End-to-End Secure and Privacy Preserving Mobile Chat Application

Raja Naeem Akram and Ryan K.L. Ko

Cyber Security Lab., Department of Computer Science, University of Waikato, Hamilton, New Zealand
{rnakram,ryan}@waikato.ac.nz

Abstract. Since the 1990s, two technologies have reshaped how we see and experience the world around us. These technologies are the Internet and mobile communication, especially smartphones. The Internet provides a cheap and convenient way to explore and communicate with distant people. A multitude of services have converged on the smartphone platform, and potentially the most notable is social networking. With increased interconnectivity and use of online services, concerns about consumers' security and privacy are growing. In this paper, we evaluate the security- and privacy-preserving features provided by existing mobile chat services. This paper also puts forwards a basic framework for an End-to-End (E2E) security and privacy-preserving mobile chat service and associated requirements. We implemented the proposal to provide proof-of-concept and evaluate the technical difficulty of satisfying the stipulated security and privacy requirements.

1 Introduction

The instant messaging services provided by applications like WhatsApp, Apple iMessage and BlackBerry Messenger are overtaking traditional SMS services [1], becoming the preferred medium of communication for millions of smartphone users[1]. However, the security and privacy-preserving features of different mobile applications have come under the spot-light [3]. There are different security and privacy features provided by different mobile chat applications, but there are not many mobile chat applications that provide an End-to-End (E2E) security and privacy-preserving service to their customers.

In this paper, we focus on such a mobile chat service. We propose a framework for building such a service and then evaluate the technical challenges involved in implementing it, to provide a proof-of-concept and understand any potential technical issues which may restrict such features from being implemented by mainstream mobile chat service providers.

[1] Financial Times report [1] put the number of daily instant messages at 41billion and WhatsApp has more than 200 million active monthly users [2].

D. Naccache and D. Sauveron (Eds.): WISTP 2014, LNCS 8501, pp. 124–139, 2014.
© IFIP International Federation for Information Processing 2014

1.1 Contributions of the Paper

This paper deals with the security and privacy-related challenges faced in the design, development and maintenance of a mobile chat service. The main contributions of this study are:

1. End-to-End (E2E) security and privacy-preserving architecture for mobile chat services
2. Secure key exchange, even when communicating parties are not online (i.e. for offline messages)
3. User-to-User (U2U) authentication mechanism[2]
4. Implementation analysis of proposed architecture

1.2 Structure of the Paper

Section 2 discusses the evolution of mobile phone technology and how mobile chat is becoming a convenient method of communication. In section 3, we explore the existing commercial applications that provide different degrees of security and privacy features. In addition, we also stipulate the security and privacy requirements for an E2E secure and privacy-preserving mobile chat service. Subsequently, we describe the proposed framework along with the details of crucial operations. In section 4, practical implementation experience is presented. Section 5 provides an overall analysis of the proposed/implemented framework. Finally in section 6 we provide potential future research directions and conclude the paper.

2 Mobile Phones

In this section, we briefly visit smartphone technology in order to understand the scale of the market, which directly relates to the security and privacy concerns of mobile chat users.

2.1 Smartphones: A Paradigm Shift

The mobile phone platform has evolved a long way from the original simple medium of voice and text communication to become the hub of the digital world. Mobile phones, along with being an entertainment hub, have also developed into a social construct that has affiliations and emotional attachments for individuals. It is also becoming the predominant medium for connecting with the world through social media sites/applications [4,5].

The so-called "App Culture" promoted by Apple Inc. [6,7], has enabled users to seamlessly download any application they desire. This has opened the smartphone platform to a wide range of companies and services. One of the most

[2] An authentication mechanism that enables individual users to authenticate each other during a chat session without involving the respective chat servers.

prominent services provided by different applications on smartphones is mobile chat. It provides a potentially convenient and cost-effective alternative to traditional voice and SMS[3]. Mobile chat services have the potential to eclipse SMS communication and this trend is becoming more obvious on a daily basis[1]. With consumers' increasing reliance on mobile chat services, security and privacy features are becoming serious concerns [8,9].

Consumers use a mobile chat service to communicate with each other, a process that can include relaying personal information. The security and privacy of such communications should be taken seriously. However, recent episodes of vulnerability in the major chat services (i.e. WhatsApp [10]) reveal that they might not be robustly implementing security and privacy features.

