavesdrop on messages. To address such issues in group message security, the Internet Engineering Task Force (IETF) Working Group develops a group messaging protocol, Message Layer Security (MLS) [4]. MLS focuses on improving the scalability of E2EE to support thousands of users, covering multiple industry use cases, including federation and Web browser support, and offering formal security guarantees. MLS strives to make the protocol a standard that everyone can use freely and securely and hopes to pave the way to a more secure infrastructure by accommodating an open standard for all industries. It is currently a work in progress and is not a complete implementation but an architectural and protocol specification.

Despite incremental progress in MLS, the current standardization work focused on handling the group key efficiently and improving scalability. However, we were unaware of capable administrators' operational correctness in intermediary servers. This paper aims to take significant steps toward an MLS security mechanism by revealing weaknesses that misbehaving server administrators can abuse. We have examined the mechanics of group-key management in MLS and found that an administrator with curiosity may be able to read and write messages in other's private conversations. Our findings suggest that deploying MLS in centralized infrastructure across individuals, organizations, and governments is flawed and should be reconsidered.

II. SECURITY REQUIREMENTS

We describe the essential security requirements for a group key management protocol. First, messages should be written and read only by the group members. Second, message's integrity should ensure in any cases. Third, message replaying should not be allowed. In addition to these three security requirements common to any network protocol, five additional requirements are specific to a messenger application.

Confidentiality to a middleman server: All messages in a group must be sent to a server in the first place to deliver them to receivers. The server should not be able to read and

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

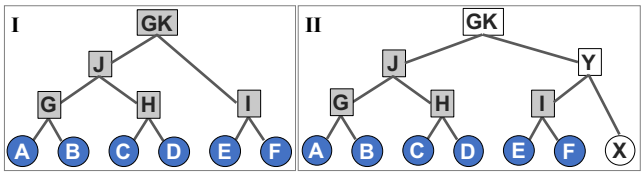

Fig. 1. (I) A structure of the group-key tree. (II) Illustration of group-key update when new member **X** is invited into the group.

write those messages. Senders must securely encrypt a message with a group key to prevent non-group members, including the server, from snooping.

Sender's authentication: A non-member who can intercept a message can insert, forge, replay and drop a message. Sender authentication can prevent non-members from taking illegal actions on messages. A group key is not entirely helpful in this case because this key does not belong to a specific member but to a group. Sender authentication is particularly useful where a group leader can invite and evict a group member. The leader sends a message to members to inform them of member departures and exits. The rest of the members can confirm the message by verifying sender's authentication.

Forward secrecy: A new member cannot derive past group keys from the current group key. It is to protect messages in the past sessions when the current session is compromised, session being defined to an epoch between group-key updates. A group key is updated immediately after a member joins a group.

Post-compromise security: In contrast to forwarding secrecy, the message is protected in future sessions when the current session is compromised. This requirement secures future sessions if the current sessions are compromised [5]. It is necessary for leaving members to be unable to read or write messages to group members. A group key is updated as soon as a member leaves the group.

Non-repudiation of delivery and receipt: A sender and a receiver may repudiate the message's sending and receiving. Nonrepudiation provides proof of the origin, authenticity, and integrity of data. It assures the sender that the message was delivered and gives proof of the sender's identity to the recipient.

III. System Model of MLS

MLS is a group communication protocol for messenger services. Designing the protocol considers both practical implementation and specification issues. There are six main operations in MLS.

A. Service subscription

A user subscribes to a service for the first time. A new user must have a unique identifier, such as phone number and an email address. Then they generate an asymmetric key pair for authentication associated with the identity. A public key is sent to a server, while a private key is kept secretly on the local device. This asymmetric key pair using for the authentication of messages and senders.

B. Group creation

A user creates a group by inviting members to the group. MLS adopts a tree-based group key management. Group members share the group's status represented in the annotated tree. Group-key management is responsible for tree synchronization among members. A group member locates at a leaf node in a left-balanced binary tree. All middle nodes are virtual because they are not directly associated with members. As illustrated in Fig. 1, six leaf nodes in the circular shape represent members **A** to **F** while other five middle nodes **G**, **H**, **I**, **J**, and **GK** in the rectangular shape are virtual.

