Breaking Message Integrity of an End-to-End Encryption Scheme of LINE*

Takanori Isobe[1] and Kazuhiko Minematsu[2]

[1] University of Hyogo, Japan. takanori.isobe@ai.u-hyogo.ac.jp
[2] NEC Corporation, Japan. k-minematsu@ah.jp.nec.com

Abstract. In this paper, we analyze the security of an end-to-end encryption scheme (E2EE) of LINE, a.k.a Letter Sealing. LINE is one of the most widely-deployed instant messaging applications, especially in East Asia. By a close inspection of their protocols, we give several attacks against the message integrity of Letter Sealing. Specifically, we propose forgery and impersonation attacks on the one-to-one message encryption and the group message encryption. All of our attacks are feasible with the help of an end-to-end adversary, who has access to the inside of the LINE server (e.g. service provider LINE themselves). We stress that the main purpose of E2EE is to provide a protection against the end-to-end adversary. In addition, we found some attacks that even do not need the help of E2E adversary, which shows a critical security flaw of the protocol. Our results reveal that the E2EE scheme of LINE do not sufficiently guarantee the integrity of messages compared to the state-of-the-art E2EE schemes such as Signal, which is used by WhatApp and Facebook Messenger. We also provide some countermeasures against our attacks. We have shared our findings with LINE corporation in advance. The LINE corporation has confirmed our attacks are valid as long as the E2E adversary is involved, and officially recognizes our results as a vulnerability of *encryption break*.

Key words: E2EE, LINE, key exchange, group message, authenticated encryption

1 Introduction

1.1 Background

An end-to-end encryption (E2EE) is a secure communication scheme for messaging applications where the only people who are communicating can send and read the messages, i.e. no other party, even service providers of communication system, cannot access to the cryptographic keys needed to encrypt the message, and decrypt the ciphertexts. After Snowden's revelation, the E2EE receives a lot of attentions as a technology to protect a user privacy from mass interception and surveillance of communications carried out by governmental organizations such as NSA (National Security Agency).

Apple first supported an E2EE scheme in their widely-deployed messaging application, iMessage, where a message that is compressed by gzip is encrypted by a sender's secret key and distributed with a digital signature for the guarantee of the integrity to the recipient. Unfortunately, several security flaws of the initial iMessage are pointed out in 2016 [26]. A Signal is a new E2EE protocol for instant messaging. The core of the Signal protocol has been adopted by WhatsApp, Facebook Messenger, and Google Allo. A novel technology called ratcheting key update structure enables advanced security properties such as perfect forward secrecy and so-called post-compromise security [21]. Since Signal is an open-source application and its source code for Android and iOS are available on Github [31], its security has been studied well from the cryptographic community [19, 20, 32].

LINE is one of the most widely-deployed messaging applications, especially in East Asia. The number of monthly active users of four key countries, namely Japan, Taiwan, Thailand and Indonesia is about 217 million in January 2017. Their market is still growing, and at the same time their applications are expanding such as banking, payment, shopping, music services. Indeed, it is currently a key platform for any IT services in these countries. For example, Japanese government recently launched a portal cite for management of Japanese social security number, called

* Full version of the paper published in the proceedings of ESORICS 2018

"My number" or "Individual number", in cooperation with LINE [7]. In fact, LINE dominates the market of mobile messaging application in Japan. It is estimated that more than 85% smartphone users in Japan are regularly using LINE in 2017 [8].

In 2015, LINE announced their new E2EE scheme, called *Letter Sealing*, for a pairwise secure communication between the end users [6]. It became a default feature in 2016, and was also deployed for the group messaging service. While the specification of Letter Sealing was initially not public, after some details were revealed by the reverse engineering [23], a whitepaper describing the high-level specification was published in 2016 [29]. Letter Sealing consists of key generation and registration, client-to-client key exchange, and message encryption phases. To the best of our knowledge, there is only one result of its security analysis by Espinoza et. al [24] which pointed out the lack of forward secrecy and the feasibility of reply attack.

1.2 Our Contribution

In this paper, we show several attacks on the E2EE scheme of LINE by a close inspection of their protocols described in the whitepaper [29] and some reverse engineering results publicly available [23, 24]. Our attacks exploit vulnerabilities of the group messaging protocol, the key exchange phase, and the message encryption phases, respectively. Some of them were already pointed out in the previous work [23, 24].

Impersonation and Forgery Attacks on Group Message Encryption. We show impersonation and forgery attacks on the group message encryption scheme by a *malicious group member*, who is a legitimate member of a target group but aims to break the integrity of the message sent by an honest group member. These attacks exploit a vulnerability of the key derivation phase in the group message encryption such that any group member, even a malicious member, is able to derive an encryption key of another member for a group messaging without any knowledge of a target member's secret. By exploiting this vulnerability, a malicious member is able to send a message to a group as if it was from an honest member, that is, an impersonation attack. Moreover, if a malicious member colludes with an *E2E adversary*, who bypasses client-to-server encryption (e.g. LINE themselves) or if a malicious member herself is the E2E adversary, she freely modifies a message sent by an honest member without being noticed by the other members about the fact that it was tampered, that is, a forgery attack.

Considering a rapid growth of group message services of LINE, there is a significant risk that a group member is malicious. We believe that our attacks crucially threaten the fundamental security of the group messaging service of LINE. Indeed, other major messaging applications based on Signal (e.g. WhatApp and Facebook Messenger) and Apple's iMessage guarantee the security against a malicious member for group messaging [19, 32].

Malicious Key Exchange Attacks. We propose a malicious key exchange of the one-to-one E2EE to mount impersonation attacks by a *malicious user* who is a legitimate member of a target group but aims to break the integrity of the message in the other sessions. Exploiting vulnerabilities both of the key exchange and the message encryption phases, a malicious user C establishes a malicious E2EE session with a victim B in which a shared secret between C and B is the same as the one used in a different E2EE session between B and another victim A. Then, the malicious user C is able to impersonate victims A and B. More specifically, the malicious user C is able to

- send a message, that is originally sent to C from B, to A as a message from B (impersonation attack 1),
- send a message, that is originally sent to B from A, to B as a message from C (impersonation attack 2).

Unlike the replay attack [24], our attacks can send a message derived in a session to a different session. It might cause crucial security problems, e.g. a private information of the victim might be unintentionally disclosed.

Our impersonation attacks are possible by a malicious user who has a trusted relationship with one of the two victims between which a pairwise secure channel is already established. For

Table 1. Comparison with previous [24] and our attacks: *E2E* is an adversary who has access to the inside of the LINE server. *Malicious member* is a legitimate member of a target group but tries to break the integrity of the message sent by an honest member. *Malicious user* is an end user who is trusted by the victim B where the victim A and B have a pairwise secure connection which is an attack target by the malicious user.

Target	Attack Type	Adversary	Reference
One-to-One Encryption	Replay	E2E	[24]
Group Encryption	Impersonation	Malicious Member	Sec. 4.1
	Forgery	Malicious Member w/ E2E	Sec. 4.2
One-to-One Encryption	Impersonation 1	Malicious User	Sec. 5.2
	Impersonation 2	Malicious User w/ E2E	Sec. 5.3
One-to-One and Group Encryption	Forgery	E2E	Sec. 6

example, a victim A and the malicious user C are B's friend in the real world, or are company's accounts that B can trust, while the victim A does not need to trust C, and might even not know C. Importantly, even a normal user, who is *not* an E2E adversary (LINE), is able to perform the impersonation attack 1 as long as the above assumption holds. We think that these situations can happen in the real-world use cases of LINE. On the other hand, for the impersonation attack 2, a malicious user needs to collude with E2E adversary (LINE) to bypass the server-client encryption.

