Forensic analysis of the backup database file in KakaoTalk messenger

Jusop Choi*, Jaewoo Park† and Hyoungshick Kim†
*Department of Computer Science and Engineering, Sungkyunkwan University, Republic of Korea
Email : cjs1992@skku.edu
†Department of Software, Sungkyunkwan University, Republic of Korea
Email : {bluereaper, hyoung}@skku.edu

***Abstract*—Instant messaging services should be designed to securely protect their users' personal contents such as chat messages, photos and video clips against a wide range of attacks. In general, such contents are securely encrypted in storage. In this paper, however, we demonstrated that the backup database file for chat messages in KakaoTalk (the most popularly used instant messaging service in Republic of Korea, http://www.kakao.com/talk/en) can be leaked to unauthorized users. We carefully examined the backup procedure in KakaoTalk through reverse engineering the KakaoTalk application to analyze how the backup database file was encrypted and the encryption key can be generated. Our analysis showed that the encrypted database is susceptible to off-line password guessing attacks. Based on this finding, we recommend that a secure key generation technique should be designed to improve resistance against offline password guessing attacks by using a random secret number to generate the encryption key.**

***Index Terms*—KakaoTalk, reverse-engineering, off-line password guessing, database encryption, key generation**

I. Introduction

Instant Messaging (IM) services such as AIM, MSN Messenger, Google Talk and WhatsApp revolutionised the way people communicate with each other. Unlike email, IM services allow users keep in touch with other users in a more interactive way. Also, users are capable of sending offline messages to other users. Therefore, unlike phone calls, an IM user can easily read missed messages and previous messages.

However, as IM services have become the most popularly used communication tool, many users seem to be concerned for their privacy in using those services since their personal information such as private chats, photos, contacts and user profiles could be exposed [3]. For better user experiences, user's personal information is typically stored in files on the user's device (e.g., PC or smartphone) running the IM application. As a consequence, such information would become one of the most attractive targets for attackers (e.g., intelligence agencies, police investigators, cyber-criminals, etc.). For example, the information about friends or contacts could be used for a variety of cyber criminal activities such as spam, phishing and rogue accounts [12]. Furthermore, law enforcement agencies have often tried to seek personal data of a suspect for criminal investigation [9], which may result in an invasion of users' privacy. Therefore, it is critical to protect the user's personal data stored in files on the device from unauthorized attackers.

A security practice commonly used in IM applications is to encrypt such sensitive files with a key that the IM application can only access. Needless to say, the security of this security practice strongly relies on the protection of the encryption key. Our research was motivated to check whether the protection of the encryption key in IM applications would actually be acceptable to the standard criteria in the information security community.

In order to provide practical answers to this research question, we examined the key protection implementation in IM applications through a case study of KakaoTalk (http://www.kakao.com/talk/en) that is the most popularly used IM application in South Korea with more than 49.1 million active users [8]. Similar to other IM applications, KakaoTalk is also trying to protect sensitive files such as KakaoTalk user's chat database by encrypting them with a secret key that is not exposed to anyone except the user's KakaoTalk application. We are particularly interested in the backup feature for the chat database, which was recently introduced in version 2.0.7.914. According to the KakaoTalk's announcement, the chat messages are stored in the KakaoTalk's sever for only three days. Thus, the backup feature seems necessary for users who want to store their previous chat messages over a long period of time. We found that the chat database (used for backup) was encrypted. To obtain the encryption key used for the backup database file, we analyzed all the steps of the backup procedure in KakaoTalk through reverse engineering the KakaoTalk application running on Windows 7. Our key contributions are summarized as follows:

- We presented the detailed forensic analysis of the KakaoTalk application on PC devices to explore the structure of the chat database used for the backup feature.
- From the reverse engineering analysis for the backup procedure in the KakaoTalk application, we found how a password derived key was generated for encrypting the backup database file, revealing that the encrypted database can be leaked to unauthorized users trying to perform offline password guessing attacks.
- We discussed the best security practices to protect encrypted databases in IM applications against offline password guessing attacks.

The rest of the paper is organized as follows. In Section II,

we provide the results of previous studies for the forensic analysis of IM applications and the background information about KakaoTalk. In Section III, we describe the experiment environments for our analysis. In Section IV, we introduce a standard process to obtain the information about databases stored in IM applications. In Section V, we present our main results from the detailed analysis of the encrypted database in the KakaoTalk IM applications. In Section VI, we describe a possible scenario for forensic investigators. In Section VII, we discuss countermeasures to mitigate the risk of key exposure. In Section VIII, we mention the ethical and legal implications of this work. Finally, in Section IX, we conclude by summarizing the main results of this work.

