Security Analysis of the Encrypted Mobile Communication Applications

Murat Demircioğlu, Halil Kemal Taşkın and Salim Sarımurat

***Abstract*—Communication over the Internet have moved from desktop computers to handheld devices over the course of the last decade. People have started using cell phones, tablets etc. in daily lives and communication methods have switched to using these devices. Although there are many different options available for instant messaging between modern time internet users, not many of them provide trusted methods for securing the communicated message and considering users' privacy.**

In this paper, we have analyzed the security and privacy concerns of three well-known and widely-used mobile communication applications called Telegram, TextSecure and Threema. We have defined the general security requirements expected from such applications and then compared their approaches individually.

***Index Terms*—Mobile Messaging, End-to-end Security, Telegram, TextSecure, Threema,**

I. Introduction

INSTANT messaging is a kind of online chat offering a real-time text transmission over the Internet. As the mobile technology evolves, a new kind of instant messaging takes place and it is called Mobile Instant Messaging. In this technology, instant messaging services can be accessible from a portable device, ranging from smart phones to tablet computers which use the operating systems such as Android [1], iOS [2], Blackberry OS [3], Windows Phone [4], etc. There are so many mobile chat applications to use in the mobile devices.

Beside this, as the ordinary users become conscious about the information security, they demand secure mobile messaging applications. For this purpose, so many applications are provided.

In this paper we compare three of the most popular secure mobile communication applications in the manner of security. These are Telegram[5], TextSecure[6] and Threema[7].

In Section II, we specified 10 security requirements for a secure mobile messaging application. Then the selected three applications are compared under these requirements in Section IV. At the end, the overall security of them are stated in Section V.

M. Demircioğlu and H.K. Taşkın are with the Department of Cryptography, Institute of Applied Mathematics, Middle East Technical University.

M. Demircioğlu, H.K. Taşkın and S. Sarımurat are with Oran Information Technologies.

S. Sarımurat is with the Department of Computer Science and Engineering, Sabancı University.

This paper is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the project number TEYDEB-1130063 titled "Mobil Terminaller Üzerinde Kriptografik Yöntemler Kullanılarak Uçtan Uca Güvenli Bir İletişim Sisteminin Tasarlanması ve Geliştirilmesi".

II. Security Requirements

A secure mobile chat application should provide the following properties.

1) Cryptographic Primitives: Cryptographic primitives are low-level cryptographic algorithms which are well-established and generally used to create cryptographic protocols. Some of the commonly used primitives are:

- Authentication,
- Symmetric key cryptography,
- Public key cryptography,
- Digital signatures,
- One-way hash functions,
- Pseudo random number generator.

2) Forward Secrecy: Forward secrecy is a kind of key-agreement protocol. It ensures that if the attacker compromises one of the long-term keys in the future, then s/he can not compromise a session key produced from a set of long-term keys. The key used to encrypt the data must not be used to produce any other keys, and if the key used to encrypt data is produced from some other keying material, then this material must not be used again to produce any other keys. By this way, compromise of a single key allows access only to data encrypted by that single key.

3) Disk Encryption: In order to protect digital data, disk encryption techniques convert data into unreadable code which can not be deciphered easily by unauthorized people. Disk encryption software or hardware is used to encrypt every bit of data on a disk or disk volume. By this way, unauthorized access to data storage is prevented.

4) Secure Group Chat: In addition to establishing secure connection between two parties, users may want to add more participants to their conversation without leaving the secure environment provided by the application.

5) Multiple Device Support: The protocol should include support for users having multiple devices. In this scenario, any user can have multiple devices with the secure chat application. If a secure message is wanted to sent via the application, it should be reachable through all devices of the recipient and previous chat logs should also be reachable from all devices of both the sender and the recipient.

6) Fingerprint Verification: In order to prevent man-in-the-middle attack (MITM) attacks, users should verify each others' identities by using offline methods. For this purpose, parties can read their fingerprints to each other on an offline channel or they can use QR code to verify each other.

7) Contact List Synchronization: Contact list of the users should not be stored on the servers in a clear text. Instead

of this, hash value of the contact list can be stored on the servers and periodically it can be checked if new users from the contact list have started to use the application.

8) Open Source: Open source software is a software development methodology whose source code is publicly available to anyone for modification, security control or enhancement. Any software, which are not open source, may have some security flaws or backdoors.