Forgery Attack on Authenticated Encryption Scheme. Finally, we evaluate the security of an authenticated encryption (AE) scheme used in the message encryption phase, which combines AES-256 and SHA-256 in a non-standard way. We show that the E2E adversary is able to mount a forgery attack, i.e. the adversary made a forgery message which is accepted as valid by the recipient. Compared to the previous attacks, this forgery attack does not require the assumption that the adversary has a trusted relation to the victim in advance. Thus, any user in the one-to-one and group message encryptions could be the victim of this attack. Furthermore, we give a rigorous security analysis of AE in LINE as a general-purpose authenticated encryption.

This attack is not much practical in terms of the time complexity of the attack, because it needs 2^b offline computation and 2^d online computation for $b + d = 128$, implying **64-bit security**. The popular AE schemes using 128-bit block cipher also have 64-bit security, however, this is about the online data complexity (i.e. it is secure if one key is used with data smaller than 2^{64}). Thus, the implications are quite different. For example, AE in LINE can be broken with 2^{80} offline computation plus 2^{40} online computation, which is not the case of generic composition of CBC mode using AES-256 plus HMAC-SHA-256, as used by Signal.

The attack with 2^b offline computation and 2^d online computation for $b+d = 128$ may be within reach by powerful national organizations such as NSA. Compared to AE schemes used in Signal and the stare-of-the-art AE schemes, we show that AE in LINE does not provide a sufficient-level security from the cryptographic point of view.

Summary. Table 1 summarizes our results. All of our attacks are possible for the E2E adversary, which is the original target adversary of the E2EE scheme, and more powerful than the previous one [24] as impersonation and forgery attacks violate the integrity of the message, which is one of the fundamental security properties. Some of our attacks are performed by not only the E2E adversary but also weaker adversaries such as a malicious group member and a malicious user. Therefore, E2EE of LINE does not provide the integrity of the message, which is one of the fundamental security requirements of E2EE.

Responsible Disclosure. In December 2017, we delivered our results in this paper to LINE Corporation via the LINE security bug bounty program. They acknowledged that *all of our attacks are feasible with the help of E2E adversary*, and officially recognized our findings as a vulnerability

of *encryption break* in the bug bounty program. LINE Corporation told us (and granted us to make it public) that they have a plan to change the key exchange protocol, the group messaging scheme, and the authenticated encryption scheme to improve the security of Letter Sealing in the near future.

On the other hand, they told us that our attacks by a malicious member (Section 4.1) and a malicious user (Section 5.2) not colluding with LINE Corporation (i.e. E2E adversary) can be mitigated by certain server-side countermeasures. For instance, a malicious key exchange can be prevented by checking the duplication of public keys in the LINE key server. Since they hope to keep the details of their server-side countermeasures secret, we will not explain them here. We consider that it seems difficult to directly apply our attacks without the help of the E2E adversary after we informed LINE Corporation in December 2017. That is, these attacks are now applicable only with the help of E2E adversary who can bypass the server-side countermeasures.

We remark that the Signal protocol guarantees these securities against a malicious member and a malicious user without relying on the server-side countermeasures, meaning that the E2EE cryptographic protocol prevents these attacks. This is a significant difference from the security point of view, because the server side countermeasures cannot be formally analyzed by the 3rd party, and cannot rely on the well-evaluated computational hard problems. In addition, server-side countermeasures are not an essential protection for the E2EE, since the server may be malicious and the clients do not have a mean to verify if they are correctly implemented by the server. Otherwise, the server-client encryptions are enough.

2 Specification of E2EE Scheme of LINE

In this section, we give a high-level description of the E2EE scheme of LINE, called Letter Sealing. The whitepaper published by LINE Corporation [29] describes two types of E2EE schemes: a one-to-one message encryption and a one-to-N group message encryption.

2.1 One-to-One Message Encryption

The one-to-one message encryption scheme of LINE consists of key generation and registration, client-to-client key exchange, and message encryption phases.

Key Generation and Registration Phase. When a LINE application is launched at the first time, each client application generates a key pair of (sk, pk) for the key exchange, where sk and pk are a secret key and a public key for Elliptic curve Diffie–Hellman (ECDH) based on Curve25519 [13], respectively. The client stores the secret key sk into application's private storage area, and registers the public key pk with a LINE messaging server. The server associates the public key pk with the currently authenticated client and sends back a unique key ID to the client. Each key ID is bound to a user and includes the version information of the key.

Client-to-Client Key Exchange Phase. To start a session for exchanging messages between a client and a recipient, the client shares the key called *SharedSecret* with the recipient as follows.

1. Retrieve the recipient public key pk_r from a LINE messaging server.
2. Generate a shared secret *SharedSecret* from the public key pk_r and a client secret key sk_c by ECDH over Curve25519 as

$$\mathit{SharedSecret} = \mathrm{ECDH}_{\mathrm{Curve25519}}(sk_c, pk_r),$$

   where $\mathrm{ECDH}_{\mathrm{Curve25519}}$ is a key exchange function (see [13] for details).

At the same time, the recipient generates the same *SharedSecret* with the recipient secret key sk_r and the client public key pk_c as

$$\mathit{SharedSecret} = \mathrm{ECDH}_{\mathrm{Curve25519}}(sk_r, pk_c).$$

Figure 1 illustrates this key exchange phase.

To make sure that the retrieved public key is a correct one, the fingerprint of a recipient's public key can be displayed in the device. Users can verify it out-of-band.

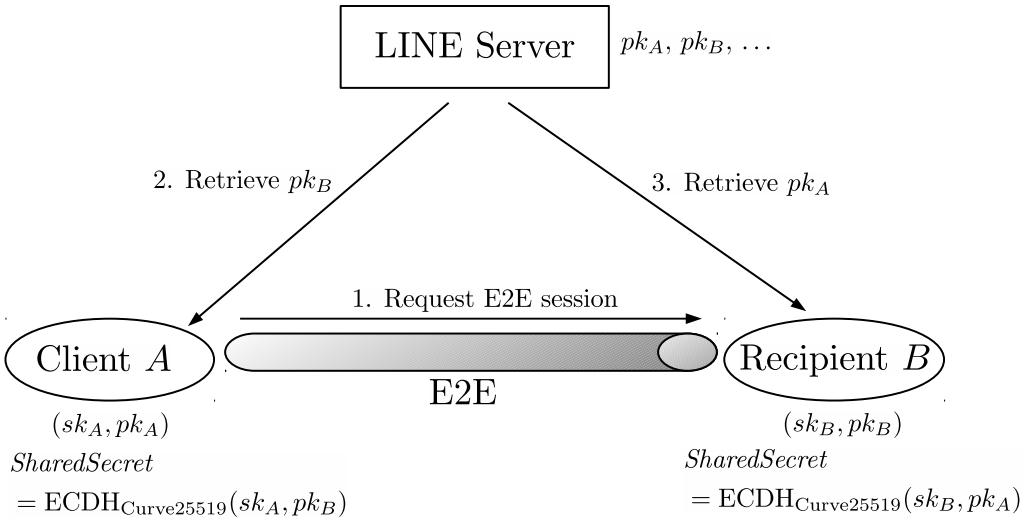

Fig. 1. Key exchange phase based on ECDH over Curve25519

Message Encryption Phase. A message is encrypted by a unique pair of a 256-bit key K_e and a 128-bit IV (Initialization Vector) IV_e, generated for each message. K_e and IV_e are derived from *SharedSecret* and a randomly-chosen 8-byte *salt* as follows.

$$K_e = \mathrm{SHA}_{256}(\mathit{SharedSecret} \,\|\, \mathit{salt} \,\|\, \texttt{Key}), \tag{1}$$

$$IV_{\mathrm{pre}} = \mathrm{SHA}_{256}(\mathit{SharedSecret} \,\|\, \mathit{salt} \,\|\, \texttt{IV}), \tag{2}$$

$$IV_e = IV_{\mathrm{pre}}^{\mathrm{left}} \oplus IV_{\mathrm{pre}}^{\mathrm{right}}. \tag{3}$$

Here, $\mathrm{SHA}_{256}(\cdot)$ denotes SHA-256 hash function that outputs a 256-bit digest from an arbitrary-length input [5], and IV_{pre} is a 256-bit variable, and $IV_{\mathrm{pre}}^{\mathrm{left}}$ and $IV_{\mathrm{pre}}^{\mathrm{right}}$ are left and right 128-bit values of IV_{pre}, respectively, i.e. $IV_{\mathrm{pre}} = IV_{\mathrm{pre}}^{\mathrm{left}} \,\|\, IV_{\mathrm{pre}}^{\mathrm{right}}$. The constants `Key` and `IV` denote the corresponding ASCII strings in base64 [23].

