The Security Analysis of Popular Instant Messaging Applications

Lijun Zhang
Science and Technology on Communication Security Laboratory,
Chengdu, 610041, China
e-mail: 41049250@qq.com

Qingbing Ji, Fei Yu
Science and Technology on Communication Security Laboratory,
Chengdu, 610041, China
e-mail: achong731@sohu.com

Abstract—**Instant messaging applications on the smartphone platform have been prevalent in our communication with each other over the past few years. Meanwhile, people pay more and more attention to conversation security and personal privacy. In this paper, we present a comprehensive security analysis of three representative instant messagers including WeChat, WhatsApp and Telegram. We explicitly demonstrate their air interface communication protocols and the underlying security mechanisms. Furthermore, the encryption principles in local storage of chat records are also analyzed particularly. Besides, we exploit that personal metadata of communication participants will be collected by these instant messaging applications. These metadata closely relevant to personal privacy are almost disregarded in the previous research.**

Keywords-Instant messagers, protocol security analysis, storage encryption, metadata privacy

I. INTRODUCTION

Instant messaging application (also called instant messager and IM for its abbreviation) has become an indispensable communication tool with the rapid popularity of smartphones. Nowadays, there are more than ten familiar instant messagers and at least three of them are representative which are WeChat, WhatsApp and Telegram respectively. These three messagers have remarkable advantage at the aspect of usability and security so that they acquire a large many of users whose total number exceeds 1 billion. Besides the daily communication of ordinary people, many terrorists tend to employ these IM to contact with each other and machinate terroristic attacks since the messagers' encryption function provides the sufficient information security which keeps them from being monitored. The confirmed evidence shows that attackers in London used WhatsApp to send messages [1] while ISIS terrorists utilized Telegram to interchange opinions of attack arrangement [2]. So it has great significance to study the security principle of these instant messagers which is beneficial not only for the necessary forensics of terroristic or criminal activities but also for the better protection of users' personal privacy.

In order to understand the underlying security mechanism, many researchers studied the IM encryption of communication protocol and local storage from the digital forensic view. For the distinguished WhatsApp, Sahu [3] focus on conducting forensic data analysis by extracting WhatsApp's local records including messages, videos, audios and images in Android smartphone. But it only explained where to acquire these data though an existent tool without any further analysis. Anglano [4] performed a more comprehensive discussion of WhatsApp messages which provided how to decode and reconstruct the lists of contact with their exchanged information. Thakur [5] also studied the forensic analysis on Android phones and showed that it was able to obtain many relevant data such as phone numbers, various media files and locations. While Mahajan [6] completed the similar work but made use of the famous Cellebrite Forensic Extraction Device (UFED) toolkit. Besides these researches on mobile device forensic, people tried to exploit the network flow characters from its protocol. Hancke [7] provided some detail of WhatsApp network protocol but only concentrating on Realtime Transport Protocol media streams failing to uncover the call signaling messages. Karpisek [8] indicated that WhatsApp used the FunXMPP protocol for message exchange which is a binary efficient encoded Extensible Messaging and Presence Protocol (XMPP) [9]. FunXMPP protocol is also briefly described from an implementation perspective [10]. However, their results are not suitable for the WhatsApp updated version from March 2016 since when the main security protocol has been changed to Signal [11, 12] in which the essential authentication procedure before is totally substituted.

In additional to WhatsApp, the security of another two messagers called WeChat and Telegram also drew researchers' attention. Qu [13] and Wan [14] exploited the interaction protocol framework and encryption mode of WeChat, but could not be able to acquire the message content. Zhang [15] presented a comprehensive forensic analysis for WeChat various kinds of local message records. Jakobsen [16] made a practical cryptanalysis of the Telegram messaging protocol and found some design vulnerabilities.

Although the current research findings derive some useful forensic results, however, they seldom analyze the IM communication protocol completely. The potential difficulty lies on some reasons: one is that the entire protocols are usually a little complicated and the other more important reason is that they are private to researchers. In this paper, our contributions consist of the following three aspects:

(1) We first recover the private protocol of our target IM by reverse engineering and then analyze it further.

(2) We present a comprehensive security analysis of three popular IM applications including their air interface protocols and local storage mechanisms.

(3) We exploit the personal metadata of communication participants collected by these messagers. These metadata closely relevant to personal privacy are almost disregarded in research before.

