Online Conversation Application with Confidentiality, Anonymity, and Identity Requirements

Pedro Fernandes[1] and António Pinto[1,2(✉)]

[1] GCC, CIICESI, ESTG, Politécnico do Porto, Porto, Portugal
8080084@estg.ipp.pt
[2] CRACS & INESC TEC, Porto, Portugal
apinto@inesctec.pt

Abstract. The increase in usage of smartphones and the ubiquity of Internet access have made mobile communications services very attractive to users. Messaging services are among the most popular services on the Internet. In recent years, this services started to support confidentiality and anonymity. A recurrent problem with the existing messaging solutions is their lack of resistance to impersonation attacks. The proposed solution addresses the impersonation problem, without neglecting user confidentiality and anonymity, by forcing users to exchange the required cryptographic material among themselves. Moreover, this exchange must use a proximity communication technology, forcing the users to physically meet.

Keywords: Impersonation · Anonymity · Online conversation

1 Introduction

The increase in usage of smartphones and the ubiquity of Internet access have made mobile communications services very attractive to users. Messaging services being among the most popular because these of its availability, functionality and lower costs of communication. In particular, these services make international communications free, if the user already has Internet connectivity, and very attractive due to functionalities as the use of emoji or photo and video sharing.

In recent years, these messaging services started to support end-to-end (E2E) encryption in order to protect the transmitted content from eavesdropping when used over unsafe communication channels. In E2E encryption, the messages are encrypted at the source terminal, sent through the network, and decrypted only at the destination terminal. Servers, if used, are expected to not be able to access the exchanged messages in clear text form [6]. This may not the be case if the server has access to the cryptographic material used to encrypt the messages.

J.F. De Paz et al. (eds.), *Ambient Intelligence—Software and Applications—8th International Symposium on Ambient Intelligence (ISAmI 2017)*,
Advances in Intelligent Systems and Computing 615, DOI 10.1007/978-3-319-61118-1_6

Whenever the server securely transfers messages from source to destination and is unable to access the messages, or to identify the user who sent the message, we are in presence of a secure messaging platform that enables both confidentiality and anonymity This is sometimes referred to zero-knowledge applications [8].

The interest in secure forms of sending online messages has grown substantially and lead the Electronic Frontier Foundation (EFF) to evaluate [3] the existing ecosystem of smartphone applications that offer secure messaging services. The list of messaging applications is very long and difficult to maintain due to the frequent appearance of new ones. This study is currently identified by EFF as outdated and motivated the authors to extend it.

All modern messaging applications, to the best of our knowledge, enable users to directly communicate without requiring them to confirm their real identify. This opens the possibility of a user assuming the identify of another, i.e. user impersonation. This is a problem that potentiates ill-intentioned users, possibly with criminal intent, to try and deceive more susceptible users such as children or less tech-savvy older people. The proposed solution, while maintaining users' privacy and anonymity, enables secure E2E communications, solves this user impersonation problem by requiring a first, physical interaction between the communicating users. This first interaction is based on a proximity communication technology.

The paper is organized in sections. Section 2 describes the related work, in particular it compares secure messaging applications. Section 3 details the proposed solution, which is then evaluated in Sect. 4. Section 5 concludes this paper.

2 Related Work

A set of secure messaging applications was selected from the ones available in Google Play and App Store. The selection was based on the ones that better performed in the previously mentioned study of the EFF and that had the more installations. In particular, *TextSecure*, *Signal*, *Telegram*, *WhatsApp*, *Threema* and *Wickr* were selected.

TextSecure is a free and open-source mobile application for the Android platform that allows the to send encrypted text messages. It was first released in 2010 by *Open Whisper Systems* [12]. In October 2015, *TextSecure* had been installed over 1 million times through Google Play [9]. The protocol initially used by *TextSecure* was a protocol derived from the Off-the-Record (OTR) protocol [11]. It comprised four stages: (1) registering; (2) sending/receiving a first message; (3) sending of a follow-up message; (4) sending of a response. The application relays its messages to the destination to the server that, in turn, transmits the messages to the destination. The parties communicate with the server via Representational State Transfer (REST)-Application Programming Interface (API) over secure Hypertext Transfer Protocol (HTTPS). The delivery of the actual message is performed via Google Cloud Messaging (GCM) which basically acts as a message delivery intermediary [18]. Currently, *TextSecure* implements the algorithm *Double Ratchet* (also known as *Axolotl ratchet*).