A message M is encrypted with K_e and IV_e, and a ciphertext C is obtained as

$$C = \mathrm{CBC}[E](K_e, IV_e, M),$$

where $\mathrm{CBC}[E](K, IV, M)$ denotes CBC encryption mode with AES-256 that takes a 256-bit key K and a 128-bit IV IV, and an arbitrary-length message M as inputs, and outputs a ciphertext C. See [1, 2] for details of AES-256 and CBC mode. A padding scheme is needed in case the bit length of M is not a multiple of 128, however it is not described in the whitepaper.

Next, the ciphertext C is hashed by a variant of SHA_{256} called SHA'_{256} defined as

$$V = \mathrm{SHA}_{256}(C), \text{ and } T_{\mathrm{pre}} = V_{\mathrm{pre}}^{\mathrm{left}} \oplus V_{\mathrm{pre}}^{\mathrm{right}},$$

where V is a 256-bit value, and $V_{\mathrm{pre}}^{\mathrm{left}}$ and $V_{\mathrm{pre}}^{\mathrm{right}}$ are left and right 128-bit values of V, and T_{pre} is a 128-bit variable. Then, a 128-bit message authentication tag T is computed by

$$T = E(K_e, T_{\mathrm{pre}}),$$

where $E(K, M)$ denotes an encryption of 128-bit M using AES-256 with 256-bit key K. Figure 2 shows the overview of the authenticated encryption scheme of LINE.

Finally, the client sends the packet D including the ciphertext C and the tag T with associated data (AD). The form of packet D is as follows.

$$D = \mathsf{version} \,\|\, \mathsf{content\ type} \,\|\, \mathit{salt} \,\|\, C \,\|\, T \,\| \\ \mathsf{sender\ key\ ID} \,\|\, \mathsf{recipient\ key\ ID}. \tag{4}$$

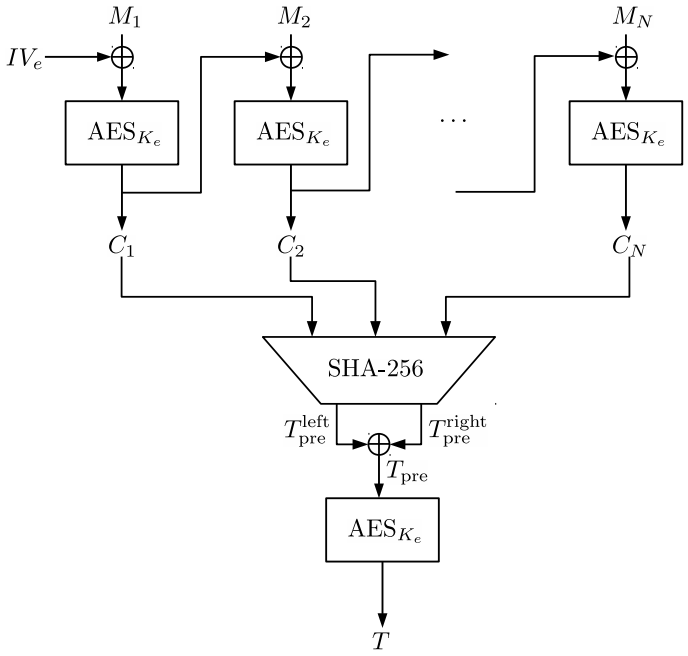

Fig. 2. Authenticated encryption scheme of LINE

Here, AD consists of version, content type, sender key ID and recipient key ID. The first two fields serve to identity the Letter Sealing version used to create the message. The recipient uses the sender key ID to retrieve the public key used to encrypt the message. The recipient key ID value helps to verify that the message can be decrypted using the current local private key. Messages that are processed by a previous key pair (such as one used before migrating to the current device) cannot be decrypted. To facilitate a device migration, the client automatically requests the recent messages processed by a previous key pair to be resent. Once the recipient received the packet D from the client, he derives the same key K_e, and IV IV_e from the shared secret *SharedSecret* as described above. Next, he calculates the tag T from the received ciphertext C, and compares it with the tag value included in the message. If they match, the contents of the message M is decrypted and displayed. Otherwise, the message is discarded.

2.2 One-to-N Group Message Encryption

In the one-to-N group messaging, a group key K_g is shared with all group members via one-to-one message encryption channels. The first member who starts a end-to-end group messaging generates K_g and shares it with all group members as follows.

1. Generate a pair of a secret key sk_g and a public key pk_g for ECDH over Curve25519, where sk_g is used as a group key K_g.
2. Retrieve public keys of all group members from the LINE server, and calculate N shared secrets *SharedSecret* for all members from own private key and a public key of each member to establish a one-to-one message encryption to each member.
3. Broadcast k_g to all members via one-to-one message encryption channels.

Whenever members join or leave the group, K_g is renewed and shared with the group.

Once K_g is shared with all members, a member A who wants to send a message to the group derives a 256-bit encryption key K_e^A and a 128-bit IV IV_e^A from K_g and A's public key pk_A as

follows.

$$SharedSecret_g^A = \mathrm{ECDH}_{\mathrm{Curve25519}}(K_g, pk_A), \tag{5}$$

$$K_e = \mathrm{SHA}_{256}(SharedSecret_g^A \,\|\, salt \,\|\, \texttt{Key}), \tag{6}$$

$$IV_{\mathrm{pre}} = \mathrm{SHA}_{256}(SharedSecret_g^A \,\|\, salt \,\|\, \texttt{IV}), \tag{7}$$

$$IV_e = IV_{\mathrm{pre}}^{\mathrm{left}} \oplus IV_{\mathrm{pre}}^{\mathrm{right}}. \tag{8}$$

The message data is encrypted and formatted as described in the one-to-one message encryption with the only difference that the recipient key ID field is replaced with the key ID of the group's shared key.

3 Security Model of E2EE

In this section, we explain adversary models and security requirements of E2EE.

3.1 Adversary Model

In a E2EE scheme, no one except a client and a recipient can be trusted, i.e. there is no any trusted third party, and even a service provider (e.g., LINE) is a potential adversary. In this setting, the client-to-server transport encryption (cf. Section 3 in [29]) is useless, as the adversary can control the LINE server which stores the secret key for the client-to-server transport encryption. We call such an adversary *E2E adversary*.

Definition 1 (E2E adversary) *An E2E adversary is able to intercept, read and modify any messages sent over the network, and has full access to the messaging server, i.e. bypasses the client-to-server encryption.*

Generally, the E2E adversary is assumed to have a very strong computational power to capture the powerful national organizations for intelligence, such as NSA and GCHQ (Government Communications Headquarters), because one of the objectives of E2EE is to protect user privacy from the mass interception and surveillance of communications against such organizations. Indeed, WhatsApp was unable to comply with Brazilian government demands for user's plaintext messages [17], because of its E2EE. Apple also opposes the order from FBI to unlock the iPhone due to the basic principle of the E2EE [22]. Previous replay attack on LINE [24] also assumes the E2E adversary. In addition, we define a weaker adversary, *malicious user*.

Definition 2 (Malicious User) *An malicious user is a legitimate of one-to-one E2EE but she tries to break one of the subsequently defined security goals of the other E2EE session by maliciously manipulating the protocol.*

A malicious user is much weaker than the original targeted adversary of the E2E adversary, because any user is potentially a malicious user. We will show that even such a weaker adversary than the original target of E2EE can attack the E2EE protocol of LINE.