In the following sections, we briefly explore the range of mobile chat applications available on Android and iOS. This discussion provides an analysis of existing work in the commercial arena. In section 2.6, we discuss existing academic work related to security and privacy-preserving chat software. The selection of commercially available chat software was made in a manner that reflects the existing approaches, and it is by no means an exhaustive list.

2.2 WhatsApp

WhatsApp is considered to be one of the biggest mobile chat services available on different platforms (e.g. iOS, and Android). The architecture of the service is proprietary and the details in this section are taken from a range of resources; notably from [11]. The main focus of the product is on messaging and privacy concerns are secondary. WhatsApp does not store any messages on the server: the chat history is stored on the client's device. The client application uses SSL [12] to connect to the server; however, a recent blog posting [10] discussed the deployment of SSL version 2. This deployment might open up WhatsApp to attacks on SSL 2.0. There is no E2E encryption to provide security in chat messages between sender and receiver. Therefore, the message server can read the messages exchanged.

2.3 BlackBerry Messenger

BlackBerry Messenger (BBM), for better or worse, is perceived to be a secure messaging service. In this section, we examine the consumer version of the BBM, not the business application. An analysis conducted by Communications Security Establishment Canada (CSEC) in 2011 found a number of issues with the BBM [13]. Messages are encrypted but the cryptographic key used is a "global key" that is common to every BlackBerry device/application. The use of a single key to encrypt all messages sent using BBM enables the message server to decrypt

[3] Smartphone-based mobile chat services use the Internet as the communication medium, and this might be provided by a Telecom operator. In some areas the cost for mobile data might reduce its benefits in comparison to traditional Telecom services.

the messages. In addition, there is a potential for malicious users to gain access to the "global key" and decrypt any intercepted messages sent or received via BBM.

2.4 Wickr

The most recent addition to the range of secure chat applications is Wickr. Although most of their architecture is proprietary, in this section we discuss the features they claim to offer[4]. They claim that they encrypt individual messages using a cryptographic key. However, it is difficult to determine whether these keys are generated by the message server or the clients. They only claim that users' private keys are not communicated to the server. Furthermore, it is claimed that device, location and Meta information about users and messages is protected, providing a strong privacy mechanism. Communication between the device and the message server is protected by TLS [14].

2.5 Silent Text

Similar to mobile chat applications discussed above, the complete architectural design of Silent Text is proprietary. There is fragmentary information available on their website[5]. Silent Text enables E2E key exchange and secure message communication using the Silent Circle Instant Messaging Protocol (SCIMP) [15]. Each message is encrypted with a new key that is expanded/derived from a master secret shared between the communicating entities. The message server does not handle any key material and does not store any messages. To share the master secret, the communicating entities have to exchange several messages (before they can actually communicate). It is not clear from their white paper [15] and website whether their key sharing protocol supports offline communication[6].

2.6 Related Work

Security and privacy issues in relation to smart phones have received considerable attention [16,17,18] with regard to mobile chat applications. Although there are a number of mobile chat applications that claim to provide a secure service, their complete architecture is not publicly available. To our best knowledge there are not many publications that describe such systems. Secure text messaging systems have a strong foundation in proposals like Media Path Key Agreement for Unicast Secure RTP (ZRTP) [19], Off-the-Record (OTR) [20] and A Secure Text Messaging Protocol [21]. In this paper, we aim to present a potential architecture along with security and privacy-preserving architecture to provide a complete architecture, thereby filling the gap in the existing work in the area of mobile chat applications.

[4] Claims were made on their website `https://www.mywickr.com/en/howitworks.php`

[5] Silent text website `https://silentcircle.com/web/silent-text/`

[6] Offline Communication: In mobile chat applications, a user can send messages to other users even when they are not online, using so-called "offline communication".

3 Secure and Privacy Preserving Mobile Chat

In this section, we first discuss the security and privacy requirements of mobile chat applications. In the remaining part of this section, we detail a proposed architecture and describe its features.

3.1 Secure and Privacy Preserving Mobile Chat Requirements

Before we present the details of the proposed architecture for mobile chat applications, this section provides a brief list of requirements that any such proposal should meet:

Req1 The sign-up process should require minimal information related to the user. The account creation process should not rely heavily on Personal Identity Information (PII)
Req2 The key exchange process should be secure, seamless and support off-line chat
Req3 Encryption/decryption of messages should not require user interaction (i.e. least interaction)
Req4 Secure offline messages can be communicated securely along with potential key share
Req5 Individual users have a mechanism to authenticate each other, assuring themselves they are communicating with the right person
Req6 Communications are not stored on the chat server. Individual chat sessions can be stored on the user's device
Req7 Local chat storage should be adequately protected
Req8 To safeguard the privacy of the users and their chat, the message-server should not be able to retrieve the messages.

