Alice and Bob, who the FOCI are they?: Analysis of end-to-end encryption in the LINE messaging application

Antonio M. Espinoza, William J. Tolley, and Jedidiah R. Crandall
Dept. of Computer Science
University of New Mexico
Corresponding author: amajest@cs.unm.edu

Andrew Hilts and Masashi Crete-Nishihata
Citizen Lab, Munk School of Global Affairs
University of Toronto

Abstract

End-to-end encryption (E2EE) is becoming a standard feature in many popular chat apps, but independent security assessments of these implementations are limited. In this paper we provide the first independent analysis of E2EE features in LINE, a messaging application popular in Asian markets, and identify a replay attack and an attack on a lack of forward secrecy. Based on our analysis and communications with LINE about the vulnerabilities we discuss challenges and new research directions to better bridge vendors, researchers, and end-users around security issues.

1 Introduction

Security and privacy features (*e.g.,* default HTTPS and multi-factor authentication) are becoming increasingly standardized in popular consumer applications. This shift is particularly apparent with the adoption of End-to-End Encryption (E2EE) in chat applications. Beginning in late 2015, popular chat applications (*e.g.,* WhatsApp, Facebook Messenger, Viber, LINE, and KakaoTalk) started to introduce E2EE features. While this trend is encouraging and some applications (*e.g.,* WhatsApp) have adopted well documented and reviewed encryption protocols (*e.g.,* Signal Protocol) there is a general lack of independent security research assessing the implementation of E2EE for many of these apps.

In this paper we provide the first independent security analysis of E2EE features in LINE, a messaging application popular in Asian markets that has a user base of over 200 million monthly active users. Our analysis reveals a replay attack and an attack on a lack of forward secrecy. Based on our analysis and experience disclosing these vulnerabilities to LINE we argue that new research directions are needed to better bridge vendors, researchers, and end-users around security issues.

The attacks we describe are within the capabilities of a well resourced attacker (such as a state actor), but the countries where users are the most at risk of surveillance of their LINE communications may be unlikely to carry out such attacks. For example, if a government is unlikely to be able to coerce LINE into colluding with them or breach LINE's infrastructure, then are attacks that require the LINE private key moot? If a replay attack requires physical access to a phone and users tend not to ever delete messages, is the state actor more likely to just look through the message history than to carry out a sophisticated cryptographic attack? Finally, how do we find a good compromise between "worst-case" scenarios and scenarios that are generalizable to the greatest number of users?

These questions underline differences between how researchers and vendors may evaluate threats. Vendors attempting to implement security at scale for millions of users face hard decisions over balancing security, usability, and resources. Researchers finding and reporting vulnerabilities may be well versed in security best practices, but are unlikely to have visibility into the decision making processes that led to security designs and implementations. Meanwhile the average end-user is increasingly presented with security features, but is unlikely to have enough knowledge to assess the merits of one implementation over the other. We proceed by detailing our analysis of E2EE in LINE and conclude with discussion of how to better bridge researchers, vendors, and end-users.

2 Background

This section provides an overview of trends in encrypted messaging and communications security in LINE.

2.1 Encrypted Messaging

Security researchers were concerned about message security long before popular chat applications began to add security features to their clients. OTR [28] was proposed in 2004 as an alternative to PGP [45] since OTR offers users forward secrecy (discussed more in Section 3.5) and is designed for use in conjunction with messaging protocols such as XMPP. The introduction of E2EE in

many popular chat applications came two years after the 2013 Snowden revelations invigorated public debate over digital communications interception. The invention of the double ratchet [15] solved the problem of devices being able to decrypt messages received while offline and is now widely adopted, but early E2EE implementations such as that of LINE use other solutions. See [44] for a more complete background of secure messaging.

2.2 LINE Overview

LINE was released in 2011 by LINE Corporation, a Japanese subsidiary of South Korea's Naver Corporation. Since its release, LINE experienced rapid growth in Japan, and later in other Asian markets (*e.g.,* Thailand, Taiwan, *etc.*). In 2016, LINE reported over 200 million monthly active users (MAU) [40].

Amid this growth, LINE has come under government pressure to implement content controls and provide access to user communications. Previous work has documented client-side keyword filtering enabled for users based in China to comply with Chinese content regulations [34]. In 2013, the government of Thailand claimed it planned to monitor LINE communications. LINE responded by stating access to user information would only be permitted with a Japanese court order and that no official request had been filed by Thai authorities [2]. Recent LINE policy documents [25] state it responds to non-Japanese requests for user data through the mutual legal assistance treaty (MLAT) process.