The existence of *malicious group member* in One-to-N group message encryption must be taken into consideration, as already discussed in some recent papers [19, 32].

Definition 3 (Malicious group member) *A malicious group member, who is a legitimate group member and possesses a shared group key, tries to break the subsequently defined security goals by deviating from the protocol.*

Note that a malicious user and a malicious group member can collude with the E2E adversary in the security model of E2EE, or the E2E adversary herself can be a malicious user and a malicious group member.

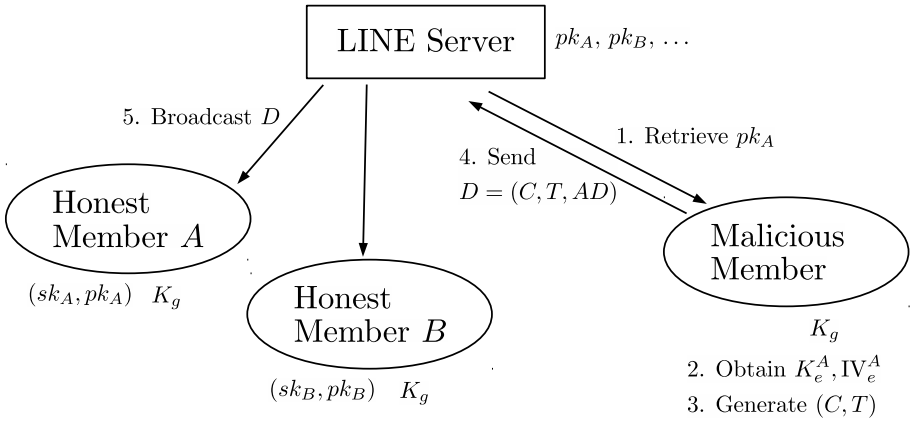

Fig. 3. Impersonation attack

3.2 Security Goals

We explain the following two fundamental security goals of E2EE.

Definition 4 (Confidentiality) *Only the two participants of pair-wise messaging or legitimate group member of group messaging can see the message plaintext.*

Definition 5 (Integrity) *If a message is received and successfully validated, then it was indeed sent by the given sender, i.e., other users cannot plant messages into it and they can not modify it.*

We remark that more advanced security properties of E2EE can be found in the literature, such as forward secrecy, post-compromise security, and traceable delivery [19–21, 32]. Still, an E2EE scheme should guarantee at least the confidentiality and the integrity against the E2E adversary. Since a malicious user and a malicious group member are weaker than the E2E adversary, a secure E2EE scheme should also be secure against any attack by them.

Since a malicious member has a group shared key K_g for the group message encryption, she is able to decrypt any group message. Thus, it is natural to assume that her purpose is to break the integrity.

4 Impersonation and Forgery Attacks on Group Message

This section gives impersonation and forgery attacks on the group message encryption by a malicious group member. Both attacks exploit a following vulnerability of the key derivation phase in the group message encryption.

Vulnerability 1 (Key Derivation of Group Message) *The key and IV for the symmetric-key encryption are derived from a group-shared key K_g and sender's public information.*

A group member A, who wants to send a message to a group, first computes a shared secret $\mathit{SharedSecret}_g^A$ from a group key K_g and sender's public key pk_A. Since a malicious member also possesses K_g and is able to retrieve sender's public key pk_A from the LINE server, she can compute $\mathit{SharedSecret}_g^A$ by which, thus is able to derive K_e^A and IV_e^A as shown in Equations (6) and (8). Hence, a malicious member is able to compute any group member's key and IV for the group message.

4.1 Impersonation Attack

Exploiting Vulnerability 1, a malicious group member impersonates an honest member A in the group message encryption as follows.

1. Retrieve A's public key pk_A from the LINE server.
2. Derive K_e^A and IV_e^A from a group-shared key K_g, pk_A and a randomly-generated *salt*.
3. Generate a ciphertext C and a tag T of the message M that the malicious group member chooses.
4. Prepare a packet D following the Equation (4) by properly choosing AD where sender key ID is set to the victim A's one.
5. Broadcast D to all members via the LINE server.

Figure 3 shows the overview of our impersonation attack. Since the tag T is generated by the valid key of the member A, group members except A do not notice that it is created by the malicious member. When A sees this fake message, A should notice, however, there is no formal way to refute. Therefore, this attack reveals that *the group message encryption of LINE does not provides the authenticity of the message against a malicious member.*

Furthermore, if the malicious member colludes with the LINE server (E2E adversary), it is possible to broadcast D, made by the malicious member, to all members except the victim A. Then, the victim A does not notice that such a attack is mounted by the malicious member.

4.2 Forgery Attack

If a malicious member intercepts a group message that the honest member A sends, she is able to mount a forgery attack as follows (see also Fig. 4).

1. Intercept a packet D sent by the member A, by watching the communication between the victim A and the LINE server.
2. Compute K_e^A and IV_e^A from a group key K_g and a public key pk_A, and *salt* which is derived from D.
3. Decrypt it with K_e^A and IV_e^A and modify the message M of the victim A.
4. Re-encrypt the modified message M' with K_e^A and IV_e^A to generate a new ciphertext C' and a tag T'.
5. Broadcast D' including C', T' and associated data to all members except A, and send the original D to the victim A via the LINE server.

To mount the above attack, a malicious member (C) must intercept packet D between the victim A and the server before it is sent to all members by the LINE server. Since this channel is protected by the client-to-server transport encryption, the malicious member is not able to get D that is encrypted by only K_e^A and IV_e^A. Recall that the E2E adversary is able to bypass the client-to-server transport encryption. If the malicious member colludes with the E2E adversary or the malicious member herself is the E2E adversary, this forgery attack is successful. Furthermore, since the E2E adversary sends unmodified packet D to only the victim A, the victim A does not notice the forgery attack is mounted. Thus, this attack shows that *the E2EE of group messages of LINE does not satisfies the integrity of the message when a malicious member colludes with the E2E adversary.*

4.3 Discussion

The specification of LINE currently allows a group of up to 500 members (as of April 2018), and the applications of the group messaging service are rapidly expanding, from private to business, banking, advertising, payments, and social network service. In some use cases, it is hard to strictly check the identity of group members, e.g. an online group for common interest and hobby. We believe that there is a significant risk that a group member is malicious. Furthermore if the LINE official account is involved in the target group, the scenario of the forgery attack, i.e., malicious member collude with the E2E adversary, is easily realized. In this case, the impersonation attack is also feasible.

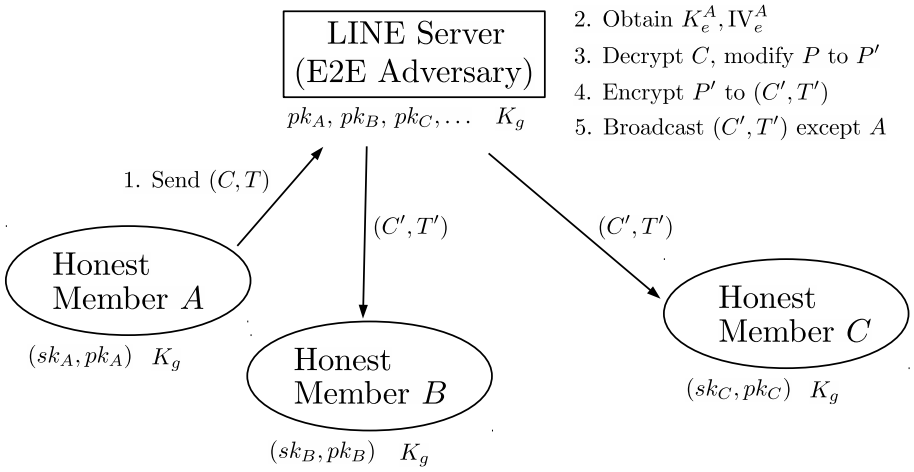

Fig. 4. Forgery attack on the group message

4.4 Countermeasures

Several countermeasures have been employed by the messaging applications other than LINE. The E2EEs of WhatsApp and Facebook Messenger, which are the most popular messaging applications, are based on Signal protocol [19, 20]. In the group message of the Signal protocol, each group member sends Sender key to all group members via pair-wise secure E2EE at the first time. Here, Sender key contains an encryption key of symmetric-key encryption schemes (e.g. AES-CBC) and a verification key of the digital signature of the sender. When a member sends a message to a group, he encrypts the message using an ephemeral key that is derived from the encryption key, and signs the ciphertext by sender's signature key. The receiver is able to check whether the message is from the sender by verifying the signature by the sender's verification key in Sender key. We refer to [20] for the details of key exchange protocol. Since a malicious group member does not knows target sender's signature key, she is not able to make a signature of the ciphertext. Thus, our attacks are not applicable to the Signal protocol. Attaching the sender's digital signature is a typical countermeasure against our attacks.