TABLE I. Two pairs of asymmetric keys used in MLS

	Asymmetric key pair for distribution	**Asymmetric key pair for authentication**
	{ $\mathbf{PU_{dist}A}$, $\mathbf{PR_{dist}A}$ } for node **A**	{ $\mathbf{PU_{dist}B}$, $\mathbf{PR_{dist}B}$ } for node **B**
Role	Secure distribution for group key	Source authentication
Usage	Group-key update	Member's Invitation and departure, group-key update
Owner	Middle nodes, leaf nodes	Leaf nodes
Update	When a group key is updated	Rarely
Public key	It is shared among group members.	It is stored in a middleman server.
Private key	Leaf knows this key of nodes in ancestral path to a root.	It is stored in its own device.

A member generates an asymmetric key pair for group-key distribution dedicated to a group. A member who joins multiple groups has a unique key pair per group. In consequence, the member has an asymmetric key pair for authentication and multiple pairs of asymmetric keys for distribution. We denote $\mathbf{PR_{dist}X}$ and $\mathbf{PU_{auth}X}$ to the node **X**'s private key for distribution and the node **X**'s public key for authentication, respectively. Refer TABLE I for a more detailed description of each key pair.

The value of the leaf node sets to the public key of the corresponding member for distribution. A middle node' asymmetric key pair for distribution is computed based on its child nodes. A leaf node knows all the private keys for distribution of nodes on the shortest ancestral path to a root node. For example, node **A** in Fig. 1-(I) has private keys of nodes **G**, **J** and **GK**. Further, all public keys of al nodes for distribution are shared publicly among all group members. The group key selects, $\mathbf{PR_{dist}GK}$, the private key of the root node, for distribution.

C. Member invitation

Fig. 1-(I) shows the six members **A** to **F** in the group-key tree. Fig. 2 illustrates an exchange of five messages among group members chronologically. The first column in Fig. 2 is the message's number from one to five. The number in the second column corresponds to the number of the group-key tree in Fig. 1. The third column is a part of the payload related to group-key management.

Member **A** invites member **X** by sending the first message in Fig. 2 to the server to request the new member's public key for authentication. The server returns the response in the second message. An inviter introduces an invitee in a group by sending the invitee's public key for authentication and identification in the third message. Note that messages must be sent to central server and delivered to receiving members. In the fourth message, an inviter deletes the group key from the tree and shares the group-key tree with the invitee, including all nodes' public keys for distribution and all members' public keys for authentication.

Currently, a leaf node for a new member is created in a left-balance tree, as shown in Fig. 1-(II). The first four messages belong to the member invitation. Group members are ready to update the group key by rebuilding the tree. Updating steps in detail is described elsewhere.

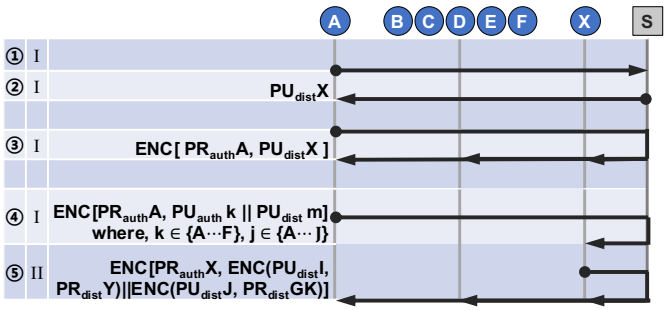

Fig. 2. A procedure for updating a group key when member **A** is inviting new member **X**.

D. Group key update

A group key is updated at an event when either the key is expired, or group membership changes. An update is led by a member who sends the first message after such an event takes place. In an illustration, we will explain an updating step where new member **X**. Fig. 1-(II) leads the update and shows the tree after new member **X** joins the group. Member **X** is responsible for completing a tree by filling up empty nodes **Y** and **GK** with keys. A parent node's private key is computed by hashing either of the child's private key. For instance, the **Y**'s private key computes from the **X**'s private key. A public key derives from its corresponding private key.