II. THE AIR INTERFACE PROTOCOL AND ANALYSIS

We mainly concentrate on security part of communication protocol which specifically refers to key agreement and encryption algorithm. It is worth mentioning firstly that all these three IM applications employ cryptographic mechanism but the security model is different with each other. Concretely, we divided the security protocol into three components including security model, key agreement and key update. For IM communication, there are two kinds of security model, one is Client-Server encryption and the other is End-to-End encryption. In the following section, we will briefly describe the security protocol of three IM applications and then analyze its key agreement and update features.

A. The Security Protocol of WeChat

Recently, the WeChat operator released an official version of WeChat's security protocol which is intended to be applied in future [17]. However, we find that the current version has not yet transformed to this announced protocol. After a laborious reverse engineering of WeChat client and combining with its communication network flow, we are successful to recover the current protocol (client version 8.5 till this paper is written).

Security mode: Client-Server encryption only.

Key agreement: the whole process is executed by several steps as shown in Fig. 1.

(1) Both client and server generate key parameters of Elliptic Curve Diffie-Hellman key exchange which are exactly public and private pairs. The pair is (Pub_Client, Pri_Client) on client end while on server end it is (Pub_Server, Pri_Server).

(2)The client generate AES first encryption key "Aes_Key" for login data upload which contains device identity, device type, IMEI number and so on, then encrypt the key parameters including "Pub_Client" and "Aes_Key" by RSA algorithm with encryption key "Rsa_Pub" and encrypt login data by AES algorithm with encryption key "Aes_Key". The client sends both ciphertexts to the server.

(3) The server decrypt the RSA ciphertext using the RSA private key "Rsa_Pri" to get "Aes_Key" and then decrypt AES ciphertext to get client's login data.

(4) The server generate a session key parameter "Session_para" and combine the key "Pub_Server" to form a response data, then encrypt it by AES algorithm with encryption key "Aes_Key".

(5) The client decrypt this AES ciphertext to get "Session_para" and "Pub_Server".

(6) Both client and server employ the receiving key parameter to derive session key "Session_key" for subsequent user's chat data.

Key update: The session key of WeChat will not update unless the running state of client changes. For example, the client state change from online to offline, when this client login online again, then the key agreement will be executed once more and the session key is updated. Of course, if the client software is reinstalled then the session key is definitely updated.

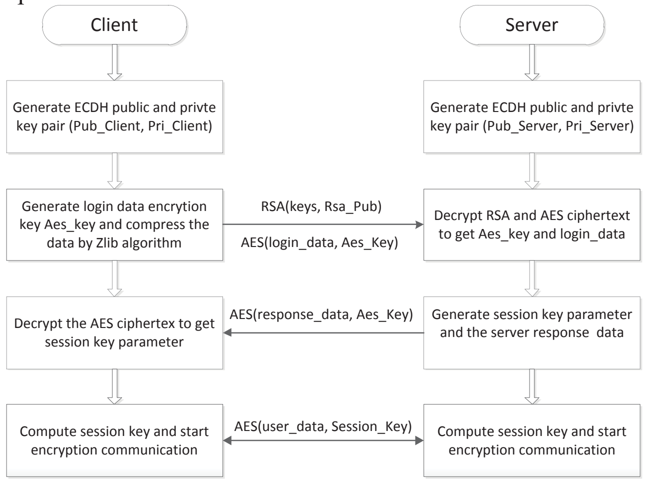
Client
Server
Generate ECDH public and privte key pair (Pub_Client, Pri_Client)
Generate ECDH public and privte key pair (Pub_Server, Pri_Server)
Generate login data encrytion key Aes_key and compress the data by Zlib algorithm
RSA(keys, Rsa_Pub)
AES(login_data, Aes_Key)
Decrypt RSA and AES ciphertext to get Aes_key and login_data
Decrypt the AES ciphertex to get session key parameter
AES(response_data, Aes_Key)
Generate session key parameter and the server response data
Compute session key and start encryption communication
AES(user_data, Session_Key)
Compute session key and start encryption communication

Figure 1: key agreementprotocol of WeChat

B. The Security Protocol of WhatsApp

The protocol of WhatsApp has been modified in a large scale since April 2016 when its operator Facebook issued a technical white paper called "WhatsApp Encryption Overview" to announce that the security protocol has totally transformed to Signal protocol. The highlight of this protocol is its support of end-to-end encryption which is designed to prevent third parties and WhatsApp server from having plaintext access to messages or calls.

Security mode: End-to-End encryption only.

Key agreement: as is described in this white paper, the WhatsApp server just plays a role of assisting key agreement for both ends without ability of decrypting messages as in Fig. 2.