Apple's iMessage implements group messaging using pairwise channels. That is, a message is sent to all group members via each one-to-one encryption channel. Although this also thwarts our attacks, it might be less efficient than Signal protocol.

5 Malicious Key Exchange Attack on One-to-One Message Encryption

This section presents a malicious key exchange of the one-to-one message encryption, which leads to an impersonation attack. Our attacks exploit the following vulnerabilities of the key exchange and the message encryption phases.

Vulnerability 2 (No key confirmation) *In the client-to-client key exchange phase, there is no key confirmation.*

In the client-to-client key exchange phase, after individually computing a shared secret *SharedSecret* in both client and recipient sides, there is no key confirmation phase between the client and the recipient. Thus, even if *SharedSecret* is not correctly shared between the client and the recipient, the client is not able to confirm that the recipient possess a shared secret, and vice versa.

Vulnerability 3 (Integrity of packet) *In the message encryption phase, the integrity of the elements of associated data in a packet D, such as* sender key ID *and* recipient key ID*, is not guaranteed.*

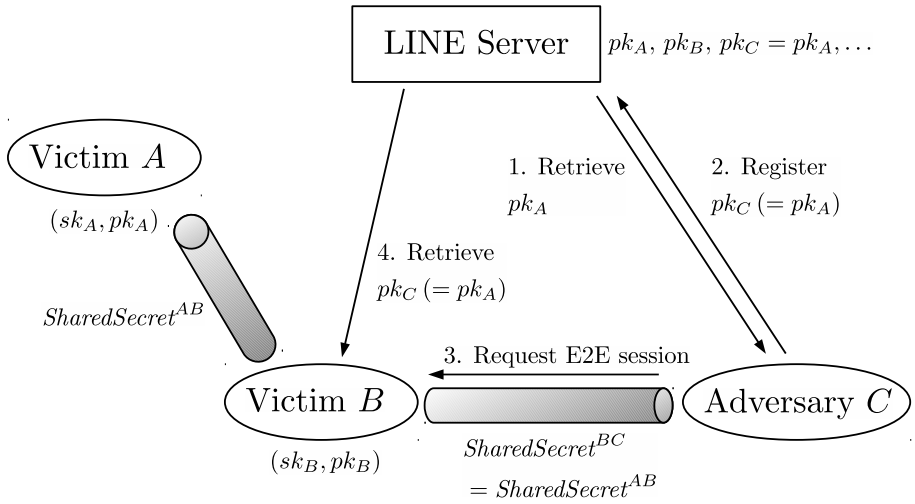

Fig. 5. Malicious key exchange

When computing a message authentication tag, T, $\mathrm{SHA}'_{256}(\cdot)$ takes a ciphertext C as a sole input, and the associated data is just appended to the ciphertext C and the tag T. Hence, the recipient is unable to verify the integrity of associated data. This vulnerability has been pointed out in the previous works [23, 24], and used for replay attacks.

5.1 Malicious Key Exchange

Our malicious key exchange is a variant of unknown key share attack [15]. An adversary (malicious user) C shares a secret key with a victim B where the shared secret key is the same as one used in a different session between victims A and B, while the victim A does not know the fact. Our attack is performed under the following assumptions.

Assumption 1 *Two victims A and B have already established a pairwise E2EE session.*

Assumption 2 *An adversary (malicious user) C is able to establish another E2EE session with B which has not been established yet.*

In other words, a victim B trusts both a victim A and a malicious user C, e.g. A and C are B's friend in the real world, or are company's accounts that B can trust, while the victim A does not need to trust C and might even not know the adversary C. Under these assumptions, C attacks on the E2EE session between A and B. Specifically, C tries to establish a fake E2EE session with B where the key and IV are the same as those used in the A-B session.

To establish this fake session, C performs the following procedures in the key generation and registration phases by exploiting Vulnerability 2 (see Fig. 5).

1. Retrieve a public key pk_A from the LINE server.
2. Registers pk_A in the LINE messaging server as C's public key.
3. Request a new E2EE session with B.

After that, the victim B computes a shared secret $SharedSecret^{BC}$ for the one-to-one, E2EE between B and C as

$$SharedSecret^{BC} = \mathrm{ECDH}_{\mathrm{Curve25519}}(sk_B, pk_A).$$

which is the same as the shared secret between A and B. Here, C does not know the value of $SharedSecret^{BC}$, because she does not know sk_A. Nevertheless, due to the lack of the key confirmation (Vulnerability 2), the protocol will not abort, and an E2E session will be established between B and C.

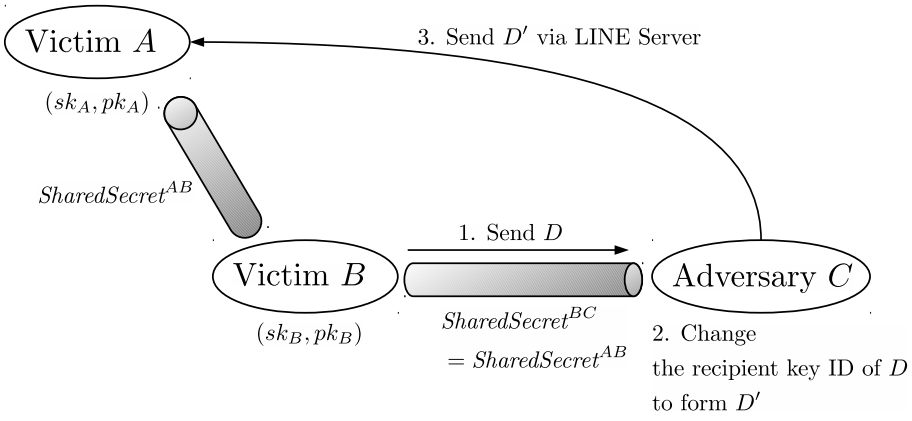

Fig. 6. Impersonation attack against B

5.2 Impersonation Attack 1: Impersonating B

After the malicious key exchange, the malicious user C is able to send a packet to A which was originally sent from B to C by impersonating B.

1. Receive a packet D which is sent from B to C.
2. Create a new packet D' where recipient key ID is modified from C to A.
3. Send the packet D' to A as a message from the malicious user C via the LINE server.

Figure 6 shows the overview of impersonation attack against B. Due to the lack of the integrity check for recipient key ID in the associated data (Vulnerability 3), the victim A believes that the message is sent from B instead of C while B believes that the message is sent to C and does not notice the message is sent to A.

5.3 Impersonation Attack 2: Impersonating A

The malicious user C can also send a message to B by impersonating A.

1. Intercept a packet D which is sent from A to B.
2. Create a new packet D' where sender key ID is modified from A to C.
3. Send the packet D' to B as a message from the adversary C.

If the adversary colludes with an E2E adversary or she herself is an E2E adversary, she can bypass the client-to-server encryption, and intercept a packet D. Indeed, a packet D is sent to B from A via the LINE server.