In addition to these pressures LINE has made a number of changes to how it encrypts traffic. Version 3.9.2 and earlier LINE releases only encrypted client-server communications over WIFI and not 3G [2]. There was speculation that the lack of encryption on 3G may have been intentional to make it easier to comply with lawful interception requests [9]. LINE explained in a blog post that when it introduced the SPDY protocol into its platform it decided to allow for non-encrypted connections over mobile networks to avoid slow connection and transfer times [1]. In version 3.9.3 (released in October 2013) LINE introduced encryption over both WIFI and 3G [2].

In 2014, LINE announced [20] its "Hidden Chat" feature, a special type of conversation that users could start in which messages were "sent in a secure state." This feature was implemented at a time when other messaging apps such as KakaoTalk [21] and Telegram [17] implemented or announced improvements to "hidden chat" or other non-default encrypted communication features. In 2015, LINE announced its "Letter Sealing (End-to-end Encryption)" feature [11], and in 2016 this became a default feature [22], which sought to make the "sense of security" that people had using Hidden Chats the default for all messages. Two months later, LINE updated the Letter Sealing feature [23] to include a lock in the UI that let users know that messages were being "stored on LINE's servers in an encrypted state." LINE then published a "Technical Whitepaper" [24] describing Letter Sealing's cryptographic implementation in detail, which we refer to in subsequent sections of this report.

Note that LINE added message security features throughout 2014, 2015, and 2016 [20, 22], whereas the double ratchet algorithm [15] was not published until late in 2016. This timeline is one reason why LINE does not use a double ratchet, but there are also usability concerns discussed in Section 5.

3 LINE Technical Analysis

This section describes our threat model, attack implementation, and proof-of-concept exploits. The motivation for our analysis is to compare the implementation described in LINE's "Technical Whitepaper" with that of the version of LINE released shortly after the whitepaper's publication.

3.1 Threat Model

For our attacks we assume the client is running LINE version 6.7.1, as that is the first version released that should have been congruent with the whitepaper released by LINE. We assume the same threat model as outlined by LINE in their Letter Sealing blog post. In the post they state, in regards to their old message encryption protocol, "While only hypothetical, there is one flaw with this method. A hacker inside the LINE servers could still be able to compromise the safety of message data." Therefore, we assume in our threat model the attacker is the server, or an attacker with the server's private key and perspective of network information. We believe this to be a reasonable assumption as it is presented as LINE's reasoning for implementing "Letter Sealing."

For the forward secrecy attack we assume the above threat model and require one of the clients' private keys to be compromised. We feel this assumption is not unreasonable since state actors could coerce individuals into unlocking their phones, which would give an attacker the ability to obtain the necessary keys. The attacker could be a nation state or entity with the ability to persuade LINE into keeping detailed message logs. For the replay attack, there is no need for either clients' private key or device to be compromised. Rather, it is possible for the LINE server to replay messages by simply sending the ciphertext again. This property is not one that an an end-to-end encryption system that follows cryptography best practices would have.

3.2 What is a Replay Attack?

A replay attack is an attack where an adversary records messages between two parties and can later replay any of

those messages to either party member as though it was sent legitimately. The attacker does not need to know what the message decrypts to in order to send it. Our replay attack does not send messages, but replaces the body of a message in transit with any message seen before. We chose to implement our attack this way as it does not require us to completely learn LINE's protocols for sending and receiving messages. See Figure 1 for screen shots of our attack's effects on both message sender and recipient. Notice that the last message seen in Figure 1a is not the same as the last message seen in Figure 1b. Instead the message has been replaced with a previous message sent from "deepthroat" to "woodward". This attack is possible due to a problem in the end-to-end encryption protocol, specifically the way deepthroat's messages are authenticated by woodward.

3.3 What is a MAC?

Message Authentication Codes (MACs) are used to guarantee the integrity of a message, ensuring the message has not been altered in any way. When a message is received the receiver calculates the MAC and checks it against the MAC the sender calculated to ensure the message received is the same as the original message sent. A good MAC has a key separate from the key used to encrypt a message. In addition, a MAC should also authenticate additional data, such as source and destination information and a message number. This additional data protects the message from replay attacks.

3.4 Replay Attack Implementation

Our replay attack is possible due to the fact that LINE only authenticates the message itself, leaving open the possibility of replay attacks. There is also a deviation from cryptography best practices in that LINE uses the same key for the MAC as for encryption (our replay attack does not exploit this weakness). Both this attack and our attack on forward secrecy assumes the attacker has the same privileges as the LINE server (Appendix A describes our implementation).