(1) Once a client software is installed and ready to register, the client transmits its public identity key "ID_PubKey", public signed pre key "Signed_PubKey", and a batch of public One-Time pre keys "OneTime_PubKeys" to the server .The server stores these public keys associated with the user's identifier.

(2) When a client (client A) initiates a session, it requests these public keys of the target client (client B) and the server return these values.

(3) Client A generates an ephemeral Curve25519 key pair EA=(A_Ep_Pub, A_Ep_Pri) and calculates a master secret as master_secret= ECDH(IA, SB) || ECDH(EA, IB) || ECDH(EA, SB) || ECDH(EA, OB) where IA and IB are Identity key of A and B, SB is signed pre key of B as well as OB is an One-Time pre keys of B. Note that in this equation, the keys of A are private keys while the keys of B are public keys.

(4) Client A uses HKDF to create a Root Key and Chain Keys from the master_secret. Then derive the message key from chain key as mess_ key = HMAC-SHA256(chain key, 0x01).

(5) Client A send the message containing the public key of EA and IA as well as ciphertext encrypted using AES algorithm with key "mess_key".

(6) Client B calculates the same message key and decrypt to get plaintext message.

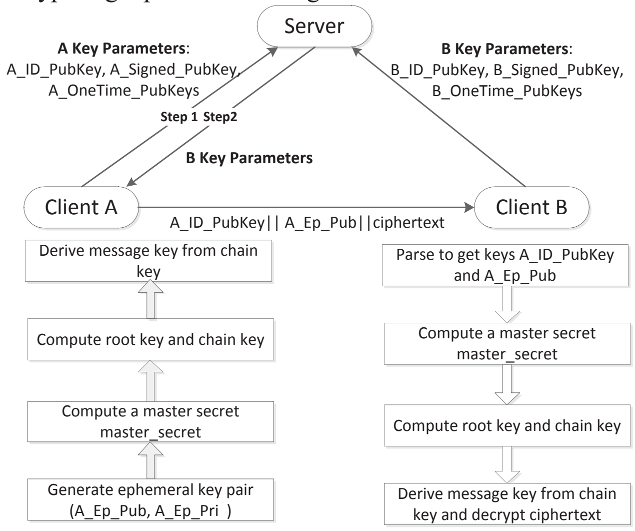
Server
A Key Parameters:
A_ID_PubKey, A_Signed_PubKey, A_OneTime_PubKeys
B Key Parameters:
B_ID_PubKey, B_Signed_PubKey, B_OneTime_PubKeys
Step 1
Step2
B Key Parameters
Client A
Client B
A_ID_PubKey|| A_Ep_Pub||ciphertext
Derive message key from chain key
Compute root key and chain key
Compute a master secret master_secret
Generate ephemeral key pair (A_Ep_Pub, A_Ep_Pri)
Parse to get keys A_ID_PubKey and A_Ep_Pub
Compute a master secret master_secret
Compute root key and chain key
Derive message key from chain key and decrypt ciphertext

Figure 2: the security protocol of WhatsApp

Key update: WhatsApp introduced the concept of double ratchet algorithm in order to reach the forward security. The double ratchet consists of symmetric-key ratchet and Diffie-Hellman ratchet.

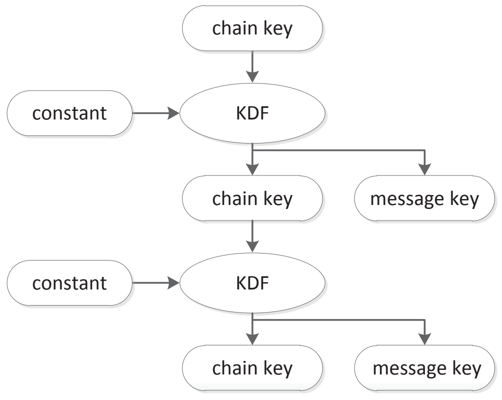

Figure 3: symmetric key ratchet of WhatsApp

The former ratchet is responsible for updating chain key and message key which assure that every message is encrypted by the different key and acquire forward security. The principle of this ratchet is shown in Fig. 3. The essential part is a key derivation function (KDF) which can be instantiated like HMAC-SHA256 function.