9) Self-Destruction: Setting a time for self-destruction provides automatically deletion of messages. After the self-destruction time runs out, the messages are wiped out automatically from all devices.

10) Emergency Passphrase: There will be two pass phrases set for the application. One of them is to secure the application data and the second one for the emergency case. If someone forces user to give the passphrase, s/he will give the second passphrase and it will erase all data.

III. Description of mobile apps: Telegram, TextSecure, and Threema

In this section, we give the general descriptions of the applications which we have compared. These applications are Telegram, TextSecure and Threema, respectively.

A. Telegram

Telegram is an open source encrypted instant messaging application for iOS, Android and Windows Phones. It has also unofficial desktop applications for OS X, Windows and Ubuntu. Browser extension is also an alternative for mobile communication, but the user needs to register his/her phone on a mobile Telegram application. Telegram provides encrypted and self-destructing messages, photos, videos and documents.

The application offers two types of chats. Standard chat uses client-server encryption and it can be accessed from multiple devices. Secret Chat uses end-to-end encryption and two participating devices can only access to it. Telegram claims that third parties, including them, can not get access to the messages. Self-destruction time can be set for messages and media in a Secret Chat. After the self-destruction time runs out, the messages are wiped out automatically from both devices.

All kind of chats are encrypted with the new protocol MTProto [9] which is created by Nicolai Durov. The protocol is based on RSA 2048 encryption [10], 256-bit symmetric AES [11] encryption and Diffie–Hellman key exchange protocols [12].

Chat history of Telegram is stored on Telegram cloud servers and accessible from any number of devices. On the other hand, secure chat logs are only accessible from two participating devices, and users' other devices can not reach to the encrypted chat logs.

Telegram provides a standard group-chat option up to 200 members. However, it does not support secure group-chat. Most people use group-chat feature of messaging applications frequently, so this is a very important drawback for Telegram.

Message read status is enabled in Telegram; one check for sent messages and two checks for read messages. In this way the sender can ensure whether the message is sent or read.

Clients of Telegram are open source, so verifying the end-to-end encryption mechanism is possible. However, server-side software of Telegram is closed source, so it is not possible to verify the claimed encryption standards are properly used and well implemented.

B. TextSecure

TextSecure is an open source encrypted messaging application for Android devices. TextSecure uses an end-to-end encryption protocol which provides secrecy for instant messages, MMS and SMS. Open WhisperSystems is the developer of TextSecure and it is released under the GPLv3 license. Whisper Systems is co-founded by roboticist Stuart Anderson and security researcher Moxie Marlinspike in 2010. Moreover, Whisper Systems produced the application RedPhone which provides encrypted voice calls. The Guardian Project [8] also supports and recommends TextSecure.

By using TextSecure, users are able to send text messages, group messages, photos, videos, documents and contact information over 3G, LTE or Wi-Fi to other TextSecure users.

TextSecure uses standard SMS/MMS to communicate with non-TextSecure users. Blue and green text bubbles are used to distinguish messages that have been sent via the user's data connection and messages that have been sent via SMS/MMS. Blue bubble indicates communication over a data connection and green bubble indicates SMS-based communication.

TextSecure will use the user's data connection as default if possible to send the message. In other words, if the user sends a message to another registered TextSecure user, it will be treated as an additional data transfer and there will be no SMS charge associated with the message. If there is not any available data connection, then the application will use standard SMS/MMS to send the message.

The application will automatically encrypt all conversations held with other registered TextSecure users. In the user interface, encrypted messages are denoted by a lock icon. Media and other attachments are encrypted in the same way as other messages.

TextSecure also encrypts users' text locally on their smartphones. Anything stored by the app is encrypted on their Android device. The application will encrypt all conversations ongoing with other registered TextSecure users and the encrypted messages are indicated by a lock icon. In addition, media and other attachments are encrypted in the same way. Therefore, if the attacker is able to gain physical access to users' phone, s/he will not be able to obtain plain data of this encrypted data.

TextSecure provides group chat. Users can create a group, name it, set an avatar, and invite unlimited number of recipients, who are all able to leave at will.

TextSecure uses Off-the-Record (OTR) [15] protocol and made some improvements to the deniability and forward secrecy aspects, and added a mechanism to allow the ephemeral key negotiation to work asynchronously. TextSecure uses Curve25519 [13], AES-256, and HMAC-SHA256.