Through reverse engineering we were able to discern which parts of the packet contain various components of the end-to-end communication. The three important sections are: the salt, the encrypted message, and the MAC. When the server sends a new message with these three fields the same as an old recorded message,the message is replayed. We demonstrated the attack using an implementation that is described in Appendix B.

3.5 Forward Secrecy Implementation

Forward secrecy is a property of an encryption system that removes an attacker's ability to decrypt past messages, even if one or more users' private keys are compromised [44]. For client-to-client communication, forward secrecy is implemented by generating a new key for each session or message exchanged between users (called an ephemeral key). The most important consideration is that the key for this session is generated in some way that is not predictable or deterministic, and that the duration of this layer of security is "end-to-end," meaning that it is encrypted from the user sending the message to the intended recipient.

LINE, however, only offers forward secrecy from client-to-server, meaning that the layer of security offered by having ephemeral keys does not protect the user from a malicious actor with the same privileges as the LINE server. The company could fall under under governmental duress that forces it to save the communications between two individuals and decrypt them. This attack becomes possible if the attacker can gain access to the secret key of just one of the users' devices. This method of attack requires physical confiscation of a device, by law enforcement, government agents, employers, *etc.*, but in any case an attacker with one private key of one user would be able to recover messages, even if they have been deleted from both users' devices.

3.6 Attack on Lack of Forward Secrecy

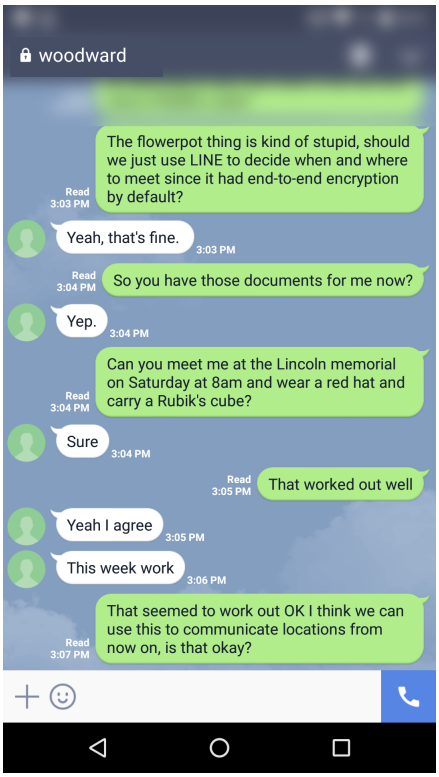
To illustrate the vulnerability posed by LINE deciding not to include forward secrecy from client-to-client, we collected messages as the server would see them using the same setup as for the replay attack. This allows us to view the messages with the first layer of encryption removed. If the device is then compromised—by an adversary confiscating it, for example—they would need only to retrieve the shared secret (by obtaining one of the users' private keys) that is used between users and the salt from the message to derive the initialization vector and key needed to decrypt the message. We demonstrated this by using our own private key from one device to decrypt a message from the E2EE ciphertext that the server sees.

4 Responsible Disclosure

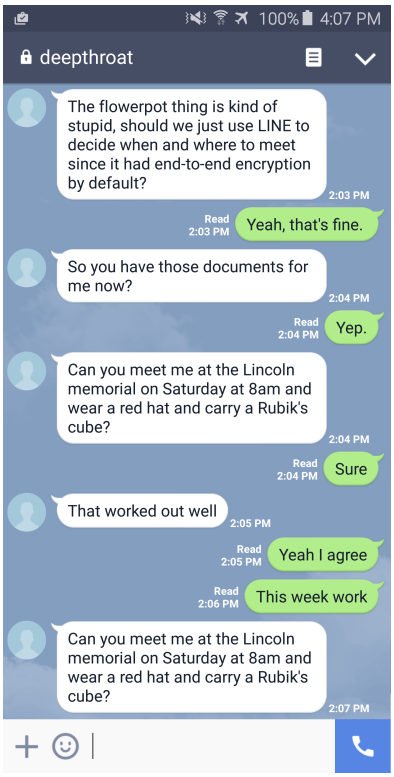
We first disclosed our findings concerning forward secrecy to LINE on December 20, 2016. Shortly after their reply to our disclosure, we discovered the replay attack, which we disclosed to LINE on January 13, 2017. On January 27, 2017, LINE replied to our second disclosure. See Table 1 for a detailed timeline of the disclosure process.