5.4 Discussion

Our impersonation attack 1 is feasible as long as the assumption 1 and 2 hold. This implies that even a normal user, who is *not* an E2E adversary, is able to mount the attack. Suppose that the victim B sends a very personal request (e.g. asking for debt) to the malicious user C, then C can send a request to the victim A by impersonating B. Then, B's sensitive information is *unintentionally* disclosed to A as a valid message from B. It is a significant problem of privacy.

The impersonation attack 2 is feasible if the assumption 1 and 2 hold and the client-to-server encryption is bypassed. For example, if the LINE official account is an E2E adversary, this account is able to mount the impersonation attack 2.

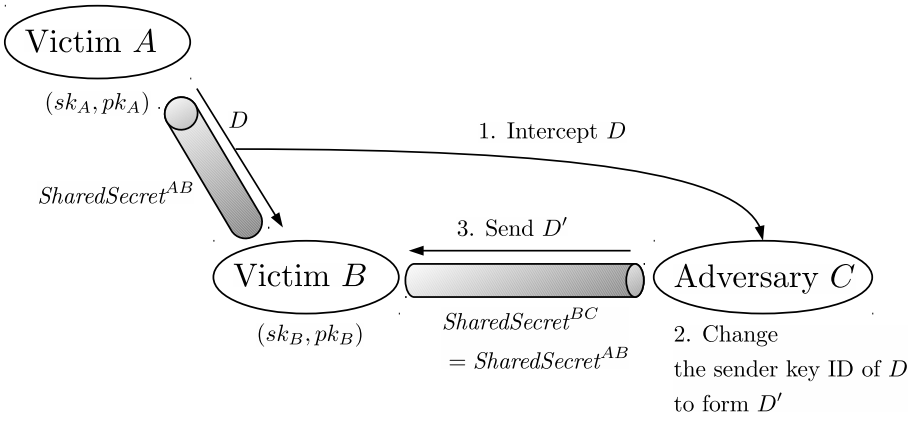

Fig. 7. Impersonation attack against A

5.5 Countermeasures

There are several countermeasures against these attacks. The first one is to implement the key confirmation phase after the key exchanges to get rid of Vulnerability 2. It prevents from the adversary to mount a malicious key exchange, i.e. the message exchanging session does not start if the two parties do not correctly share their secret values. Indeed, the key exchange protocol of TLS1.3 provide a key confirmation [25], and prevents such attacks.

The second countermeasure is to guarantee the integrity of associated data by adding these to a input of $\mathrm{SHA}'_{256}(\cdot)$ for computing a message authentication tag T, which is a common practice for authenticated encryption, provided message authentication itself is secure. Then, even if the malicious key exchange is succeeded, the adversary is not able to change sender key ID and recipient key ID.

In Signal protocol [19,20], the computation of a shared secret in the key exchange phase involves One-Time Pre Key, which is used only once. This changes the shared secret at every execution of the key exchange even if the public keys are the same, thus our attacks do not work.

6 Security Evaluation of Message Encryption Scheme

This section evaluates the security of the authenticated encryption scheme in the message encryption phase (hereafter, we call it LINE-AE), and presents a forgery attack by the E2E adversary. Our forgery attack exploits the vulnerability of LINE-AE, which is an original authenticated encryption scheme, i.e. it is not a standard scheme such as generic composition of an encryption and a message authentication code (MAC), e.g. CBC mode and HMAC, or dedicated schemes (modes) such as GCM [4] and CCM [3].

Compared to the previous attacks in Section 5, this forgery attack does not require the assumption that the adversary has a trusted relation to the victim in advance. Thus, any user in the one-to-one and group message encryptions could be the victim of this attack.

6.1 Authenticated Encryption: LINE-AE

As shown by Section 2.1, LINE-AE first encrypts a message by CBC mode with AES-256, and generates a ciphertext C. After that, a tag T for the ciphertext C is computed as follows: the ciphertext C is hashed by $\mathrm{SHA}'_{256}(C)$, and then T_{pre}, which is an output of $\mathrm{SHA}'_{256}(C)$, is encrypted by ECB mode with AES-256.

Let LINE-MAC denote the tag generation function, that is, $E(K_e, \mathrm{SHA}'_{256}(C))$. Our attack exploits the following vulnerabilities of LINE-AE.

Vulnerability 4 (LINE-MAC) *A 128-bit intermediate value T_{pre} of LINE-MAC is computable without any secret information.*

In LINE-MAC, $\mathrm{SHA}'_{256}(\cdot)$ is a public function for which anyone is able to evaluate. A set of input/output pairs of $\mathrm{SHA}'_{256}(\cdot)$ can be computed by the adversary without any knowledge of the 256-bit key.

Vulnerability 5 (Same Key in encryption and LINE-MAC) *In the CBC encryption and* LINE-MAC*, the same key K_e is used for AES-256.*

For each message encryption, a 256-bit key K_e is given to not only AES-256 for CBC mode but also AES-256 ECB mode for LINE-MAC. This vulnerability was already pointed out in the previous works [23, 24], though they did not find any actual attacks based on it.

6.2 Forgery Attacks

We propose a forgery attack on LINE-AE. Assuming the E2E session between victim A and B, an E2E adversary collect the data between A and B and tries to create a forgery message for this session.

In the offline phase, the adversary precomputes a set of pairs of input/output of $(X, Y(= \mathrm{SHA}'_{256}(X)))$ in LINE-MAC by exploiting Vulnerability 4. In the online phase, she obtains the packets and extracts the sets of C, T and associated data sent by a victim, and she computes pairs of $(T_{\mathrm{pre}}(= \mathrm{SHA}'_{256}(C)), T)$ which are pairs of input/output of one-block AES-256. If T_{pre} matches with Y computed in the offline phase, a new valid pair of input and output of LINE-MAC is obtained as $(X, T(= E(K, \mathrm{SHA}'_{256}(X))))$ without knowing a 256-bit key K. If the adversary sends the pair of (X, T) with properly-chosen associated data, the victim A or B is not able to detect whether it is made by the adversary. Our attack consists of offline and online phases, see Figure 9). The detailed procedures are as follows.

Offline Phase.

1. Compute 2^b pairs of input/output of $(X, Y(= \mathrm{SHA}'_{256}(X)))$ in LINE-MAC.
2. Store these results into a table indexed by values of Y.

Online Phase.

1. Get 2^d sets of C, T and associated data sent by a victim.
2. Compute T_{pre} $(= \mathrm{SHA}'_{256}(C))$ from a set of C, T and additional data.
3. Check whether T_{pre} collides with Y in the table created in the offline phase. If a collision exists, obtain a new valid pair $(X, T(= E(K, \mathrm{SHA}'_{256}(X))))$ of LINE-MAC.
4. Repeat Steps 2 to 3 for all 2^d sets of C, T and additional data .

Evaluation. The offline phase requires 2^b hash computations and 2^b memory. The online phase requires 2^d data and 2^d hash computations. The success probability of our forgery attack is estimated as $2^{-128+b+d}$. The whole time complexity (Time) is $2^b + 2^d$ and data complexity (Data) is 2^d. Thus if $\mathsf{Data} \cdot \mathsf{Time} \geq 2^{128}$, our attack is successful with a high probability. We remark that the order of the offline and online can be changed. If the online phase is first, the memory consumption for storing $(T_{\mathrm{pre}}(= \mathrm{SHA}'_{256}(C)), T)$ is 2^d, and the offline phase does not requires memory.

This attack is not much practical in terms of the time complexity of the attack, because it needs 2^b offline computation and 2^d online computation for $b + d = 128$, thus 64-bit security. The popular AE schemes using 128-bit block cipher also have 64-bit security in terms of online data complexity (i.e. it is secure if one key is used with data smaller than 2^{64}), however, the implications are quite different. For example, AE in LINE can be broken with 2^{80} offline computation plus 2^{40} online computation, which is not the case of generic composition of CBC mode using AES-256 plus HMAC-SHA-256, as used by Signal.