3.2 Proposed Architecture

The generic architecture of a secure and privacy-preserving mobile chat application is shown in Figure 1.

After downloading a mobile chat application, the user of mobile 'A' initiates the sign-up process. The sign-up process is used either to create a new account or to sign in using an existing account (credentials). The chat server, which consists of a membership server and a message server, initiates the account creation process. The membership server manages the user's accounts, associated credentials and (optionally) the user's contact-list. The message server handles the message communication between users, whether both users are online or if the intended recipient is offline. If the recipient is offline, the message will be stored in the offline message store. These messages are temporarily stored and once they are delivered to the respective recipients they are deleted. The dotted line represents virtual communication between the users of mobiles 'A' and 'B', via the message server.

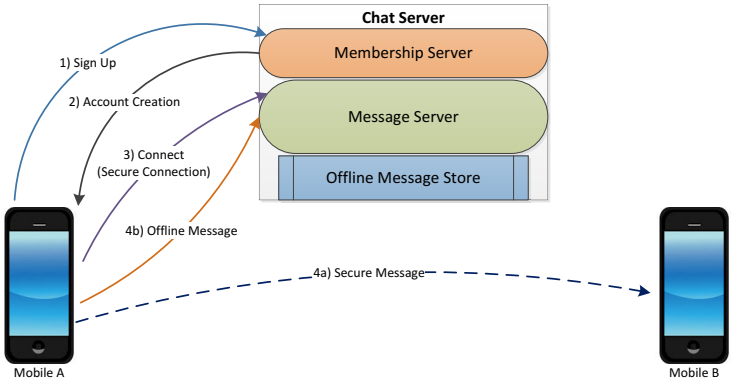

Fig. 1. Generic Architecture of a Mobile Chat Application

The communication link between the mobile application and the message server is protected using cryptography. In addition, the virtual communication link between the users of mobiles 'A' and 'B' is encrypted using the cryptographic keys generated and known only to the respective applications (of users 'A' and 'B').

The generic architecture shown in Figure 1 is essentially deployed by all of the mobile chat services discussed in section 2. However, the differentiator is the features deployed by the proposed architecture that are discussed in subsequent sections.

3.3 Signing Up

When a user completes the installation of an application and wants to create a new account or connect using existing credentials, they can initiate the sign-up process described below:

1. User selects either the new account option or an existing account.
   (a) If the user is an existing customer, then she will provide her account credentials (login/password)
   (b) If the user is a new customer, she will provide her email address[7] and provides a password (for mobile applications)
2. The server checks the provided information from the previous step, and instructs the application to generate local credentials. In addition, the server also generates a unique alphanumeric sequence that acts as the user identifier.
3. The mobile application generates a set of keys

[7] Email verification is carried out as part of the account creation process, in which an account only becomes active after the user clicks the account activation link sent to her (provided) email address.

(a) A signature key pair for TLS [14]
(b) A public key pair for encryption/decryption operations
(c) A symmetric storage key to encrypt/decrypt local storage that includes contact list, chat history and key store.

4. On the mobile application's request and verification by the chat server, it issues cryptographic certificates to both the signature key and the public key of the application

In subsequent communications between the mobile chat application and the chat server, a unique alphanumeric user ID and cryptographic certificate is used to authenticate the user/application and establish a TLS session (i.e. a two-way authentication-based SSL/TLS session). To add users to contact lists, their email address or unique identities can be used. When user 'A' makes a request to add user 'B' and she accepts the request, both users will then share their public keys along with associated certificates (issued by the chat server). The certificates provide the necessary guarantee that the received public key indeed belongs to the appropriate user.

3.4 Key Exchange

Keeping in mind the requirements listed in section 3.1, we have to design a key exchange process that can work even when the intended recipient(s) are offline. The requirement for an offline key exchange rules out any synchronous two-way key sharing protocols. The rationale behind having an offline key exchange is to avoid restricting a user to communicating only if a key is already shared. In addition, sharing a key only when both parties are online might not be a feasible proposition. Furthermore, in group chats all users would have to be online to share the key before they could start secure communication with each other. In this situation, if a single participant is offline then either she might not be able to read messages exchanged in the group chat or she has to be removed from the group if the chat session has to be established/continued.