Verification of the users communicating with is done via comparing their fingerprints. Thus, MITM attacks are prevented. The keys which are used to encrypt the user's messages

are stored on the device alone, and they are protected by an additional layer of encryption if the user enabled a passphrase.

In order to ensure that new AES keys are used for every single message, TextSecure uses the cryptographic ratchet [6]. It also provides both forward secrecy and future secrecy properties. The protocol also features enhanced deniability properties.

TextSecure-Server is the software that handles message routing for the data channel. Client-server communication is protected by TLS/SSL. REST API and push messaging are used to handle communication.

The servers do not store any contact information. Hashed contact numbers (with no other accompanying information) are periodically uploaded to the servers in order to determine which contacts are also TextSecure users, but this data is never stored.

C. Threema

Threema is a commercial encrypted messaging application for iOS and Android devices. It provides asymmetrically end-to-end encrypted instant messages, group conversations, multimedia files and voice messages. Users can also share their GPS locations securely via Threema.

After the application installed for the first time, a key pair is generated and the public key is sent to the server while the private key is kept on the user's device. Then all messages and media files that are sent to other Threema users are encrypted with their respective public key.

The Swiss company Threema Kasper Systems GmbH develops Threema. The application is especially popular in German speaking countries.

Threema is not an open source application. Therefore, it can not be reviewed without the agreement of its authors. Validation Logging feature of the application can be used to log the incoming and outgoing messages. This feature logs the bodies of the message, but not all the data that is exchanged with the server. So, there is no way to prove that the message that is being logged correlates to data that is being sent by the application. So, it is hard to trust completely to this application.

Threema is using the NaCl Cryptography Library [16] and it works properly on both iOS and Android.

In order to show the verification level of contacts Threema shows three dots next to every contact. One red dot means no verification, two yellow dots mean that a user's identity has been verified by the server , and three green dots mean that user's key and ID have been personally scanned and verified. This is a good way to direct users to use strong security and prevent MITM attacks.

The protocol does not provide a Perfect Forward Secrecy, but they claim to have implemented it at the transport layer. Forward secrecy is provided on the network connection (not on the end-to-end layer). Client and server negotiate ephemeral random keys. They are stored in RAM and replaced every time the app restarts (or at least once every seven days).

Asymmetric cryptography is used to protect the communication between two users and the communication between the application and the servers. ECDH on Curve25519 is used with a hash function and a random nonce to generate a unique 256 bit symmetric key for each message, and then XSalsa20[14] stream cipher is used to encrypt the message. A 128 bit MAC is added to each message to detect changes. Servers' public keys are hard-coded into the application.

Threema has two layers of encryption; one is the end-to-end layer between two users, and the other is an additional layer between the app and the servers to protect against eavesdropping. All local files are also stored encrypted on the device. Encryption and decryption of the messages are made on the device. Therefore, they claim third parties can not decrypt the messages.

IV. Comparison

In this section the security requirements are compared for Telegram, TextSecure and Threema. Each topic mentioned in Section II are explained separately and the differences between these applications are shown briefly in the Table I.

1) **Cryptographic Primitives:** State of art cryptographic algorithms should be used in the protocol.
   Telegram uses its own protocol called MTProto Mobile Protocol. It consists of AES-256 encryption with infinite garble extension (IGE), and hash function SHA-1. Using SHA-1 is not a good option, because studies [17] showed that SHA1 has some major vulnerabilities.
   TextSecure uses ECDH on Curve25519, AES-256 for symmetric key encryption, and HMAC-SHA256 in its protocol. These algorithms have been used in hundreds of applications and tested for many years. So, TextSecure has chosen to use state of art algorithms.
   Threema uses ECDH on Curve25519 and stream cipher XSalsa20 to encrypt the message. 128 bit MAC is also added to the end of each message.
   In conlusion, TextSecure and Threema prefer to use state of art cryptographic algorithms. But Telegram uses SHA-1 which has some security problems, and their AES-IGE mode selection is questionable.
2) **Forward Secrecy:** In forward secrecy, if one of the long term keys is compromised, a session key produced from a set of long-term keys can not be recovered.
   Telegram supports forward secrecy in an impractical way. In order to provide forward secrecy, user should delete message logs and start a new secure conversation. So, forward secrecy is achieved manually.
   TextSecure supports forward secrecy and it uses its own OTR protocol.
   Threema provides forward secrecy between client and server. There is no end to end forward secrecy between clients.
   Therefore, TextSecure is the only application providing forward secrecy among these three applications.
3) **Disk Encryption:** All data of the application should be stored encrypted in the disk of the mobile device. Otherwise, physical access to the mobile device will lead to a security vulnerability.
   Telegram does not provide disk encryption.