Signal is the successor of both the *RedPhone* (voice calls encryption) and of the *TextSecure*. *Signal* was launched in 2015 and is a secure instant messaging and voice call application that uses E2E encryption. The encryption keys of the users are generated and stored in their smartphones and not on the application servers [3]. *Signal* was built with mechanisms to resist Man-in-The-Middle (MiTM) attacks. For voice calls, the application displays a word on the screen, if the two words match at both ends of the call, then the call is considered secure [13,15]. There is a similar mechanism for messages that consists on the mutual verification of digital signatures. The protocol used by the *Signal* application is open source and is known as the *Axolotl* protocol.

Telegram provides E2E message encryption and self-destructing messages. *Telegram* has more than 100 million monthly active users. The application offers two types of chats: the standard chat that uses an encryption key that is shared with the server and can be accessed from multiple smartphones; and the secret chat (*Telegram* (*Secret Chats*)) that uses E2E encryption and can only be accessed by the devices that are in possession of the required cryptographic material. The adopted protocol, named *MTProto*, uses RSA2048 [1], Advanced Encryption Standard (AES) 256 bits and the Diffie-Hellman key exchange [2]. The *MTProto* protocol comprises three components [14]: the high-level component, the authorization component, and the transport component. The high-level component defines the method by which the API queries and responses are converted into binary messages. The authorization component defines the methods used for user authentication and messages encryption. The transport component defines the method to be used by clients and server for message transmission over a network protocol. The *MTProto* supports multiple transport modes, such as Hypertext Transfer Protocol (HTTP), HTTPS, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

WhatsApp is another Internet messaging application. Numbers of September of 2015 put their user base in the 900 million users [10]. *WhatsApp* uses a store and forward approach for message transmission. When a user wants to send a new message, it is first stored on the *WhatsApp* server, and then the server relays the message to the destination user. Once the message is received by the destination, it is removed from the server database [5]. *WhatsApp* uses the same protocol as *TextSecure* (*Axolotl* or *Double Ratchet algorithm*).

Threema is another free and open source Internet messaging application with support for E2E encryption and user anonymity. As of June 2015, *Threema* had 3.5 million users, most of them from German-speaking countries. Each user, at the application start, receives a randomly assigned a *Three-ID* that will be used for user identification. In this process, neither the user mobile phone number or email address is required. Users can verify the identity of their *Threema* contacts by scanning their QR code when they physically meet. Using this feature, users can be sure that they contain the correct public key of their contacts, providing message confidentiality and resistance to MiTM attacks [16]. *Threema* was designed to store as little data on servers as possible. The contact lists are managed only on the users smartphones and messages are deleted immediately after they have been delivered. *Threema* supports E2E encryption [17].

Wickr is another free Internet messaging application that offers confidentiality. It includes the ability to set a time-to-live for each message. The recipient's application erases the encrypted message from their smartphones, trying to ensure that the message can no longer be retrieved. The *Wickr* Secure Messaging protocol is specifically designed to prevent servers from accessing both the keys or detailed information from users. Supports E2E encryption by implementing multiple encryption layers [7]. The data, stored or in transit, is encrypted with AES256. Each message is encrypted with a new encryption key, deleted after its use (Perfect Forward Secrecy). Message encryption keys are then encrypted with the public key (Elliptic Curve Diffie Hellman (ECDH) 521) of the recipient. All user content is deleted after the user logs off. The unique identifier of the smartphone, Unique Device Identifier (UDID), is never sent to the *Wickr* servers in an effort to assure user anonymity. *Wickr*'s Secure Shredder erases all data on the smartphone so it can not be recovered.

These applications were compared in the Secure Messaging Scorecard of EFF [4]. Table 1 extends this comparison for the selected applications. The comparison criteria is fivefold: (1) transport; (2) provider; (3) entities; (4) anonymity; (5) identity. If all communications are encrypted while being transported through the network, then the first criteria is satisfied. The second criteria requires that all communications must be encrypted E2E, which means that the keys needed to encrypt/decrypt messages must be generated and stored on the smartphones and not on the servers. The third criteria requires that there is an internal method for the verification of the identity of the involved entities and of the integrity of the channel, even when the server or third parties are compromised. The forth criteria requires that the identity of the user is not known by the server. The last criteria requires that the users can be sure of the other users identity because that had a physical interaction as proof of their identity.