Fig. 2. Key Share Message Structure

Therefore, we propose a scheme for key sharing that can accommodate such requirements. Each communicating party will generate a random key and encrypt it using the public key of the intended recipient. The encrypted message contains a number of elements shown in Figure 2. Timestamps [22] are included to avoid any potential replay attacks. Clock synchronisation between the communicating entities (users) is not required to use the timestamp, because when a mobile application connects to the chat server, it gets a time (server time) and uses it as an internal application. The timestamp included in the message is taken from

this internal application time and not from the device/user time. All applications will get their internal time synchronised with the chat server whenever they connect with it, thus removing the need to synchronise device clocks between users.

The key lifetime field gives a choice to the user/mobile-application to set a limit on key usage. The key lifetime field can configured to a session-based lifetime or to any arbitrary time (e.g. seven days). The cryptographic algo field communicates the preferred symmetric key algorithm that the sender would like the receiver to use when communicating with her. The cryptographic algorithms are chosen from a list of selected algorithms which are part of the mobile chat application. The four random numbers included in the message are to generate individual message keys that are discussed in detail in section 3.6. Finally, the last block contains the master key (symmetric key) that the sender requires the receiver to use during any future communications.

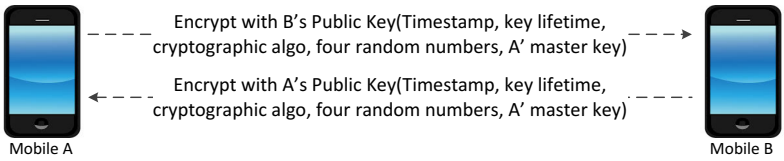

Fig. 3. Key Exchange Two Users

A point to note is that the key and cryptographic-algorithm choice shared by the sender communicates to the receiver that when decrypting any future messages from this sender, she should use them, as shown in Figure 3. Similarly, the receiver will also send the key share message shown in Figure 2 encrypted by the sender's public key. Therefore, both users will use their own generated keys to encrypt all their outgoing messages. The only difference in a group chat is that the chat organiser (group creator/administrator) will generate the master key and share it with all participants, who will then use this key to encrypt all their messages. This avoids using multiple keys (equal to the number of users in the group) to communicate with all users in the group.

3.5 Mutual User-to-User (U2U) Authentication

Mobile chat authenticates itself to the chat server, but U2U authentication is between the users themselves. This authentication process does not involve the chat server and the objective of the process is to assure communicating entities that they are talking with the right *person*. The U2U authentication process has two phases: the opt-in and the authentication phase.

In the opt-in phase, users agree on establishing a U2U authentication mechanism. To accomplish this, two users will initiate the U2U opt-in phase shown in Figure 4. The numeric items in the figure 4 relate to the process steps listed below:

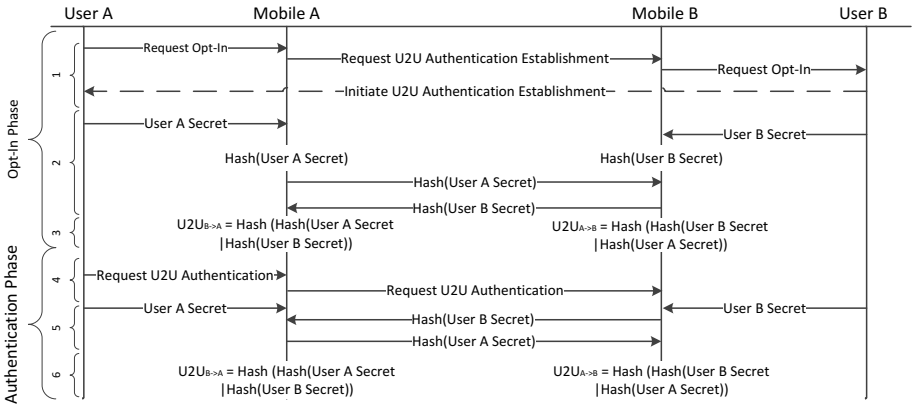

Fig. 4. Overview Of the U2U Authentication

1. User A requests establishment of a U2U authentication mechanism with user B. The request is communicated to user B: if she accepts the establishment of U2U authentication, her decision is communicated back to user A
2. Both users A and B provide their secrets (keyword) to their respective mobile chat applications that will generate hash values of the secrets. Chat applications communicate these hash values to each other.
3. In this step, the mobile applications of A and B generate U2U secrets (individually). The mobile application of A will generate its U2U secret by hashing the concatenation of the secret of A and the hash of B's secret. Similarly, the mobile application of B will generate its U2U secret by hashing the concatenation of the secret of B and the hash of A's secret. This means that both mobile chat applications have different U2U authentication values
4. Either of the users can request U2U authentication; however in figure 4, A initiates the authentication phase
5. Both A and B provide their secrets to their respective mobile chat applications
6. Individual applications will have these secrets and communicate them to each other
7. Applications will then generate the U2U secret and match with the stored value. If it matches then they can ascertain that the person with whom they are chatting is not an imposter