Using the fifth message in Fig. 2, member **X** distributes the **Y**'s private key and **GK**'s private key. Members **A**, **B**, **C**, and **D** do not need the **Y**'s private key for group-key management. Further, these nodes share a common ancestor node **J** and its private key. Member **X** encrypts the **GK**'s private key for operational efficiency with the node **J**'s public key. In the continuing sense, member **X** encrypts information with the node **I**'s public key for members **E** and **F**. If member **X** were to send information to each member by encrypting information with a recipient's public key, it would cost the complexity of **O(N)** for distribution. The operational efficiency improves the complexity from **O(N)** to **O(log2N)** by sharing ancestor nodes' private keys for distribution.

Let **ENC[PU$_{dist}$J, PR$_{dist}$X]** denote encryption of **PR$_{dist}$X** with a secret key of **PU$_{dist}$J.** The fifth message in Fig. 2 sent to members **E** and **F** includes **ENC[PU$_{dist}$I, PR$_{dist}$Y]**. Members **E** and **F** are node **I**'s children and can decrypt the fifth message using the node **I**'s private key. They derive the root node's private key from the **Y**'s private key

E. Member departure

Member **A** in Fig. 1-(II) is about to leave the group. News about the member's departure is distributed immediately among the rest of the members. They update the group key by removing the node **A** and its ancestral nodes **G**, **J**, and **GK**. The group-key update is led by the first member sending a message after the departure in the same manner as described in the member invitation..

F. Message transmission

A member sends a message to each group member after encrypting and then signing the message. A group member has a distinct secret key called a sender key. It derives from the group key and the sending member's ID. Furthermore, the sender key is different for messages to keep forward secrecy. The following sender key updating by ratcheting the current sender key. Here what we mean by ratcheting is applying a one-way hash function iteratively. The sender key never uses again, even if a message retransmits. Consequently, the sender key is unique per message and member. Replayed and out-of-sequence messages would not be accepted by receiving members due to the desynchronization of sender keys.

A sending member sends a message to the middleman server over a connection secured by TLS. The server sends a message to the group members over separate connections secured by TLS. The sender signs a message with its private key for authentication. The receiving member authenticates a message and authorizes a user by verifying the signature.

IV. Security Analysis

A. Confidentiality, Message Integrity, and Prevention to Message Replay

Messages are encrypted in two layers: the sender key in the first layer and the TLS session key in the second. It is almost impossible for non-members to obtain secret keys in two layers simultaneously. Further, a sender encrypts messages with a private key for authentication. Signature verification at recipients ensures message integrity even if a message replays. Replaying a message by a non-member through impersonation will not work because the sender key is unique per message.

B. Confidentiality to a middleman server

It is based on firm trustworthiness that the server is ignorant of the group key entirely in any case. However, we found that in some cases, this condition is only sometimes actual. An administrator with curiosity may manipulate updating messages for a group key to enforce the group key to set their choice. Furthermore, the curious administrator could access the server's private key used for the TLS. In this way, the curious administrator can bypass the two layers of security for message confidentiality.

C. Sender authentication and non-repudiation

A sender signs messages with their private key for authentication. By verifying a signature on the message, receiving members can authenticate the sender's group membership and authorization in the group. Besides, a sender cannot repudiate later when they have sent the message as far as the receiver has a valid signature. In contrast, the receiver's non-repudiation of message receipt is not supported in MLS as it levies many housekeeping overheads due to the one-to-many communication nature [7]. A sender requests a receipt of delivery for messages, and a receiver returns the receipt. If there are **N** members in a group each message will follow **N** responses which may bring about overheads in the network and the server.

D. Forward secrecy

It is guaranteed if a new member cannot distinguish past group keys from the current ones. Group-key management in MLS satisfies forward secrecy by updating the group key whenever a member joins a group. At the same time, middle nodes on the ancestral path from the new member's node to the root node update their asymmetric key pairs for distribution because these keys were used for to distribute the last group key.