In reply to our first disclosure LINE agreed that forward secrecy would improve the security of their E2EE implementation, but explained they had decided not to include it in the first release of the feature. They acknowledged our threat model, but believed their model addressed more immediately practical concerns:

> "While FS [Forward Secrecy] for messag-



(a) What deepthroat sees



(b) What woodward sees

Figure 1: **Both sides of the conversation.**

Table 1: **Disclosure Timeline.**

Date	Contact
12/20/2016	Forward secrecy disclosure.
12/21/2016	Reply to initial disclosure concerning forward secrecy.
1/13/2017	Replay attack disclosure and inquiry about inconsistencies.
1/27/2017	Reply to replay attack and inconsistency questions.

> ing could be added in a future version, currently FS is only available at the transport level. We believe this covers the more realistic case where LINE server keys are leaked, stolen, or confiscated by authorities. As for your threat model, if a device is confiscated, whoever has the device will be able to read stored messages, even without going through the trouble of extracting keys."

We replied with the following:

> "Regarding the practicality of the attack we proposed, against an attacker with the privileges of the LINE server the True Delete feature described here [16] is ineffective. Because a user has no guarantee that the LINE server is not recording ciphertexts, this means that deleting messages on their device, even if both the sender and receiver delete their copy of the message, does not preclude the possibility of plaintext being recovered in the future. LINE could collude with law enforcement to confiscate a phone and recover all deleted past messages by extracting the E2EE private key of the user."

LINE responded that the True Delete feature is only intended against an attacker with physical access to the device, and is not intended to provide users with any guarantee about copies of messages on servers being deleted, which LINE argued would be moot anyway because an attacker who could compromise transport security could make copies of the messages.

In this second round of communication, LINE also provided a detailed response to our questions conveying their design decisions with respect to forward secrecy. They explained that forward secrecy was left out of the first release due to complications with synchronizing be-

tween mobile and secondary devices (primarily desktop clients). They explained they would "strengthen the MAC calculation" in the next Letter Sealing version, and work to address the replay attack issues, but that the secondary applications synchronization issue, in addition to being a challenge for implementing forward secrecy end-to-end, also presents some challenges for making other algorithm changes.

In the second round of communication we also pointed out some inconsistencies between LINE's whitepaper [24] and the implementation, which are enumerated in Appendix C. LINE responded that these issues had been fixed but the fixes were disabled in the version we reverse engineered because of a bug.

5 Discussion

While consumer applications have been steadily announcing new features to protect the confidentiality of user communications, there exists a divide between how different stakeholder groups conceptualize and communicate their threat models to one another [41]. Building from our case study we discuss gaps in mutual understanding and communications between researchers, companies, and end-users and ways to address them.

5.1 Bridging Gaps Between Researchers and Vendors

Security researchers seek to probe popular and emerging systems for novel vulnerabilities that can contribute to the research literature and general security understanding. In some cases, exploitation scenarios for novel vulnerabilities can be difficult to communicate to other stakeholder groups. Our case study provides an example of how such difficulties can impact the responsible disclosure process. While we sought to communicate to the LINE security team the applicability of our replay attacks and the lack of client-to-client forward secrecy despite their stated threat model, there was a disconnect in terms of how serious the threats were viewed to be. A common theme was that the complexity of the interaction between our reported vulnerabilities and the broader design decisions made by LINE allowed both sides of the communication to reframe discussions at will.

When we reported vulnerabilities in their cryptographic protocol that assumed what we understood to be their own threat model [23, 11], part of their response was to point out that an attacker could just confiscate a user's phone and see past messages that way. As pointed out in Section 4, another way to view our attacks on the lack of forward secrecy (and the ability to replay messages from the server, for that matter) is to view them as violations of LINE's True Delete feature [16], which zeroes out messages on a user's device upon deletion so that common forensics techniques for persistent storage cannot be used to retrieve deleted messages. Even if both sides of a conversation delete a message from that conversation, it can be replayed or recovered using our attacks.

In summary, our disclosure process with LINE began with us discussing forward secrecy as a transport security issue, LINE replied that physical device confiscation was a more realistic threat, we recast forward secrecy as a forensics issue, and LINE cited the lack of perfect solutions for transport security as a fundamental limitation on anti-forensics techniques. Both parties were arguing in good faith with the same goal (making the application more secure), but cryptography and the real-world threats that give it context are so complexly intertwined that technical conversations about threats tend to shift in topic very easily.