Besides the symmetric-key ratchet, WhatsApp also considered the case that if an attacker steals one party's sending and receiving chain keys, the attacker can compute all future message keys and decrypt all future messages. To prevent this, the double ratchet is proposed to update chain keys based on Diffie-Hellman outputs as shown in Fig. 4. To implement this ratchet, each party generates a DH key pair (a Diffie-Hellman public key and private key) which becomes their current ratchet key pair. Every message from either party begins with a header which contains the sender's current ratchet public key. When a new ratchet public key is received from the remote party, a DH ratchet step is performed which replaces the local party's current ratchet key pair with a new key pair. This working behavior enables that the parties take turns replacing ratchet key pairs. An eavesdropper who briefly compromises one of the parties might learn the value of a current ratchet private key, but that private key will eventually be replaced with an uncompromised one. At that point, the Diffie-Hellman calculation between ratchet key pairs will define a DH output unknown to the attacker. This mechanism will alleviate the endangerment of private key leakage to the greatest extent.

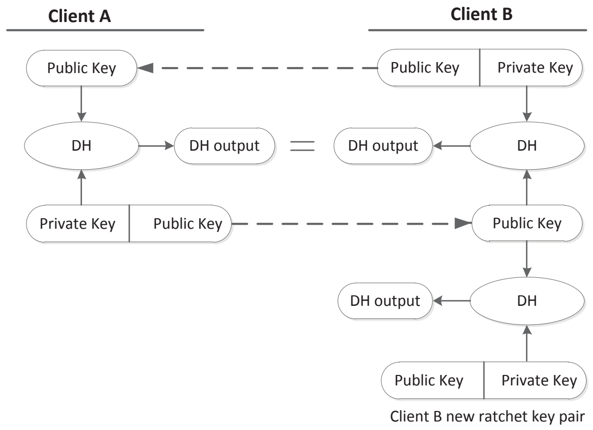

Figure 4: Diffie-Hellman ratchet of WhatsApp

C. *The Security Protocol of Telegram*

Telegram is an instant messaging service with open source designed by a Russian team. Instead of using proven cryptographic constructions, Telegram chose to create its own original protocol known as MTProto. Although its primitives like RSA and AES are secure, the constructing way of them in this protocol is controversial.

Security mode: both Client-Server encryption and End-to-End encryption are supported.

Key agreement: For the Client-Server encryption mode, the key agreement is executed in the following steps as shown in Fig. 5.

(1) A client sends a random number "nonce" to server.

(2) The server responses with another random number"s_nonce", a composite number "n" and the fingerprint of a public RSA key (fingerprint is actually a truncated SHA-1 value).

(3) The client decomposes n into primes p and q and pick out the RSA public key "server_pub" from locally stored public keys by the matched fingerprint.

(4) The client chooses a new random number "new_nonce" and encrypts the payload of all nonces, numbers n, p and q with encryption key "server_pub".

(5) The server responses payload of the Diffie-Hellman (DH) parameters g, a large prime "DH_p" and DH public

key $g_a=g^a$ mod DH_p. This payload is encrypted by AES algorithm with key and IV value derived from "s_nonce" and "new_nonce".

(6) The client generates DH public key $g_b=g^b$ mod DH_p and sends g_b back to server. Now the long term key between this client and server is "auth_key"$=(g_a)^b=(g_b)^a$ mod DH_p. Every message key will be derived from this long term key.

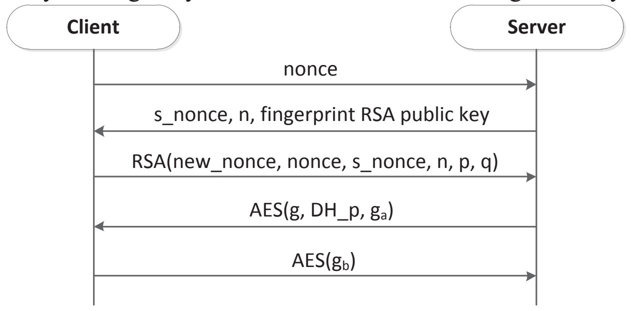
Client
Server
nonce
s_nonce, n, fingerprint RSA public key
RSA(new_nonce, nonce, s_nonce, n, p, q)
AES(g, DH_p, g_a)
AES(g_b)

Figure 5: Telegram key agreement in Client-Server mode

For the End-to-End encryption, the key agreement process is simpler as shown in Fig. 6.

(1) Client A contacts the server to require DH parameters including "DH_p" and generator g.

(2) The server returns these parameters and salt to help client A generate private key "a".

(3) Client B also performs the same as client A and get these parameters.

(4) Both clients send each other the DH public key and complete the key agreement which derives the shared long term key "auth_key"$=(g_a)^b=(g_b)^a$ mod DH_p.