In *TextSecure*, when the application started, it asks the passphrase which was set on the first installation. All data used by TextSecure are encrypted by this passphrase. Unless the application is completely terminated, it continues to operate on background and does not ask for the passphrase again.
Threema uses different methods of disk encryption for iOS and Android. In iOS, NSFileProtectionComplete Library is used to encrypt the data, and the device PIN is used for the encryption key. While the screen is unlocked, the application can be used without a key, but a four digit PIN can be set in the application settings for an extra security.
Therefore, TextSecure and Threema both provide disk encryption, but Telegram does not.

4) **Secure Group Chat:** A group chat is a standard part of the mobile chat applications. But in a secure chat application, it is expected that the group chat is also secured.
Telegram provides a group chat up to 200 participants, but it is not a secure chat. The application provides secure chat for only one to one conversations.
TextSecure provides a secure group chat option with unlimited participants.
Threema supports a secure group chat up to 20 users, but none of the participants of the group can be removed. For any change in the group participants, the conversation is restarted again with new encryption keys.
Therefore, only TextSecure and Telegram support secure group chat, but TextSecure looks like a little bit better.
5) **Multiple Device Support:** This option is not available in both *TextSecure* and *Threema*. But in Threema it is planned to add multiple device support in the future.
In *Telegram*, users can log-in multiple devices with the same phone number. While adding a new device, a verification code is sent to other previously registered devices. After verification completed, the name of the new added device and its IP address is also sent to other devices. But when a secure chat starts, the messages are received by only one device.
So, none of the above applications support multiple devices.
6) **Fingerprint Verification:** Verification of users by using offline methods will prevent MITM attacks.
In *Telegram*, the symmetric key used for the session is showed as a QR code like picture on the device screen, and then users can compare them.
TextSecure users can verify their identities by showing their fingerprints in text format to each other, or by scanning QR code of their finger prints.
In *Threema*, if users verify each other via QR code, it will be stated as the highest security level verification and these users are labeled with three dots.
So, all applications have an offline verification methods.
7) **Contact List Synchronization:** Both *Telegram* and *TextSecure* send the contact list of the user to the servers in a clear text format to find the other users using the same application. On the other hand, *Threema* sends the hash values of phone numbers and e-mail addresses in the contact list of the user to the server.
Since the space for all phone numbers is $10^{10} \approx 2^{33}$, it is easy to build a rainbow table for phone numbers and compare the hash values. So, taking hash of the phone numbers is not an enough security measure.
8) **Open Source:** *Threema* is not an open source application, so it is not possible to verify what they say about their security. On the other hand *Telegram* and *TextSecure* are open source. Therefore, one can check the code easily to verify its security.
9) **Self-Destruction:** This option is only available for *Telegram* users. In a session, the self-destruction time can be set at anytime. After the time expires, the message is deleted automatically.
10) **Emergency Passphrase:** This option is not available in these applications. But in security forums, this feature is wanted in the case of emergency.

V. Conclusion

In this paper, 10 security criteria are defined to compare Telegram, TextSecure and Threema. These criteria are explained in Section II and the applications are compared in Section IV. Both three applications doesn't support multiple devices and emergency passphrase.

Although Telegram becomes very popular in a short time, adding some more features (e.g. disk encryption and secure chat) to it will be good for its security.

Threema comes the second in this comparison. Not having an efficient secure group chat and not being an open source are the main disadvantages of this application.

Among these three applications TextSecure provides better features for a secure chat. The lack of iOS support is the main shortage.

Acknowledgment

The authors would like to thank Ali Aydın Selçuk and Mutlu Çiçek for their support and assistance with this project.