To better understand forward secrecy as an anti-forensics technique, imagine the following hypothetical situation. Alice the reporter has a source for a story code named Bob. The story is about the government in Country X. Using LINE for messaging, Alice and Bob have a private conversation about the story. Bob uses a "burner" phone to communicate with Alice, meaning that after their communications, but before Alice's story is published, he physically destroys his mobile phone and all the data on it. After Alice publishes the story, she plans to travel to Country X. Knowing that LINE has end-to-end encryption and the True Delete feature, she decides that she should delete all of the messages between her and Bob, but keeps LINE installed on her phone to preserve other conversations and contacts in the LINE app.

Alice is detained at the border of Country X and has her phone confiscated by authorities. Several hours later during an interrogation with authorities she is presented with a plaintext decryption of her entire conversation with Bob. Remember that Bob's phone was destroyed, Alice's phone had no copies of the messages on it, and they always used LINE's end-to-end encryption feature. So how did authorities obtain the plaintext conversation?

The answer is that LINE's end-to-end encryption does not have forward secrecy for end-to-end client communications. Therefore, when the government confiscated Alice's phone they were able to extract her private key and use it to decrypt all of the encrypted messages that they had recorded. Realistically, however, many governments prefer less technically sophisticated attacks and many users do not bother to delete old messages, so keeping threat models grounded in likelihood is another challenge for communication between researchers and vendors.

Part of the general disconnect between researchers and vendors may be due to a lack of common understanding. LINE engaged with us in good faith and have a bug bounty program [38] to encourage independent security

research of its platform. However, like other major software companies it has hundreds of millions of users and likely has to prioritize many different security issues and consider their impact on usability, uptime, and other variables. Researchers analyzing the platform from outside these processes may not fully appreciate the competing priorities. An example of this challenge is the technical problem of secondary devices for chat apps with E2EE features. In WhatsApp, for example, a master device has to be online for secondary devices (*e.g.*, their web app) to encrypt and decrypt messages. We tested this implementation on WhatsApp version 2.17.31. LINE did not want to place a similar restriction on their users which prompted them to implement their forward secrecy client-to-server, rather than client-to-client.

While ample literature exists on end user understanding and implementation of security advice [26, 35, 31, 42], there is little comparable research examining how security teams at software vendors understand and act upon vulnerability reports. Existing work largely focuses on the process of and timing of disclosure, patching and publication, and not on the effectiveness of communications (*e.g.*, [29, 27]). A study focused on the substance of and reaction to vulnerability disclosure communications could help both security researchers better communicate their results and vendors better appreciate the severity of issues.

5.2 Better Communicating Research

Our community (both academic and the privacy community at large) discovers attacks against E2EE [33, 14, 37, 36, 5, 18], reverse engineers apps such as LINE for a variety of purposes [7, 8, 12, 3, 10], performs formal security analyses of protocols [32, 30], and compares E2EE implementations [19, 4]. Even within our community, threat models vary from showing a lack of semantic security without demonstrating any practical attacks to very real attacks on widely used apps [33]. There are few actual attacks by state actors and contractors to use as case studies, and very little information about those that get reported [13, 6]. This wide variety of threat models with no common understanding about what presents real risks makes it difficult to communicate with vendors and users.

5.3 Better Educating End-users

Vendors, the media, and in some cases, security trainers, all communicate with end users about how application privacy and security features work and how to stay safe online. Despite these efforts studies have shown that end users have difficulty understanding the basic premises of how end-to-end encryption works [26] and have different mental models for how to stay safe online than experts [35]. Other research shows that even users who demonstrate higher levels of security literacy may not be any more secure when looking at the actual security of their devices [31]. Further work has demonstrated a "digital security divide," whereby users with lower socioeconomic status or education levels rely on lower quality security information [42].

Media organizations can write sensational stories about security vulnerability reports (*e.g.*, [39]), spreading uncertainty to users who already have difficulty adopting secure practices. While sensational media stories are certainly not limited to security vulnerabilities, given the already low level of security literacy among the general public, researchers have the responsibility to educate and inform the media, who in turn have the responsibility to provide balanced and accurate information to the public. Research examining the communication between security researchers and journalists and reporting of security issues could help encourage more thoughtful and accurate media stories. Such an investigation could provide recommendations that could help ensure reporting on security vulnerabilities do not needlessly spread fear, uncertainty, and doubt.

6 Conclusion

We found that version 6.7.1 of LINE was vulnerable to a replay attack and an attack on the lack of end-to-end forward secrecy between clients. These attacks assumed the same threat model described in LINE's security documentation. Based on our analysis and communications with LINE we identified an open question for vendors and security researchers: How do we find a good compromise between "worst-case" scenarios and scenarios that are generalizable to the greatest number of users? We proposed future avenues of research around how to bridge vendors, researchers, and end-users to help address this question.