A point to note is that all the messages listed in Figure 4 are communicated confidentially, using the shared cryptographic keys. These messages are encrypted as if they were chat messages, which are discussed in detail in the next section. A point to note is that our mechanism is not comparable to the SafeSlinger[8] as the U2U authentication is no associated in any way with the key generation.

[8] Website: `https://www.cylab.cmu.edu/safeslinger/`

3.6 Message Communication

In this section, we discuss how individual messages are constructed and how the shared master key is used to generate message keys. The keys are then used to encrypt and decrypt the messages. Each message send by individual users is encrypted by a different key, generated using the shared master key and four random numbers (figure 2).

To generate individual message keys, we use the Pseudorandom Number Generator (PRNG) design from [23], illustrated in Figure 5. The shared (four) random numbers are taken as the seed file: for each iteration a random number (n) is encrypted using the shared master key. The output is used as the message key. The output is again encrypted using the shared master key, the output is XOR with n. The output of the XOR operation then replaces the value of n in the seed file.

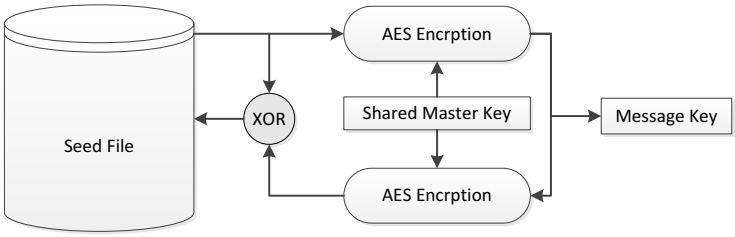

Fig. 5. Message Key Generation [23]

The message generated will then be used to encrypt the outgoing message. Similarly, each user will also have a second message key generator for messages they are receiving. For incoming and outgoing messages, each communicating entity will have two different shared master keys and seed files. Therefore, message keys for outgoing and incoming messages will be different.

3.7 Chat and Profile Storage

All data related to the application should be securely stored on the device. As chat histories are not stored on the server, users might need to have them on their devices. Therefore, chat histories, shared master keys, public key pairs, signature key pairs and contact lists should be stored on the device, protected with the user's Personal Identification Number (PIN) or password. For our proposal, we use a PIN-based mechanism with a short velocity limit (only three wrong tries permitted). If a user locks her applications, she can delete the application and then reinstall it again. If she provides her account credentials she can get her contact list back, but her application will generate new public and signature key pairs (revoking previous keys) along with establishing new master shared keys with her contacts.

In our proposal, the PIN is not the account credential that the user requires to connect with the chat server. The PIN is to authenticate the user to her mobile chat application and open her profile. The PIN is not communicated by the mobile chat application to the chat server and it is stored locally. To protect the PIN value, the application has to rely on the underlying platform and its security mechanism, which is beyond the scope of this paper.

4 Practical Implementation

Using the proposed architecture/features of the mobile chat application in the previous section, we detail the basic implementation carried out to provide a proof-of-concept for the proposed architecture. In this section, we discuss the practical implementation.

4.1 Technology Overview

As part of the design, we opted for using components that are publicly available and have the least license restrictions. For this reason, most of the features presented in the proposal are built using public libraries.

As shown in Figure 1, the deployment of a secure chat service requires the implementation of the chat server and the mobile application. For mobile application development, we choose the Android platform [24]. The chat server was hosted on the Intel[9] Core i7, 2.70 GHz and 8GB RAM machine running Ubuntu[10] 13.10.

In subsequent sections, we briefly discuss the implementation experience for both the chat server and the mobile application.

4.2 Server Side Implementation

On the chat server, we deployed two logical servers. One server ran XAMPP 1.8.3 as a membership server and Mosquitto 1.2.3 as a message server. The XAMPP server is an Apache [25] distribution containing MySQL [26], PHP [27], and Perl [28]. In addition to this, we also included Mcrypt [29] and OpenSSL [30] extensions.