In the following, we will explain some idea how to efficiently collect the required data to make our attack practical.

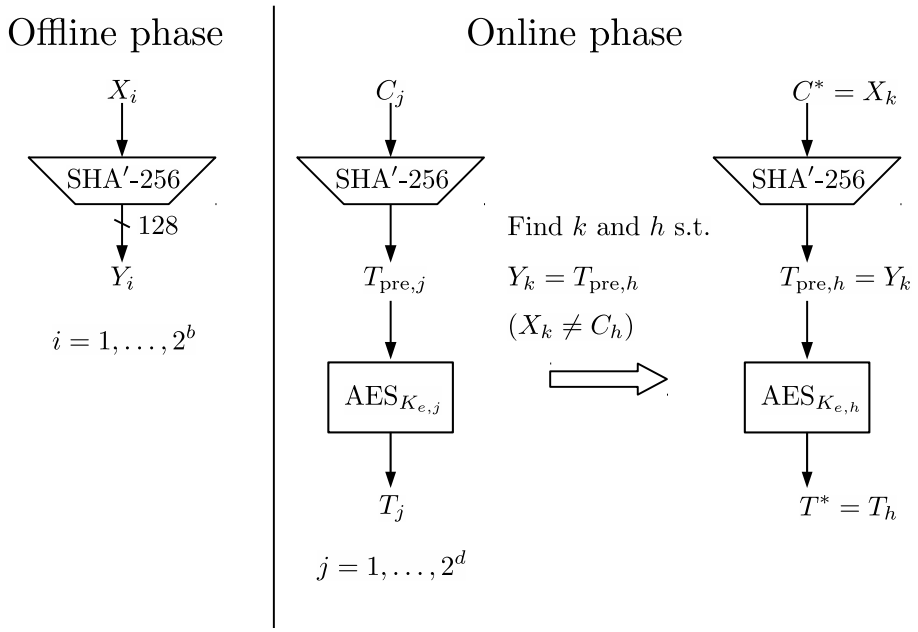

Fig. 8. Forgery Attack

Multi-User Setting. The essence of our forgery attack is to find a matching pair of (X, Y) and (T_{pre}, T) such that $Y = T_{\mathrm{pre}}$. Here, (X, Y), which is computed in the offline phase, is independent from the key K_e used in the computation of (T_{pre}, T). Thus, the precomutation table of (X, Y) in the offline phase can be used for any value of K_e in the online phase.

Since a encrypted key K_e is refreshed with a new *salt* by the key derivation function in each encryption call, we assume that K_e is randomly chosen in each message. It means that in the online phase, each (T_{pre}, T) is generated by different values of K_e. Therefore, there is no difference between *single user setting*, where an attack is successful if it compromises the security of a single target user, and *multiuser setting*, where an attack is successful if it compromises the security of one out of many users.

The multiuser setting enables increasing the number of available data for the E2E adversary. In the case of LINE, since all messages are send to all recipients via the LINE servers, all transactions of the E2EE are used for our attacks. In the multiuser setting, the adversary aims to make a forgery message of one of multiple users.

The number of monthly active users of four key countries, namely Japan, Taiwan, Thailand and Indonesia, is about 170 million ($= 2^{27.34}$) in 2017 [18]. Assuming that each user sends 500 message per each month, the number of all transactions of each month and year is estimated as $2^{36.29}$ and $2^{39.87}$, respectively.

Partial Known Plaintext Setting. Since each packet D, consisting of C, T and associated data, gives only one pair of $(T_{\mathrm{pre}}(= \mathrm{SHA}'_{256}(C)), T)$, to get 2^d pairs of $(T_{\mathrm{pre}}(= \mathrm{SHA}'_{256}(C)), T)$, 2^d packets are required.

Using the known plaintext setting and Vulnerability 5, the number of available pairs $(T_{\mathrm{pre}}(= \mathrm{SHA}'_{256}(C)), T)$ from each D can be increased. Suppose that 2^z blocks of a plaintext are known, $2^z - 1$ pairs of input/output of one-block AES-256 is obtained from a packet D, where an input of the first block is not known because IV_e is kept secret to the adversary. From Vulnerability 5, these input/output pairs of one-block AES-256 can be used in the online phase as $(T_{\mathrm{pre}}(= \mathrm{SHA}'_{256}(C)), T)$. For example, assuming each packet D gives us four pairs of $(T_{\mathrm{pre}}(= \mathrm{SHA}'_{256}(C)), T)$ on average, the number of available pairs in each month and year is estimated as $2^{38.29}$ and $2^{41.87}$, respectively.

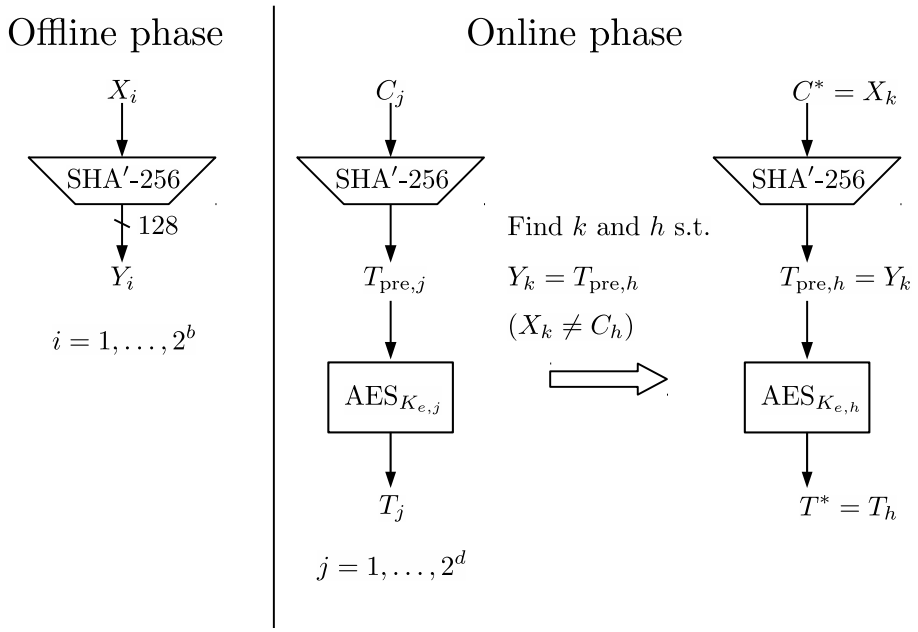

Fig. 9. Forgery Attack

6.3 Discussion

Given $2^{41.87}$ pairs of (T_{pre}, T) in the one-year online phase, it requires $2^{86.13}$ computations in the offline phase for a successful forgery attack. At the time of this writing, a bitcoin mining hardware which is capable of 8.6 TH/sec ($= 2^{39.65}$ hashing/sec) is available by only 1,000 dollars [33]. It enables about 2^{61} and $2^{64.6}$ hashing operations in a month and one year, respectively. Thus, if the adversary prepares $2^{21.53}$ hardwares, $2^{86.13}$ computations is feasible in one year. The cost is approximated as about 3 billion dollars (3,070,410,245 dollars), which seems affordable for the national organizations[3]. Also, the current Bitcoin mining cost for generation of one block is approximately 2^{72} hashing, and it is done with about 15 minutes [14], as of April 2018. Hence, $2^{86.13}$ computations is feasible within one month if the adversary has a power of whole mining computers, though the cost of network and memory is surely a barrier even with this hypothetical attack scenario.

We emphasize that our estimation is based on only *commercially available* ASIC hardware *at this time* (April 2018). If the adversary (e.g. NSA) develops a dedicated hardware, the cost can be further reduced. Also, the cost of such hardware is going to be cheaper with the development of technology year by year. Thus, we conclude that LINE-AE is not sufficiently secure as the E2EE scheme from the point of the view of a long term security.