II. Background

In this section, we first summarize the main results of previous studies that analyzed databases used for IM applications and then briefly introduce the KakaoTalk IM application that is needed to understand our work well.

A. Forensic analyses in IM applications

Forensic analysis of popular IM applications such as WhatsApp (https://web.whatsapp.com/), Viber (http://www.viber.com/) and WeChat (https://web.wechat.com/) have attracted considerable attention in the field of information security.

For WhatsApp and Viber, Mahajan and Sanghvi [1] used a UFED (Universal Forensic Extraction Device) physical analyzer to extract user's personal data (e.g., chat messages, contact list, profile picture and image) from an Android device's internal memory. In WhatsApp, chat message artifacts, timestamps and names of the exchanged files sent and received were extracted, however the storage locations of those files were not found without manual analysis. Moreover, they failed to obtain any artifact with the UFED physical analyzer. Chat messages and phone call history data were obtained through intensive manual analysis of code. For WhatsApp, Thakur [16] extended their work to determine which user data can be extracted from the non-volatile external storage and internal memory of an Android device. Thakur particularly considered how to extract the deleted messages from an Android device's internal memory. Gao and Zhang [10] analyzed database files in WeChat and found the file extensions and locations of databases, the structures for chat databases.

Recently, Anglano [4] also presented the forensic analysis of the artifacts left on Android devices by WhatsApp messenger, revealing the structures (e.g., tables and fields) of databases used in WhatsApp. Unlike existing studies, however, Anglano introduced a new approach to decode and interpret the obtained artifacts from the WhatsApp messenger on Android devices.

Encryption is a common security practice used in IM applications because several important personal data such as chat messages and user profile are stored and managed for many different purposes. According to the study [3], users were concerned about chat message logs that could be abused. However, there have been few studies focusing on the analysis of encrypted databases. Barghouthi and Said [2] analyzed several IM applications (Yahoo, Google Talk, Skype, Facebook, and Gmail) to identify the encryption methods used for protecting their chat messages through the Wireshark packet sniffer and forensics investigating tools. In this paper, we presented a forensic analysis of KakaoTalk by focusing on the encrypted chat database for the backup feature in KakaoTalk. We were particularly interested in the encryption key generation procedure to determine whether user data can also be extracted from the encrypted database.

B. What is KakaoTalk?

KakaoTalk is the most popularly used IM service in South Korea [6] with more 49.1 million active users [8]. Figure 1 shows an example of chat messages. KakaoTalk has many useful features. For example, a user can send multimedia data to his or her friends. Advanced services such as group chat and VoIP are also supported. The KakaoTalk service was originally developed as a mobile application (similar to WhatsApp) for smartphones such as Android and iOS devices, but the PC and Mac versions of KakaoTalk applications were also recently introduced.

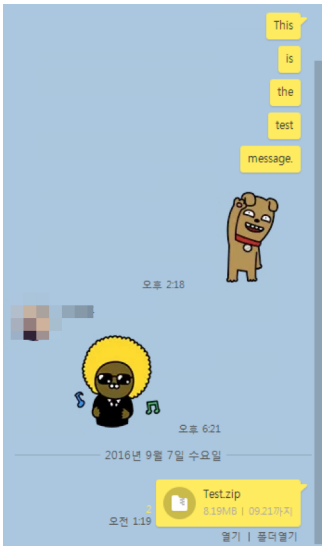

Fig. 1. Example of the chat messages.

Previous studies on KakaoTalk have found that KakaoTalk could be vulnerable to some attacks [12], [14]. Kim et al. [12] demonstrated that the automatic friend registration service provided by KakaoTalk could be abused for enumeration attacks that are used to collect users' personal data. Park and Kim [14] demonstrated that KakaoTalk users' activities could be inferred with a high probability from the captured network packets even when those packets were securely encrypted using a network protocol like SSL/TLS. However, their security analysis was mainly focused on user privacy issues from looking at some features (e.g., automatic friend registration) or a traffic analysis. To the best of our knowledge, there is

no previous work that has actually analyzed the security of encrypted databases.

KakaoTalk recently introduced the backup feature to store the database for chat messages. Similar to other IM applications, KakaoTalk is also trying to protect this database by encrypting it with a key that is not exposed to anyone except the KakaoTalk application. Our study focuses on the security analysis of the implementation for the key protection.

III. Experimental Environment

KakaoTalk version 2.0.6.854 and below versions do not support the encrypted backup; users can export the chat database into some readable plain-text format. The backup feature based on *encryption* was introduced in 2015 (since version 2.0.7.914); the chat databases are encrypted with a password-based encrypted key and can be shared between multiple devices by entering the password used for encryption.