Mosquitto [31] is an open source message server based on the MQ Telemetry Transport (MQTT) protocol [32]. The most recent release of Mosquitto supports the MQTT protocol version 3.1, which provides a very lightweight messaging architecture. We chose Mosquitto for its implementation of MQTT and the rationale behind the choice of MQTT is:

MQTT provided several useful features, such as "push notifications" (so that constant polling for new messages is not required by the Android-based chat application), assured message delivery and reliability, low battery usage, and

[9] Intel website: `www.intel.com`
[10] Ubuntu website: `www.ubuntu.com`

also "offline message delivery". Offline messages are stored on the message server until the recipient comes online, at which point these messages are sent to the recipient and removed from the server.

MQTT also provides smaller message sizes due to being a binary protocol, compared to other protocols like XMPP [33], which uses XML [34] for its messages. Using MQTT means that text messages are smaller in size than the same messages created with other message protocols such as XMPP (an example is the two character message ":)", which using MQTT generates 70 bytes, whereas the same message in XMPP is represented with 100 bytes). The low data usage for communicating the message is important from the point of view of both performance (delivery of messages) and bandwidth usage (i.e. mobile data packages).

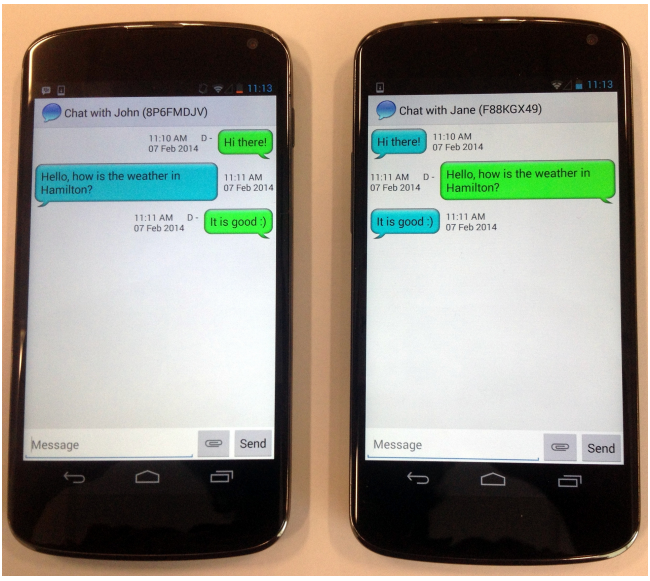

Fig. 6. Screen Shot of Mobile Chat Application Running on Two Android Devices

4.3 Mobile Side Implementation

For mobile chat, we developed an application supporting Android 4.1+. Additional APIs included GSON [35] for converting JSON, SQLCipher [36] for fully encrypted databases, Eclipse Paho [37] for MQTT messaging and Spongy Castle[11] [39] for cryptographic algorithms. The PIN is stored in the application/share preferences, and it unlocks the SQLCipher encryption/decryption key. Shared master keys, (optional) chat histories, U2U authentication secrets, and contact lists are stored in the SQLCipher database.

[11] Repackage of Bouncy Castle [38] for Android.

5 Overall Analysis

In section, we briefly analyse the architecture and implementation of our secure mobile chat service.

5.1 Analysis of the Proposed Architecture

In section 3.1, we list[12] seven basic requirements for a secure and privacy preserving chat service. Taking these seven requirements, we have provided a comparison between our proposal and commercially available products discussed in section 2. The comparison based on the listed requirements is shown in table 1.

Table 1. Comparison with Existing Chat Applications

Criteria	WhatsApp	BlackBerry Messenger	Wickr	Silent Text	Proposed Chat
Req1	-	*	*	*	*
Req2	-	-	(*)	(*)	*
Req3	*	*	*	*	*
Req4	-	-	-	-	*
Req5	-	-	-	-	*
Req6	*	*	*	*	*
Req7	-	*	*	*	*

Note: "*" means that it meets the requirement. "-" stands for either does not support the requirement or information is not publicly available. "(*)" means that the requirement is partially supported.

It is clear that our proposal meets all the requirements, and that one of the most widely-used mobile chat services, "WhatsApp" does not satisfy even half of the requirements. However, in support of WhatsApp we can argue that they do not market or claim to provide a secure and privacy-preserving chat service. Therefore, it is only natural that it does not meet the majority of the requirements.