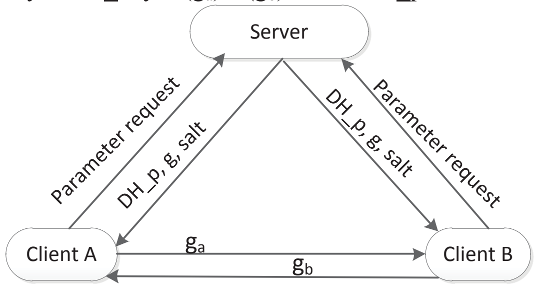

Figure 6: Telegram key agreement in End-to-End mode

Key update: Telegram adopted a relatively simple key update mechanism. First, every message key is different with each other since the SHA-1 digest is enrolled in message key derivation function. For the long term key "auth_key", Telegram introduced a counter to keep track of the number of times an auth-key is used to derive an AES key for encryption or decryption. If this counter reaches 100 or the secret key has been in use for more than one week, a new auth-key will be exchanged and the old one destroyed.

D. The Security Analysis of IM Applications

After acareful examination of these three IM security protocols, we derive the following results.

The design of WeChat protocol conforms to the well-recognized integration method of public and symmetric cryptographic algorithms. The key agreement protocol is secure enough while the key update mechanism is not considerate. If the network state is good, then the message encryption key will keep the same for at least several days. So once this key is compromised, an attacker could decrypt the messages for a relatively long time. Furthermore , the message key derivation design is too simple. All the messages are encrypted by the same message key which is distinctly weaker than One-Time-Pad key encryption mode whose message key varies after every message is transmitted.

The current protocol of WhatsApp employs the strong cryptographic primitives and standard interactive process of key agreement. Especially, it proposes the double ratchet mechanism which enables the One-Time-Pad message encryption with ephemeral message key and the frequent Diffie-Hellman key exchange. These operations provide the forward security and alleviate the damage of key leakage on some time.

The protocol of Telegram is designed by its own operation team without any security proof and formal analysis. Although they make use of strong primitives, however, it is surprising that the upper Diffie-Hellman key exchange uses the original version which is vulnerable to man-in-the-middle attack. The encryption mode of AES algorithm also has some problem. The Infinite Garble Extension (IGE) mode was the first attempt at an "authenticating encryption mode" originally for Kerberos. But it was a failed attempt since it does not provide integrity protection. Moreover, in the registration, the usage of RSA appears to be some amateur mistakes that the decomposition of a 64-bit composition "n" is weird and brings no additional security benefits.

III. THE SECURITY ANALYSIS OF LOCAL ENVIRONMENT

WeChat database SQLite makes use of an open source encryption tool called "sqlcipher" to encrypt the entire database file. The encryption algorithm is AES. In the realization of this algorithm, it chooses the operation mode of cipher block chaining (CBC mode) and key of 256 bits length where the initialization value (IV) can be extracted from the database's ciphertext.

Furthermore, we find out that the encryption key is derived from a password with length of 7 bytes by using key derivation function "PBKDF2" [18]. While this password can be calculated by MD5 hash function from the parameters of phone's IMEI (international mobile equipment identity number) and UIN (user information number) belonging to the encrypted database where IMEI is usually a string of numbers with length 14 or 15 and UIN is a string of numbers with length 9 or 10. Note that UIN of every database file uniquely corresponds to a WeChat's user ID number. More concretely, let Hash=MD5(IMEI||UIN), then password is truncated the first 7 characters of "Hash", i.e., password=Hash[0-6].

As a comparison, WhatsApp also uses SQLite database to store chat records which is also encrypted entirely in a more straightforward way. According to different application software versions, there are several types of databases like "msgstore.db.crypt8" or "msgstore.db.crypt12". The encryption algorithm is AES with CBC operation mode and key of 256-bit length. This key and IV value needed in this encryption are stored in a file "key" located in the path "/data/data/files/key". However, note that it is not able to exact this file if the device is not root.

It is worth mentioning that Telegram has an exciting feature which could enable the message sender to erase the chat record in a configured time when this message is sent. This is called "self-destruct" technique which is expected to leave no trails of chat records.