The motivation for our paper's title was a security trainer who works with at-risk populations, and made the point that users care more about actual safety and security concerns than about the concepts that researchers work with. As pointed out by Rogaway [43], we should "Regard ordinary people as those whose needs [we] ultimately aim to satisfy." The challenge is how to do so while avoiding unintended consequences and with deference to the perspectives of other stakeholders. Users who decide not to use a specific messaging application because of security concerns discovered by researchers, for example, might simply fall back to SMS messaging, which has no end-to-end security at all. Practical concerns, such as secondary devices, are rarely considered in academic work, forcing vendors to create their own solutions. Our aim with this paper is to start a discussion about what role the research community should play in addressing these types of issues.

Acknowledgments

This material is based upon work supported by the U.S. National Science Foundation under Grant Nos. #1314297, #1420716, #1518523, and #1518878. Antonio Espinoza was supported by the Open Technology Fund Information Controls Fellowship Program. This research is part of the Net Alert (https://netalert.me) project funded by the Open Technology Fund. We would like to thank Jeffrey Knockel for commenting on drafts, providing code for modifying packets in flight, and implementing LEGY HMAC in python source code. We also thank Ramy Raoof for inspiring the title of our paper, and the LINE security team for comments on drafts and their engagement during the disclosure process. We are also grateful to the anonymous FOCI reviewers and our shepherd, Nick Weaver, for valuable feedback.

References

[1] Adopting SPDY in LINE. part 1: An overview. http://tech.naver.jp/blog/?p=2381.

[2] Asia chats: LINE corporation responds. https://citizenlab.org/2013/12/asia-chats-line-responds/.

[3] Codegate 2014. https://www.alexrad.me/discourse/codegate-2014.html.

[4] How private are your favourite messaging apps? https://www.amnesty.org/en/latest/campaigns/2016/10/which-messaging-apps-best-protect-your-privacy/.

[5] How service providers of messengers with end-to-end encryption could be compelled to decrypt messages. https://plus.google.com/+RolfWeber/posts/SYw6AD8xkK7.

[6] Iranian Hackers Just Cracked This Super Secure Instant Messaging Service. Fortune, http://fortune.com/2016/08/02/telegram-hackers-iran/.

[7] LINE instant messenger protocol documentation. http://altrepo.eu/git/line-protocol.

[8] LINE protocol analysis. https://hexa-unist.github.io/2013/09/05/LINE-protocol-analysis/.

[9] LINE vulnerable to man-in-the-middle attack. https://www.telecomasia.net/blog/content/line-vulnerable-man-middle-attack.

[10] Multiple vulnerabilities in LINE instant messenger platform. Full Disclosure, http://seclists.org/fulldisclosure/2016/Aug/45.

[11] New generation of safe messaging: "Letter Sealing". LINE Blog, https://engineering.linecorp.com/en/blog/detail/65.

[12] python wrapper for LOCOP protocol. http://carpedm20.blogspot.kr/2013/08/python-wrapper-for-loco-protocol.html.

[13] Spy Tech 'Hacks WhatsApp Encrypted Chat From A Backpack'. Forbes, https://www.forbes.com/sites/thomasbrewster/2016/09/29/wintego-whatsapp-encryption-surveillance-exploits/#5b6aaccb1aa9.

[14] Telegram, AKA "Stand back, we have Math PhDs!". https://unhandledexpression.com/2013/12/17/telegram-stand-back-we-know-maths/.

[15] The Double Ratchet Algorithm . https://whispersystems.org/docs/specifications/doubleratchet/.

[16] True Delete. LINE Engineering Blog, https://engineering.linecorp.com/en/blog/detail/64.

[17] Usernames and secret chats 2.0. https://telegram.org/blog/usernames-and-secret-chats-v2.

[18] WhatsApp Forensic Artifacts: Chats Aren't Being Deleted. https://www.zdziarski.com/blog/?p=6143.

[19] Wire Privacy. https://wire.com/en/privacy/#table-competition.

[20] New "hidden chat" feature released, enables sending of time-limited messages : LINE official blog. http://official-blog.line.me/en/archives/1006361166.html, July 2014.

[21] Secret chat and decline invites now available on kakaotalk | kakao blog. https://blog.kakaocorp.com/?p=943, December 2014.

[22] Hidden chat users to enjoy "Letter Sealing" from July! : LINE official blog. http://official-blog.line.me/en/archives/1058913293.html, June 2016.

[23] Letter sealing gets enhanced! http://official-blog.line.me/en/archives/1060089042.html, August 2016.