The attacker might have recorded messages used to distribute the recent group key before they joined the target group. Once the attacker joins the group, they begin to know their ancestors' private keys for distribution. One of these keys is used to decrypt the recorded messages for group key distribution and, finally, to obtain the recent group key. A

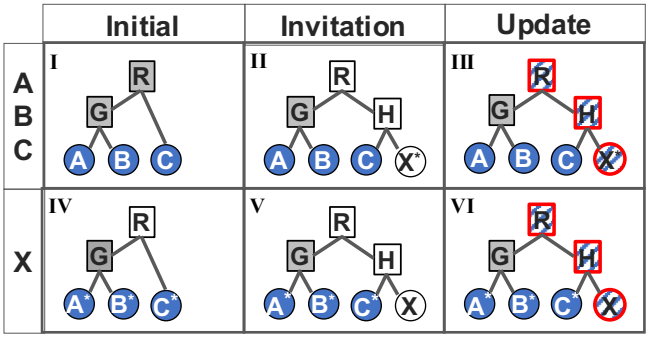

Fig. 3. Transition of the group-key trees in the three states.

unique sender key for a message enables MLS to maintain forward secrecy because updated sender keys cannot be derived from the current sender key even if the current sender key is compromised.

E. Post-compromise security (PCS)

It achieves if a previous compromise of the group key did not affect security of the incumbent group key. It means that a leak of keys to a passive eavesdropper would be healed. An attacker, once in possession of the complete status of a user, can impersonate that user trivially, thus achieving no security. However, suppose the attacker becomes passive at some point. Reestablishing secrets between group members, excluding the attacker should be possible because a group key is updated whenever there are changes in group membership. If the victim member remains in the group, the attacker can access group messages by impersonating the user. In addition, asymmetric key pairs of leaving member's ancestor nodes are deleted for the same reasons as forward secrecy.

V. ATTACK IN THE MIDDLEMAN SERVER

The security of group key management in MLS builds on the assumption that the middleman server is behaving trustworthy. However, we found situations where trust must break because an administrator abuses their authority on the server. Without loss of generality, we can safely assume that such an administrator can access most of the information saved on the server, such as the private key for TLS, members' public keys for authentication, etc.

When a new member joins a group, the existing member and the new member must exchange each other's public key for authentication. These messages are relayed to each group member by the middleman server, allowing curious administrators to change the public keys to their choice. A curious administrator can launch a man-in-the-middle attack to learn the new group key. This attack is possible because the member responsible for distributing the public key for authentication must authenticate in key management. Fig. 3 and Fig. 4 demonstrate the curious administrator's attack. Fig. 3 illustrates a transition of the group-key trees in the three states where a new member **X** joins a group. The first row in Fig. 3 is for an existing member's tree, while the second is for a joining member's tree. There are two different trees in the same group because the curious administrator manipulated the public key of authentication.

A. Attack preparation

Fig. 3-(I) shows the tree for the group of three members, **A**, **B**, and **C**. The curious administrator's attack initiates when member **A** invites new member **X** to the target group. Fig. 4

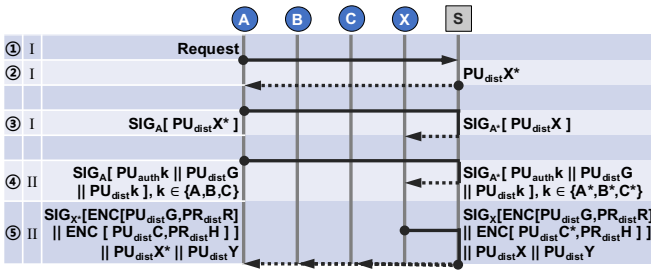

Fig. 4. A procedure of the curious administrator's attack in the group key update.

shows the first message that the inviter requests the server the member **X**'s public key for authentication. Member **A** share group members $\mathbf{PU_{auth}X^*}$ in the third message. This message signs digitally with $\mathbf{PR_{auth}A}$; that is $\mathbf{SIG[\ PUa_{uth}X^*\ ]}$. The curious administrator is careful not to send the third message to member **X**. Otherwise, member **X** will report a discrepancy in its public key, and as a result, the attack will fail. The curious administrator simply forwards the message to existing members, **B** and **C**. To new member **X**, the curious administrator sends **X**'s real public key signed by **A**'s fake private key.