5.2 Implementation Analysis

The objective of the implementation was to provide a proof-of-concept for constructing a secure and privacy-preserving mobile chat application with publicly available specifications. Therefore, this exercise was a study of the technical difficulties that a chat service provider might face in developing such a service. During our development process, we did not face any technical issues. In most cases, the important components required for such a proposed service are already present, apart from concerns about handling a large number of simultaneous connections.

[12] * We do not claim that this is an exhaustive list and it should be considered as a basic list of requirements.

We only tested the application for its features and whether it adequately supports the listed requirements (section 3.1). In addition, we did not test the scalability of the implementation of the chat server. However, with regard to message generation, the implementation was comparable to any commercially available mobile chat application. However, we cannot claim the same for the chat server as we did not simulate the load test to make it comparable to other mobile chat services. Such a test is beyond the scope of this work.

6 Conclusion and Future Research Directions

In this paper, we provided an open specification for a secure and privacy-preserving chat service. We described the basic requirements, architecture and implementation experience in deploying such a service. The aim of the paper is to develop mobile chat services and explore any potential complexities involved in such a service providing privacy protection to its customers. In this work, we explored the theoretical foundations and technical challenges faced if privacy protection is built into a chat service. We found that most of the theoretical and technical components are already available. With a few minor modifications, a strongly privacy-based chat service can be constructed. We have shown that a secure and privacy-preserving chat application is technically feasible. During the implementation of the framework, we did not face any serious issues concerning the technology or performance that might make this proposal infeasible. Whether it is a viable business is a different aspect of such a service, and was not considered in this paper.

In future research, we would like to experiment with the scalability and performance of the chat server: this might reveal some bottlenecks in building and maintaining a privacy-based chat server. Another potential aspect is investigation of how the text chat service proposed in this paper could be extended to a voice and video chat service. The challenges presented in providing a secure and privacy-preserving voice and/or video chat service might be more than those presented by a text-based chat service. This will give a much better insight into the development of secure and privacy-preserving services, their running costs and usability requirements, providing an opportunity to understand the underlying reasons why such services are not prevalent or widely adopted by customers.

Acknowledgements. The authors would like to thank the Faculty of Computing of Mathematical Sciences for the funds supporting this research, and acknowledge the implementation inputs of Corey Brown in this research.

References

1. Thomas, D., Bradshaw, T.: Rapid Rise of Chat Apps Slims Texting Cash Cow for Mobile Groups. Online. Financial Times (April 2013), http://www.ft.com/intl/cms/s/0/226ef82e-aed3-11e2-bdfd-00144feabdc0.html#axzz2urfG5LDi

2. Paczkowski, J.: WhatsApp: Bigger Than Twitter. Online. All Things D (April 2013), http://allthingsd.com/20130416/whatsapp-bigger-than-twitter/
3. reenwald, G.: English NSA Collecting Phone Record of Millions of Verizon Customers Daily. Online. The Guardian (June 2013), http://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order
4. Vincent, J.: Affiliations, Emotion and the Mobile Phone. In: Esposito, A., Vích, R. (eds.) Cross-Modal Analysis. LNCS (LNAI), vol. 5641, pp. 28–41. Springer, Heidelberg (2009)
5. Ling, R.: New Tech, New Ties: How Mobile Communication Is Reshaping Social Cohesion. The MIT Press (2008)
6. Laugesen, J., Yuan, Y.: What Factors Contributed to the Success of Apple's iPhone? In: Proceedings of the 2010 Ninth International Conference on Mobile Business / 2010 Ninth Global Mobility Roundtable ICMB-GMR 2010, pp. 91–99. IEEE Computer Society, Washington, DC (2010)
7. Akram, R.N., Markantonakis, K., Mayes, K.: Building the Bridges – A Proposal for Merging different Paradigms in Mobile NFC Ecosystem. In: Xie, S. (ed.) The 8th International Conference on Computational Intelligence and Security (CIS 2012). IEEE Computer Society, Guangzhou (2012)
8. Shabtai, A., Fledel, Y., Kanonov, U., Elovici, Y., Dolev, S., Glezer, C.: Google Android: A Comprehensive Security Assessment. IEEE Security and Privacy 8(2), 35–44 (2010)
9. Becher, M., Freiling, F.C., Hoffmann, J., Holz, T., Uellenbeck, S., Wolf, C.: Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices. In: 2011 IEEE Symposium on Security and Privacy (SP), pp. 96–111. IEEE (2011)
10. Goodin, D.: Crypto Weaknesses in WhatsApp "The Kind of Stuff the NSA would Love". Online. ARS Technica (February 2014), http://arstechnica.com/security/2014/02/crypto-weaknesses-in-whatsapp-the-kind-of-stuff-the-nsa-would-love/
11. The WhatsApp Architecture Facebook Bought for $19 Billion. Online. High Scalability, (February 2014) http://highscalability.com/blog/2014/2/26/the-whatsapp-architecture-facebook-bought-for-19-billion.html
12. Freier, A., Karlton, P., Kocher, P.: RFC:6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0. Online. IETF (August 2011)
13. Security of BlackBerry PIN-to-PIN Messaging. Online. Communications Security Establishment Canada, http://www.cse-cst.gc.ca/its-sti/publications/itsb-bsti/itsb57b-eng.html (March 2011)
14. Dierks, T., Rescorla, E.: RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2., Tech. Rep. (August 2008)
15. Moscaritolo, V., Belvin, G., Zimmermann, P.: Silent Circle Instant Messaging Protocol: Protocol Specification, Online, White Paper (December 2012)
16. Landman, M.: Managing Smart Phone Security Risks. In: 2010 Information Security Curriculum Development Conference, pp. 145–155. ACM (2010)
17. Felt, A.P., Egelman, S., Wagner, D.: I've Got 99 Problems, but Vibration ain't One: A Survey of Smartphone Users' Concerns. In: Proceedings of the 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 33–44. ACM (2012)
18. La Polla, M., Martinelli, F., Sgandurra, D.: A Survey on Security for Mobile Devices. IEEE Communications Surveys & Tutorials, 446–471 (2013)
19. Zimmermann, P., Johnston, A., Callas, J.: ZRTP: Media Path Key Agreement for Unicast Secure RTP. IETF, RFC 6189 (April 2011)