[24] LINE encryption overview: Technical whitepaper. https://scdn.line-apps.com/stf/linecorp/en/csr/line-encryption-whitepaper-ver1.0.pdf, September 2016.

[25] LINE transparency report (july-december 2016). https://linecorp.com/en/security/tr_report_2016_2, April 2017.

[26] ABU-SALMA, R., SASSE, M. A., BONNEAU, J., DANILOVA, A., NAIAKSHINA, A., AND SMITH,

M. Obstacles to the adoption of secure communication tools. In *Security and Privacy (SP), 2017 IEEE Symposium on (SP'17). IEEE Computer Society* (2017).

[27] ARORA, A., KRISHNAN, R., TELANG, R., AND YANG, Y. An empirical analysis of software vendors' patching behavior: Impact of vulnerability disclosure. *ICIS 2006 Proceedings* (2006), 22.

[28] BORISOV, N., GOLDBERG, I., AND BREWER, E. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society* (2004), ACM, pp. 77–84.

[29] CAVUSOGLU, H., AND RAGHUNATHAN, S. Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge. *IEEE Transactions on Software Engineering 33*, 3 (2007).

[30] COHN-GORDON, K., CREMERS, C., DOWLING, B., GARRATT, L., AND STEBILA, D. A formal security analysis of the Signal messaging protocol. Cryptology ePrint Archive, Report 2016/1013, 2016. http://eprint.iacr.org/2016/1013.

[31] FORGET, A., PEARMAN, S., THOMAS, J., ACQUISTI, A., CHRISTIN, N., CRANOR, L. F., EGELMAN, S., HARBACH, M., AND TELANG, R. Do or do not, there is no try: user engagement may not improve security outcomes. In *Symposium on Usable Privacy and Security (SOUPS)* (2016).

[32] FROSCH, T., MAINKA, C., BADER, C., BERGSMA, F., SCHWENK, J., AND HOLZ, T. How secure is TextSecure? Cryptology ePrint Archive, Report 2014/904, 2014. http://eprint.iacr.org/2014/904.

[33] GARMAN, C., GREEN, M., KAPTCHUK, G., MIERS, I., AND RUSHANAN, M. Dancing on the lip of the volcano: Chosen ciphertext attacks on Apple iMessage. In *25th USENIX Security Symposium (USENIX Security 16)* (Austin, TX, 2016), USENIX Association, pp. 655–672.

[34] HARDY, S. Asia chats: Investigating regionally-based keyword censorship in LINE. Tech. Rep. Research Brief No. 25, Citizen Lab, November 2013.

[35] ION, I., REEDER, R., AND CONSOLVO, S. "... no one can hack my mind": Comparing expert and non-expert security practices. In *Proc. SOUPS* (2015).

[36] JAKOBSEN, J., AND ORLANDI, C. On the CCA (in)security of MTProto. Cryptology ePrint Archive, Report 2015/1177, 2015. http://eprint.iacr.org/2015/1177.

[37] JUNG-LODDENKEMPER, A. A 2^{64} attack on telegram, and why a super villain doesn't need it to read your telegram chats. https://www.alexrad.me/discourse/a-264-attack-on-telegram-and-why-a-super-villain-doesnt-need-it-to-read-your-telegram-chats.html.

[38] LINE. Line security bug bounty program, 2017. https://bugbounty.linecorp.com/en/.

[39] LOMAS, N. Security researchers call for Guardian to retract false WhatsApp "backdoor" story. Tech Crunch.

[40] MILLWARD, S. Line is getting dangerously dependent on users in its 4 top countries. https://www.techinasia.com/line-q1-2016-dependent-four-countries, April 2016. (Accessed on 05/26/2017).

[41] RADER, E., AND WASH, R. Identifying patterns in informal sources of security information. *Journal of Cybersecurity 1*, 1 (2015), 121.

[42] REDMILES, E. M., KROSS, S., AND MAZUREK, M. L. How I learned to be secure: A census-representative survey of security advice sources and behavior. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2016), CCS '16, ACM, pp. 666–677.

[43] ROGAWAY, P. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. http://eprint.iacr.org/2015/1162.

[44] UNGER, N., DECHAND, S., BONNEAU, J., FAHL, S., PERL, H., GOLDBERG, I., AND SMITH, M. Sok: Secure messaging. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2015), SP '15, IEEE Computer Society, pp. 232–249.

[45] ZIMMERMANN, P. R. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, USA, 1995.