TABLE I
COMPARISON OF TELEGRAM, TEXTSECURE AND THREEMA

	Telegram	**TextSecure**	**Threema**
Cryptographic Primitives	Own protocol which consists of AES-256 in IGE mode and SHA-1	ECDH on Curve25519, AES-256, HMAC-SHA256	ECDH on Curve25519, XSalsa20
Forward Secrecy	Impractically supported	Supported	Supported between client-server communication
Disk Encryption	Not supported	Supported	Supported
Secure Group Chat	Not supported	Supported up to 200 participants	Supported up to 20 users
Multiple Device Support	Supported, but secure messages are received by only one device	Not supported	Not supported
Fingerprint Verification	The symmetric key used for the session is showed as a QR code like picture on the device screen	Showing their fingerprints in text format to each other, or by scanning QR code of their finger prints	Verifying each other via QR code
Contact List Sync.	Sends the contact list of the user to the servers in a clear text format	Sends the contact list of the user to the servers in a clear text format	Sends the hash values of phone numbers and e-mail addresses in the contact list of the user to the server.
Open Source	Open source software	Open source software	Proprietary software
Self-Destruction	Supported	Not supported	Not supported
Emergency Passphrase	Not supported	Not supported	Not supported

REFERENCES

[1] Official Android Page, *http://www.android.com*
[2] Official iOS Page, *https://www.apple.com/ios*
[3] Blackberry 10 OS *http://us.blackberry.com/software/smartphones/blackberry-10-os.html*
[4] Official Windows Phone Page, *www.windowsphone.com*
[5] Telegram, *https://telegram.org/*
[6] TextSecure, *https://whispersystems.org/*
[7] Threema, *https://threema.ch/en/*
[8] The Guardian Project, *https://guardianproject.info/*
[9] MTProto Mobile Protocol, *https://core.telegram.org/mtproto*
[10] RSA Cryptosystem, *http://en.wikipedia.org/wiki/RSA*
[11] AES Block Cipher, *http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf*
[12] Diffie-Hellman, *http://www.ietf.org/rfc/rfc2631.txt*
[13] Curve25519: high-speed elliptic-curve cryptographyl, *http://cr.yp.to/ecdh.html*
[14] The Salsa20 core, *http://cr.yp.to/salsa20.html*
[15] Nikita Borisov, Ian Goldberg, Eric Brewer (2004-10-28), *Off-the-Record Communication, or, Why Not To Use PGP*, Workshop on Privacy in the Electronic Society. Retrieved 2014-03-06
[16] NaCl: Networking and Cryptography library *http://nacl.cr.yp.to/l*
[17] Xiaoyun Wang , Yiqun Lisa Yin , Hongbo Yu, *Finding collisions in the full SHA-1*, in Advances in Cryptology, CRYPTO'05

Murat Demircioğlu Murat Demircioğlu (S'14) was born in Istanbul, Turkey, in 1986. He graduated from Department of Mathematics at Middle East Technical University, Turkey, in 2009. He then received his M.Sc. degree in Cryptography from Institute of Applied Mathematics at Middle East Technical University, Turkey in 2011 and he has been continuing his Ph.D. studies at the same department since 2011. During his studies, he has worked on Quantum cryptography and also involved in a project about Differential Cryptanalysis funded by TÜBİTAK. He worked as a Research Assistant in Institute of Applied Mathematics of Middle East Technical University, Turkey, from 2012 to 2013. Since 2013, he has been working as Cyber Security Specialist in Oran Teknoloji, Turkey.

Halil Kemal Taşkın Halil Kemal Taşkın (S'14) graduated from Department of Mathematics Teaching at Gazi University, Turkey in 2009. He then received his M.Sc. degree in Cryptography from Institute of Applied Mathematics at Middle East Technical University in 2011 and he has been continuing his Ph.D. studies at the same department since 2011. During his studies, he has worked on Block Cipher Cryptanalysis and involved in a project about Differential Cryptanalysis funded by TÜBİTAK. Since 2013, he has been working as Cyber Security Specialist in Oran Teknoloji, Turkey.

Salim Sarımurat Salim Sarımurat graduated from Computer Engineering Department of Bilkent University, Turkey, in 2011. He then received his M.S. degree from Computer Science and Engineering Department of Sabanci University, Turkey, 2013. During his M.S. studies, he was working on a project, funded by Scientific and Technological Research Council of Turkey (TÜBİTAK), to develop secure key predistribution methods for mobile and multiphase Wireless Sensor Networks that improves resiliency performance against node capture attacks. Salim currently works as an Information Security Consultant.