In the fourth message of Fig. 4, the leading member sends the new member a welcome package of information. It includes 1) a structure of the tree after deleting the root node as shown in Fig. 3-(IV), 2) leaf nodes **A**, **B**, **C**'s public keys for authentication and distribution, 3) middle node **G**'s public key for distribution ($\mathbf{PU_{dist}G}$) and etc. The leaf nodes' public keys for authentication and distribution are replaced by fake ones chosen by the curious administrator. The fourth message signs by member **A**'s fake private key for authentication. Without knowing it is fabricated, member **X** accepts the fourth message because it authenticates. As a result, member **X** would have the group key tree as shown in Fig. 3-(V) while member **A** has the one as shown in Fig. 3-(II).

B. Attack in action

Let us assume that member **X** is a leading member for the next key schedule. Ancestor nodes **H**'s and **R**'s private keys for distribution are sequentially computed from its child node's private key as shown Fig. 3-(VI). The leading member must share the new group key with members. Firstly, the new group key is encrypted with node **G**'s public key for distribution, $\mathbf{ENC\ [\ PU_{dist}G,\ PR_{dist}R\ ]}$, for members **A** and **B**.

Secondly the node **H**'s private key is encrypted with the node **C**'s fake public key for distribution, $\mathbf{ENC\ [\ PU_{dist}C^*,\ PR_{dist}H\ ]}$. These two blobs are signed by member **X** and delivered in the fifth message of Fig. 4. The curious administrator can decrypt the second blob in the fifth message with the fabricated private keys of node **C** ($\mathbf{PR_{dist}C^*}$) to compute $\mathbf{PR_{dist}H}$. In consequence, the administrator can compute the group key from node **H**'s private key for distribution.

The second blob needs further processing by the attacker before distributing it among members. Node **H**'s private key encrypts with the node **C**'s genuine public key for distribution, $\mathbf{ENC\ [\ PU_{dist}C,\ PR_{dist}H\ ]}$. This newly computed encryption replaces the second blob in the fifth message. Existing members are unaware of the new member's public key for distribution. An additional field in the fifth message is the new member's public key for distribution. This public key is necessary for other members to share future group keys with this new member.

The fifth message node **X** sends comprises is four fields: **ENC [$PU_{dist}G$, $PR_{dist}R$] || ENC [$PU_{dist}C^*$, $PR_{dist}Y$] || $PU_{dist}X$ || $PU_{dist}Y$**. The attacker modifies this message to **ENC [$PU_{dist}G$, $PR_{dist}R$] || ENC [$PU_{dist}C$, $PR_{dist}Y$] || $PU_{dist}X^*$ || $PU_{dist}Y$**. As a result of the intervention, the existing member and the new member have group-key trees as shown in Fig. 3-(III) and Fig. 3-(VI), respectively. The middle node's public keys are identical in the two trees, while the leaf nodes' public keys are not.

VI. Fixing the Protocol

The curious administrator's attack is mainly caused by the lack of source authentication in critical messages and unconditional trustworthiness of the middleman server.

Out-of-Band Authentication (OOB) requires that the secondary channel for authentication be separate from the primary channel used to perform a transaction. Using a separate channel makes it significantly difficult for an attacker to intercept and subvert the authentication process. Examples of OOB authentication include codes or links sent to a mobile device through SMS and an email. OOB authentication can be incorporated into the MLS to use a secure channel for exchanging member's public keys. The public key exchange needs to happen once per member when the member joins a group. The cost of the exchange would be proportional to the number of group members, **O(N)**. As surprising as its cost may sound, an entire operation of OOB authentication can be automated.

Identity-based Encryption (IBE) is a type of public-key encryption where a public key is a form of unique information about a member's identity, such as a phone number, email address, and logic ID. A member can generate a public key from an officially unique identity. A trusted third party (TTP) generates corresponding private keys from the public key for the member. As the public key derives from an official member identifier, IBE eliminates the need for certificates and certificate revocation lists in PKI [9]. Ferris Research looked at the costs of one commercial IBE system and found that the total cost is one-third that of a typical public-key system.