For instance, the *TextSecure* application despite satisfying the three initial criteria, does not provide user anonymity nor requires any prior physical interaction between users in order to establish a conversation between them. Off the selected applications, only *Threema* permits the use of an interaction between the users as a form identity confirmation, but because it can be made remotely, undermines its fulfilling of the last criteria.

Table 1. Application comparison

	Transport	Provider	Entities	Anonymity	Identity
Text-Secure	Yes	Yes	Yes	No	No
Signal	Yes	Yes	Yes	Yes	No
Telegram	Yes	No	No	No	No
Telegram (*Secret-Chats*)	Yes	Yes	Yes	No	No
WhatsApp	Yes	No	No	No	No
Threema	Yes	Yes	Yes	Yes	No
Wickr	Yes	Yes	Yes	Yes	No

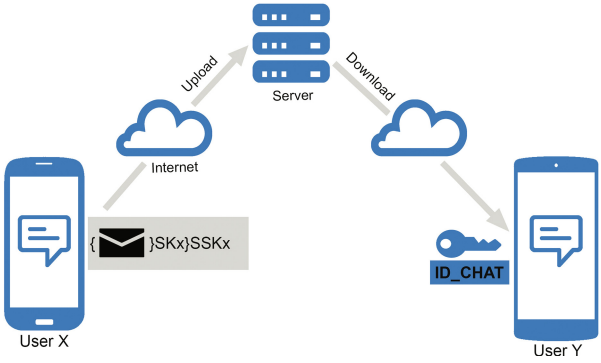

Fig. 1. Application architecture

3 Eko

The architecture of the proposed solution, named Eko, is depicted in Fig. 1. It comprises end user terminals, which are smartphones, and a server. An user X, in order to send a secure message to an user Y, prior to its upload to the server, firstly encrypts the message with the chat symmetric key (SK_x), and then encrypts this result with a second chat symmetric key (SSK_x). The SK_x is only known by the users X and Y, while the SSK_x is known be the users and the server.

All cryptographic material is generated at the smartphones, except for the SSK_x that is generated by the server. The conversation identifier (Id) is also generated by the server to avoid Id collision. Each conversation is represented by a data structure named ID_CHAT that is detailed in Table 2. In addition to Id and the cryptographic material, it includes a time stamp and a message validity in number of days.

All data transmitted by the application uses E2E encryption, guaranteeing that all communications with other users is confidential. The server uses the conversation identifiers to select the appropriate decryption key to process each message but is unable to decrypt the message content and to obtain the identification of the users that take part in each conversation, thus providing anonymity.

Table 2. Conversation data structure: ID_CHAT

Variable	Description
SK_i	Conversation user key
SSK_i	Conversation key shared between users and server
Id	Conversation identifier
T_s	*Timestamp*
V	Validity (1, 10, 15, 30 days)

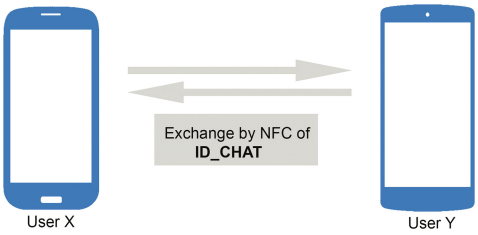

Fig. 2. *ID_CHAT* exchange between users

Users, in order to start exchanging messages with each other, first have to undergo a physical interaction consisting of exchanging an instance of the data structure *ID_CHAT* be means of a proximity communication technology. Example technologies being Near Field Communications (NFC) or Bluetooth Low Energy (BLE). The *ID_CHAT* data structure is generated by one user, and then directly shared with the remaining users, without interaction with the server, as shown in Fig. 2.

4 Validation

Prototype client and server applications were implemented to validate the proposed solution. The server was developed using the Laravel framework and contains the necessary access points for the operation of the clients. The client application was developed using the Ionic framework that allows multi-platform development. The functional assessment was successfully performed and the identified requirements were satisfied.