6.4 LINE-AE as a General-Purpose Authenticated Encryption

If we take LINE-AE as a general-purpose AE, we expect it to have a sufficient level of security in terms of standard AE security notions, i.e., privacy and authenticity [11, 12]. In this respect, except the lack of authenticity of associated data (Vulnerability 3), the most significant shortage of LINE-AE is its short salt. Apparently, 64-bit salt would collide after 2^{32} encryptions, which leads to a break in the privacy notion (confidentiality of plaintext) of AE. Besides, as mentioned at Sec. 2.1 the whitepaper [29] does not specify the padding scheme needed for CBC. Hence, depending on the actual padding scheme, there might be a risk of padding-oracle attack introduced by Vaudenay [34], which exploits the weakness of the padding scheme applied to CBC. Padding-oracle attack is notoriously hard to avoid in practice, as shown by POODLE [16] or Lucky13 [10].

[3] For example, FY2012 budget for the U.S. Consolidated Cryptologic Program (which includes the NSA) was 10.5 billion [9].

Due to the structural similarity, it may make sense to compare LINE-AE with a generic composition of CBC encryption using AES-256 and HMAC-SHA-256 in terms of security. Here, we assume a random 128-bit initial vector (IV) or salt, and HMAC output is truncated to 128 bits, and CBC and HMAC are composed in the encrypt-then-mac fashion, using independent keys. Given the composition is correctly done, following Krawczyk [28] and Namprempre et. al. [30], and the standard cryptographic assumptions on AES and (the compression function of) SHA-256, the composition CBC+HMAC has 64-bit security for privacy, which comes from the provable security of CBC encryption (where 64-bit security of CBC is from the collision probability among the inputs to AES), and 128-bit for authenticity from the security of HMAC [27]. The privacy bound of LINE-AE is 32 bits, and if the salt was 128 bits, it seems not hard to derive 64-bit privacy bound, which is largely equivalent to CBC+HMAC. On the contrary, it seems less trivial to derive the authenticity bound. We expect it is possible to derive one assuming the second preimage resistance of the 128-bit SHA$'$-256 hash function. Our attack of Section 6.2 supports this observation, since it essentially breaks the second preimage resistance of SHA$'$-256 using 2^d targets.

However, we stress that our attack against LINE-AE allows treading-off of offline and online computations, and needs only single forgery attempt to have a sufficiently high success probability. At the extreme case, we can attack LINE-AE using 2^{128} offline computation with single ciphertext and single forgery attempt, which seems not possible with CBC+HMAC using 256-bit keys.

7 Conclusion

In this paper, we have evaluated the security of the E2EE scheme of LINE, one of the popular messaging applications in East Asia, and proposed several practical attacks. We first showed impersonation and forgery attacks on the group messaging scheme by a malicious group member. Next, we presented the malicious key exchange attack on the one-to-one messaging scheme. Then, we evaluated the security of the authenticated encryption scheme used in the message encryption phase, and presented the forgery attack against the the authenticated encryption scheme by the E2E adversary. We discussed practicality and feasibility of our attacks by considering the use cases of LINE. As a result, we conclude that the E2EE scheme of LINE do not provide a sufficient level of security compared to the start-of-the-art E2EE schemes such as Signal, which is used by WhatApp and Facebook Messenger, and Apple's iMessage.

Acknowledgments The authors would like to thank the anonymous referees for their insightful comments and suggestions. We are also grateful to LINE corporation for the fruitful discussion and feedback about our findings.

References

1. FIPS PUB 197, Advanced Encryption Standard (AES), 2001. U.S.Department of Commerce/National Institute of Standards and Technology.
2. NIST SP 800-38A, Recommendation for Block Cipher Modes of Operation, 2001. U.S.Department of Commerce/National Institute of Standards and Technology.
3. NIST SP 800-38C, Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality, 2007. U.S.Department of Commerce/National Institute of Standards and Technology.
4. NIST SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, 2007. U.S.Department of Commerce/National Institute of Standards and Technology.
5. FIPS PUB 180-4, Secure Hash Standard, 2015. U.S.Department of Commerce/National Institute of Standards and Technology.
6. New generation of safe messaging: Letter Sealing. LINE Blog,, 2015. `https://engineering.linecorp.com/en/blog/detail/65`.
7. LINE Enters Agreement with Japan's CAO for Mynaportal Interconnectivity, 2017. `https://linecorp.com/en/pr/news/en/2017/1771`.
8. Line Will Top 50 Million Users in Japan This Year. eMarketer, 2017. `https://www.emarketer.com/Article/Line-Will-Top-50-Million-Users-Japan-This-Year/1016207`.

9. David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella Béguelin, and Paul Zimmermann. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 5–17. ACM, 2015.
10. Nadhem J. AlFardan and Kenneth G. Paterson. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 526–540. IEEE Computer Society, 2013.
11. Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptology*, 21(4):469–491, 2008.
12. Mihir Bellare, Phillip Rogaway, and David A. Wagner. The EAX Mode of Operation. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004.
13. Daniel J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.
14. BitcoinWisdom.com. Bitcoin Difficulty, 2017. `https://bitcoinwisdom.com/bitcoin/difficulty`.
15. Simon Blake-Wilson and Alfred Menezes. Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 1999.
16. Bodo Möller and Thai Duong and Krzysztof Kotowicz. This POODLE Bites: Exploiting The SSL 3.0 Fallback, 2016.
17. David Bogado and Danny O'Brien. Punished for a Paradox., Mar. 2, 2016. `https://www.eff.org/deeplinks/2016/03/punished-forparadox-brazils-facebook`.
18. Andy Boxall. Line's monthly active user count drops, but Q2 2017 revenue and profit climbs, 2017. `http://www.businessofapps.com/lines-monthly-active-user-count-drops-but-q2-2017-revenue-and-profit-climbs/`.
19. Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. Cryptology ePrint Archive, Report 2017/666, 2017. `http://eprint.iacr.org/2017/666`.
20. Katriel Cohn-Gordon, Cas J. F. Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, pages 451–466. IEEE, 2017.
21. Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. On post-compromise security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 164–178. IEEE Computer Society, 2016.
22. Tim Cook. A Message to Our Customers, 2016. `https://www.apple.com/customer-letter/`.
23. Tyler Curtiss. Encryption out of LINE Reverse engineering end-to-end encrypted messaging. Ekoparty 2016, 2016.
24. Antonio M. Espinoza, William J. Tolley, Jedidiah R. Crandall, Masashi Crete-Nishihata, and Andrew Hilts. Alice and bob, who the FOCI are they?: Analysis of end-to-end encryption in the LINE messaging application. In *7th USENIX Workshop on Free and Open Communications on the Internet (FOCI 17)*, Vancouver, BC, 2017. USENIX Association.
25. Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. Key confirmation in key exchange: A formal treatment and implications for TLS 1.3. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 452–469. IEEE Computer Society, 2016.
26. Christina Garman, Matthew Green, Gabriel Kaptchuk, Ian Miers, and Michael Rushanan. Dancing on the lip of the volcano: Chosen ciphertext attacks on apple imessage. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 655–672, Austin, TX, 2016. USENIX Association.
27. Peter Gazi, Krzysztof Pietrzak, and Michal Rybár. The Exact PRF-Security of NMAC and HMAC. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 113–130. Springer, 2014.
28. Hugo Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.

29. LINE Corporation. LINE Encryption Overview, 2016.
30. Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering Generic Composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, 2014.
31. Open Whisper Systems. Signal Github Repository, 2017. `https://github.com/WhisperSystems/`.
32. Paul Rosler, Christian Mainka, and Jorg Schwenk. More is Less: How Group Chats Weaken the Security of Instant Messengers Signal, WhatsApp, and Threema. 3rd IEEE European Symposium on Security and Privacy 2018, 2018.
33. Jordan Tuwiner. Bitmain Antminer R4 Review, 2017. `https://www.buybitcoinworldwide.com/mining/hardware/antminer-r4/`.
34. Serge Vaudenay. Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ... In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer, 2002.