20. Alexander, C., Goldberg, I.: Improved User Authentication in Off-the-record Messaging. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society, WPES 2007, pp. 41–47. ACM, New York (2007)
21. Belvin, G.: A Secure Text Messaging Protocol. Cryptology ePrint Archive, Report 2014/036 (2014), http://eprint.iacr.org/
22. Dyreson, C.E., Snodgrass, R.T.: Timestamp semantics and representation. Information Systems 18(3), 143–166 (1993)
23. Akram, R.N., Markantonakis, K., Mayes, K.: Pseudorandom Number Generation in Smart Cards: An Implementation, Performance and Randomness Analysis. In: Mana, A., Klonowski, M. (eds.) 5th International Conference on New Technologies, Mobility and Security (NTMS). IEEE Computer Society, Turkey (2012)
24. Rogers, R., Lombardo, J., Mednieks, Z., Meike, B.: Android Application Development: Programming with the Google SDK. O'Reilly, Beijing (2009)
25. Apache, Apache Tomcat (May 2007) http://tomcat.apache.org/
26. MySQL 5.6 Reference Manual, Online, Manual (March 2014), http://downloads.mysql.com/docs/refman-5.6-en.pdf
27. Wenz, C., Hauser, T.: PHP 5.1. Markt Technik, München (2006)
28. Wall, L., et al.: The Perl Language Reference Manual (for Perl version 5.12.1.). 5th edn. Perl Reference Manual (for Perl version 5.12.1), vol. 1. Network Theory Ltd, United Kingdom (2010), http://www.network-theory.co.uk/docs/perlref/
29. PHP Cryptogrpahy Extensions: Mcrypt. Online PHP (November 2013), http://nz2.php.net/mcrypt
30. The OpenSSL Project, OpenSSL: The Open Source Toolkit for SSL/TLS (April 2003), http://www.openssl.org
31. Mosquitto: An Open Source MQTT v3.1/v3.1.1 Broker, http://mosquitto.org/
32. MQ Telemetry Transport (MQTT) Protocol, http://mqtt.org/
33. Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. Internet RFC 3921 (October 2004)
34. Bray, T.: Extensible Markup Language - SW (XML-SW). Tech. Rep. (February 2002), http://www.textuality.com/xml/xmlSW.html
35. Singh, I., Leitch, J., Wilson, J.: GSON User Guide, User Guide, https://sites.google.com/site/gson/gson-user-guide
36. SQLCipher Documentation, http://sqlcipher.net/documentation
37. Eclipse Paho Project, http://www.eclipse.org/paho/
38. Bouncy Castle Crypto Package. Bouncy Castle, http://www.bouncycastle.org/documentation.html
39. Spongy Castle, http://rtyley.github.io/spongycastle/