IV. The Privacy Metadata Collection Analysis

In contrast with the concern of personal conversation content security, people usually do not care about the metadata collection by their IM App. We first want to know what metadata is and why it also matters. The precise definition of metadata may be blurry but it is easily to be understood. For example, WhatsApp cannot know the call's content but could collect the identity of both parties, the conversation initialized time and spent time. In fact, the IM App can collect much more user's metadata including your phone model, operating system, contacts, location and so on. After a professional analysis and data mining, one can acquire a great deal of information. So at least we realize that even using end-to-end encryption does not prevent messaging services from collecting metadata. We list the metadata collection by these three popular IM services in Table Ⅰ where the symbol "✓" means conformability while "?" means "to be determined".

In table 1, device information contains device type, device identifier, operating system and version, and device settings. chat information means the related participating parties, conversation starting and end time as well as chat type. Now the IM application may be linked to other social network platform which would share the user's metadata. For example, WeChat is linked with QQ while WhatsApp is linked with Facebook. These relations will make metadata collection more convenient and extend the collection scope.

TABLE I. METADATA COLLECTION FROM IM SERVICE

IM APP	Metadata types			
	Device Info.	Contact	Location	Chat Info.
WeChat	✓	✓	✓	✓
WhatsApp	✓	✓	✓	✓
Telegram	?	?	✓	✓

V. Conclusion

This paper analyzed the security of air interface protocol and local storage encryption with respect to three representative instant messagers. We recovered the private protocol of WeChat messager and found out the severe design vulnerability of Telegram protocol. We also presented the analysis of privacy metadata collection which was almost neglected by IM users before. We expect that our security analysis results of IM protocol will be beneficial to network security researchers as well as protocol designers.

References

[1] Huffpost, London Attacker Sent Encrypted WhatsApp Message Before Rampage, URL, http://m.huffingtonpost.ca/news/whatsapp-london-attack/, 2017.3.

[2] Pamela Engel, Inside the app that's become ISIS's biggest propaganda machine, URL, http://www.businessinsider.com/telegram-isis-app-encrypted-propaganda-2015-11, 2015, 11.

[3] Sahu S. An Analysis of WhatsApp Forensic in Andorid Smartphones, International Journal of Engineering Research, Vol.3 No.5, 2014: 349-350.

[4] Thakur NS. Forensic analysis of WhatsApp on android smartphones (Master's thesis). University of New Orleans; 2013. URL, http://scholarworks.uno.edu/td/1706/

[5] Mahajan A, Dahiya M, Sanghvi H. Forensic analysis of instant messenger applications on android devices. 2013.arXiv preprint arXiv:1304.4915,URL,http://arxiv.org/abs/1304.4915

[6] Anglano C. Forensic analysis of whatsapp messenger on android smartphones. Digit Investigation 2014;11:201-213.

[7] Hancke P. Whatsapp exposed: Investigative report. 2015. URL, https://webrtchacks.com/wpcontent/uploads/2015/04/WhatsappReport.pdf

[8] Karpisek F, et al., WhatsApp network forensics: Decrypting and understanding the WhatsApp call signaling messages, Digital Investigation , 2015, http://dx.doi.org/10.1016/j.diin.2015.09.002

[9] WHAnonymous. Funxmpp-protocol. URL, https://github.com/WHAnonymous/Chat-API/wiki/ FunXMPP-Protocol, 2015.

[10] LowLevel-Studios. Whatsapp protocol 1.2: a brief explanation. URL, http://lowlevel–studios.com/whatsapp-protocol-1-2-a-briefexplanati-on/, 2012.

[11] libsignal-protocol-java. GitHub repository. URL, github.com/WhisperSystems/libsignal-protocol- java, 2016

[12] Moxie Marlinspike. Advanced cryptographic ratcheting. Blog. URL, https://whispersystems.org/ blog/advanced-ratcheting/, 2013.

[13] Qu X, Xue Z. The analysis and research of microletter encryption, Microcomputer Applications, 2014, 5 (1): 13-16.

[14] Wan Y, Gu Y, Qiu W. Research on Interactive Protocol and Encryption Mode of Wechat, Microcomputer Applications, 2015 (31), No.2, 31-34.

[15] Zhang L, Yu F, Ji Q. The Forensic Analysisof WeChat Message, The Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC2016), 2016, pp.500-503.

[16] Jakobsen J.B. A practical cryptanalysis of the Telegram messaging protocol, Master's thesis, Aarhus University, 2015.9.

[17] WeMobileDev, The introduction to WeChat secure communication protocol based on TLS 1.3 version, URL, http://www.qcloud.com/ community/article/134506

[18] Kaliski B. Password Based Cryptography Specification Version2.0. URL, http://toolsiet.org/ html/rfc2898, 2000-09.