A security analysis was also performed considering the confidentiality, anonymity and impersonation. The *confidentiality* of the messages exchanged between client and server is obtained by means of E2E encryption. The keys used to encrypt messages in conversations, the SK_x key, is generated in the smartphone of the user that creates the conversation and stored locally. It is never uploaded to the server. The second key, the SSK_x key, is known by the participants of a conversation and by the server. Finally, being a web-based service, the server is deployed only in its secure mode (HTTPS).

The proposed solution guarantees user *anonymity* by the way it creates and uses the *ID_CHAT* data structure. The identity of the user that sends each messages is unknown to the server. Each message assumes the form:$(chatId : Ts : \{\{Username : Msg\}_{SK_X} : chatId : Ts\}_{SSK_X})$, where $\{a\}_b$ means a encrypted with key b, and $a : b$ mean the concatenation of a with b. The identity of the user that sends each message can only be decrypted with the SK_x key that exists only in the *ID_CHAT* data structure on the smartphones.

The proposed solution addresses the *impersonation* problem by imposing a previous interaction between users for the conversation to take place. This interaction comprises the generation of a new *ID_CHAT* data structure and

its direct exchange with the other user's smartphone by means of a proximity communication technology. This way, the users must physically meet in order to communicate.

5 Conclusion

Messaging services are among the most popular services on the Internet. In recent years, this services started to support confidentiality and anonymity. A recurrent problem with the existing messaging solutions is their lack of resistance to impersonation attacks. The proposed solution addresses the impersonation problem without neglecting user confidentiality and anonymity. A prototype of the proposed solution was implemented and functionally verified. A analysis of the security of the proposed solution was also performed.

References

1. Calderbank, M.: The RSA Cryptosystem: History, Algorithm, Primes (2007)
2. Demircioglu, M., Taskın, H.K., Sarımurat, S.: Security analysis of the encrypted mobile communication applications (2014)
3. Electronic Frontier Foundation: Secure messaging scorecard. Which apps and tools actually keep your messages safe?. Accessed 11 Sept 2016
4. Electronic Frontier Foundation, Julia, A., Joseph, B.: Secure Messaging Scorecard. https://www.eff.org/secure-messaging-scorecard (2014)
5. Gaurav, R.: How WhatsApp Works. http://digitalperiod.com/explore-whatsapp-clock-sign-and-tick/ (2015)
6. Greenberg, A.: Hacker Lexicon: What Is End-to-End Encryption?. In: WIRED (2015). Accessed 17 Mar 2016
7. Inc, W.: How Wickr's Encryption Works. https://www.wickr.com/security/how-it-works. Accessed 17 Feb 2016
8. Pedersen, C., Dahl, D.: Crypton: Zero-Knowledge Application Framework (2014). Accessed 2 Mar 2016
9. Play, G.: TextSecure Private Messenger. Accessed 17 Feb 2016
10. Sun, L.: Facebook Inc.'s WhatsApp Hits 900 Million Users: What Now?. The Motley Fool. http://www.fool.com/investing/general/2015/09/11/facebook-incs-whatsapp-hits-900-million-users-what.aspx (2015)
11. Systems, O.W.: Advanced cryptographic ratcheting. https://whispersystems.org/blog/advanced-ratcheting/. Accessed 28 Sept 2016
12. Tactical Technology Collective and Front Line Defenders: TextSecure for Android, security in-a-box. https://securityinabox.org/en/guide/textsecure/android (2009)
13. TechCrunch. AOL: Talk Private to Me: Free, Worldwide, Encrypted Voice Calls with Signal for iPhone. Accessed 13 Sept 2016
14. Telegram: Mtproto mobile protocol
15. The Zfone Project: Exactly how does Zfone and ZRTP protect against a man-in-the-middle (MiTM) attack?. Accessed 13 Sept 2016
16. Threema GmbH: Threema
17. Threema GmbH: Threema Cryptography Whitepaper. https://threema.ch/en/faq. Accessed 10 Sept 2016
18. Tilman, F., Christian, M., Christoph, B., Florian, B., Jorg, S., Thorsten, H.: How Secure is TextSecure? (2014)