A Attack Environment

Our attacks assume the attacker can see what the LINE server sees. In order to accomplish this we used the network setup shown in Figure 2. In this setup all traffic to and from *Client B* must pass though the man-in-the-middle (MITM) machine. We give the MITM machine

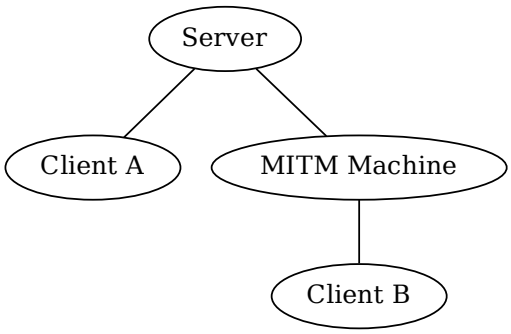


Figure 2: **Attack network setup. Note that our setup relies on *Client B* colluding with the MITM machine, this allows us to simulate our server having the same privileges as LINE's server.**

the AES key used by *Client B* and the server to communicate with each other. Each message from the server to *Client B* is encrypted with this key and a static initialization vector that is hard-coded into the LINE app. With these two values our MITM machine can decrypt and re-encrypt any client-to-server LINE message (note the E2EE portion of the message is still encrypted). This allowed us to simulate performing the attack from the vantage point of the LINE server.

B Replay Attack Details

In order to demonstrate the replay attack, we waited for a message of similar size to be sent from one client to another and replaced the three critical sections: the salt, the encrypted message, and the MAC. We can replace these values in any message we see in transit on the wire. This allowed us to carry out the attack without needing to completely reverse engineer the client-to-server protocol. Once the message is replaced, the LEGY HMAC is recalculated and replaced. This simple attack is outlined in Figure 3. Variable sized messages with the attack are possible, as are attacks not requiring a new message to be sent, but we did not implement these features in our attack because we only sought to demonstrate that replay is possible. Although in Figure 1 the replayed message happens only minutes later in our example attack, the same message played a week later would have a totally different context.

Note that the packet is further encrypted with an AES key known only to the client and server, and so we used MITM to remove that layer because our threat model views the server as the attacker. Note also that a more sophisticated version of the attack could replay messages of arbitrary length, or not even wait for new messages to replay old messages, but our proof-of-concept exploit was kept simple because it was only for demonstration.

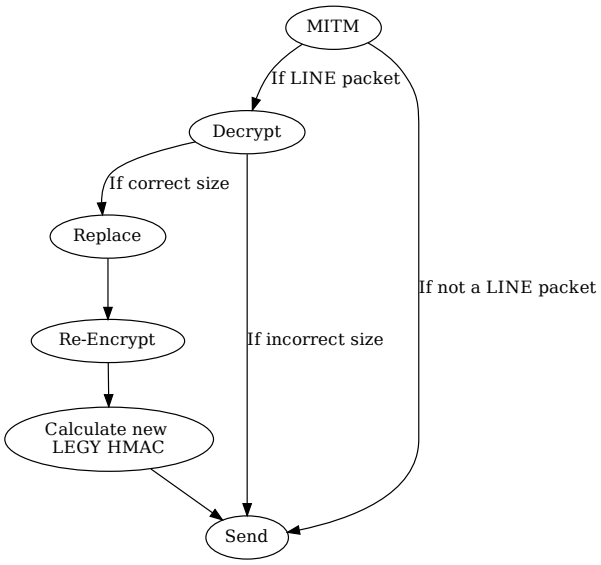


Figure 3: **Attack flow graph.**

C Other Issues

We also documented several places where LINE's implementation deviated from their whitepaper description and/or common best practices for implementing crytography. These include:

- The whitepaper says that the Client-to-Server Transport Encryption protocol uses an ephemeral Initialization Vector (IV) along with the ephemeral encryption key for AES, but we found that the IV is hard-coded and never changes.
- The whitepaper says that the Client-to-Server Transport Encryption protocol uses AES-GCM, but we found that it uses AES in CBC mode.
- LEGY-HMAC, the MAC used for client-to-server communications (not documented in the whitepaper) has only a 32-bit digest and is based on a hash algorithm that is not cryptographically strong.
- For E2EE encryption, the same key is used for encryption as well as for the Message Authentication Code (MAC). It is considered a common best practice to use separate keys, to preclude the possibility of chosen plaintext attacks leading to message forgery.
- For E2EE encryption, the MAC is a simple "hash and encrypt," compared to something like HMAC that precludes length extension attacks.

As pointed out in Section 4, LINE stated that these issues had been fixed but the fixes were disabled in the version we reverse engineered because of a bug.