In contrast, it requires a centralized server acting as the TTP. IBE's centralized approach implies that secret keys must be created and held in escrow, leaving the secret at significant risk of disclosure. If the TTP is compromised, all messages protected over the entire lifetime of the public-private key pair are also compromised. This makes the TTP a high-value target to adversaries. The TTP needs to be trusted. Because the TTP generates private keys for members, it may decrypt any message without authorization. It should note that many IBE systems already employ key escrow in a centralized server to recover encrypted data if the member's private key is lost. This implies that IBE systems cannot use for security requirements of non-repudiation. Once a setup for a key pair with the server is complete, the following messages do not need to pass the centralized server to reach receiving members. Two members can establish a secure connection directly, preventing the curious administrator from intercepting messages. In case receiving members are asynchronously offline, a sending member can check the status of that member with a server and wait to make connections until they become active again.

TTP-free Solutions, Decentralization: A centralized TTP would add overheads to the protocol and become a bottleneck for communications. Furthermore, there is a practical need for messenger applications operated in a more decentralized setting. Typically suggested solutions are the blockchain and the web of trust. They facilitate trusting relationships between untrusted entities in a decentralized environment [8]. A decentralized messenger Session is available in the market and ready for download. However, these are not best-fit solutions when we evaluate their performances with respect to the latency of writing to the blockchain and the cost of signing on the public keys. Developing fully decentralized services will be a challenging and well-motivated direction for future research.

VII. Conclusion

We have discussed and verified that MLS has satisfied many security requirements for a messenger application, significantly forwarding secrecy and post-compromised security. These two requirements make it possible to share messages with group members. New and old members cannot read and write messages before and after their membership expires. We also discovered that a curious administrator, if authorized to access specific resources on the server, could illegally read messages as a non-member. This is due to the protocol's insecure design when they exchange messages to build a group-key tree. The receiving member does not authenticate the sending member, and all messages should be relayed by the middleman server and then delivered to each member. As a result, a man in the middle can manipulate messages to reveal new group keys and further read group conversation. This ancient and notorious problem is perhaps unsolvable. As in many protocols, the strength of trust is only as weak as its weakest point. When the security of a middleman server breaches the whole chain of trust in the messenger application is broken. We must not rely security on our understanding of any trustworthy objects.

References

[1] L. Ceci, "Global number of mobile nessaging users 2018-2025," Statista, Nov. 2021.

[2] R. M, Albrecht, J Blasco, R. B. Jensen and L. Marekova, "Collective information security in large-scale urban protests: the case of Hong Kong," Procs of 30th USENIX Security Symposium, pp. 3363-3380, Aug. 2021.

[3] M. Weidner, M Kleppmann, D. Hugneroth and A. R. Beresford, "Key agreement for decentralized secure group messaging with strong security guarantees," Procs of 2021 ACM SIGSAC Conference on CCS, pp.2024-2045, Nov. 2021.

[4] R. Barnes et. al., "The Messaging Layer Security (MLS) Protocol draft-ietf-mls-protocol-12," Internet Engineering Task Force, Oct. 2021.

[5] K. Cohn-Gordon et al., "On ends-to-ends encryption asynchronous group messaging with strong security guarantees," Procs of 2018 ACM SIGSAC Conference on CCS, pp.1802-1819, Oct. 2018.

[6] P. Rösler, C Mainka, J Schwenk, "More is less: on the end-to-end security of group chats in Signal, WhatsApp, and Threema," Procs of 2018 IEEE European Symposium on Security and Privacy, pp. 415-429, Apr. 2018.

[7] K. Elmaghraby, T Dimitriou, "Blockchain-based fair and secure certified electronic mail without a TTP," IEEE Access, Vol. 9, pp.100708-100724, Jul. 2021.

[8] M. Chase, A. Deshpande, E. Ghosh, H Malvai, "SEEMless: secure end-to-end encrypted messaging with less trust," Proceedings of 2019 ACM SIGSAC Conference on Computer and Communications Security, pp.1639-1656, Nov. 2019.

[9] Vipul Goyal, "Reducing trust in the PKG in identity based cryptosystems," Procs of Annual International Cryptology Conference, pp.430-447, Aug. 2007.

[10] Identity-based Encryption", available at http://en.wikipedia.org/wiki/Identity_based_encryption.