We found that a sophisticated packer called `Themida`[1] was used to protect the KakaoTalk program from debuggers. `Themida` could be used to protect a program with various features such as detecting debuggers/dumpers/virtual machines/monitor blocker, obfuscating entry points, encrypting resources and so on. To bypass anti-debugging and monitor blocker features, we used `Ollydbg version 1.10`[2] with the `Phantom`[3] plug-in. We also used several `Ollydbg` plug-ins to easily identify the code used for encrypting chat databases. To search the locations of a specific value in the memory, `CheatEngine`[4] was used. We particularly analyzed the KakaoTalk version 2.1.2.1124 running on Windows 7.

IV. Our Analysis Method

To successfully obtain the plaintext chat database, we performed a forensic analysis of the KakaoTalk application at PC using the following process:

1) We first tried to find the backup database file. This process could be automatically implemented because the backup database file has the specific file names such as "`chatLogs_backup_[creation date]_[creation time]`" and file extension (i.e., `.edb`) by default in KakaoTalk application.
2) We checked whether the backup database file was encrypted or not, and then conjectured that the database file was encrypted because the file contains random text data that cannot be read by text editors or database viewers.
3) In order to determine the exact point in time at which to perform decryption on the encrypted backup database file, we searched for that file contents loaded from memory.
4) We found that AES-128 in CBC mode was used to encrypt the backup database file by examining the assembly code (see Figure 2) and the memory contents (e.g., Sbox) for the decryption routine.
5) In addition, we carefully examined the key generation procedure to analyze how the encryption key and parameters have been generated.

```
0073B4DB  . 8B4D 10       MOV ECX,DWORD PTR SS:[EBP+0x10]
0073B4DE  . 8B71 04       MOV ESI,DWORD PTR DS:[ECX+0x4]
0073B4E1  . 8B51 08       MOV EDX,DWORD PTR DS:[ECX+0x8]
0073B4E4  . 3350 08       XOR EDX,DWORD PTR DS:[EAX+0x8]
0073B4E7  . 3370 04       XOR ESI,DWORD PTR DS:[EAX+0x4]
0073B4EA  . 57            PUSH EDI
0073B4EB  . 8B39          MOV EDI,DWORD PTR DS:[ECX]
0073B4ED  . 3338          XOR EDI,DWORD PTR DS:[EAX]
0073B4EF  . 894D E4       MOV DWORD PTR SS:[EBP-0x1C],ECX
0073B4F2  . 8B49 0C       MOV ECX,DWORD PTR DS:[ECX+0xC]
0073B4F5  . 3348 0C       XOR ECX,DWORD PTR DS:[EAX+0xC]
```

Fig. 2. AES-128 in CBC mode procedure in the KakaoTalk messenger.

Our analysis results are described in detail in Section V.

V. Analysis Results

Our analysis results were divided into two steps. We first presented the key generation procedure for encrypting the backup database file and then explained the structure of that database.

A. Key generation procedure

The procedures for generating the encryption key and initial vector (IV) were as follows (see Figure 3):

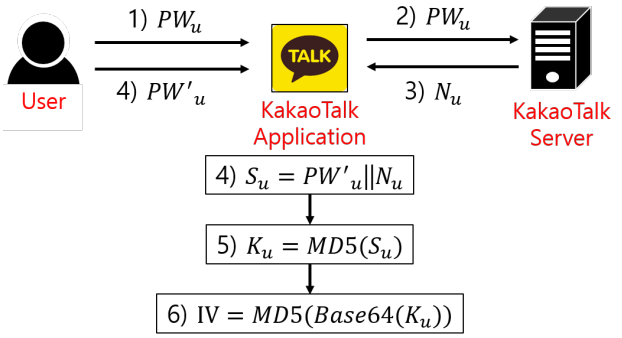

Fig. 3. Key generation procedure for encrypting the backup database.

1) To use the KakaoTalk application, a user u types his or her password PW_u into the password field in the KakaoTalk application.
2) The KakaoTalk application submits the login request with PW_u to the KakaoTalk server.
3) If PW_u is valid, the KakaoTalk server accepts the login request and replies to the KakaoTalk application by sending the specific number N_u which is uniquely assigned to user u of the KakaoTalk application.
4) To additionally restore the backup database file, another password PW'_u, which was previously used for creating that backup database file, is required. We note that PW'_u can be different from PW_u. The KakaoTalk application generates a string S_u by concatenating PW'_u with N_u received from the KakaoTalk server. The string S_u is

[1] http://www.oreans.com/themida.php
[2] http://www.ollydbg.de/
[3] https://tuts4you.com/download.php?view.1276
[4] http://www.cheatengine.org/

TABLE I
TABLES IN THE BACKUP DATABASE.

Table Name	Description
backupInfo	This table contains the creation date and the version of the backup database.
chatAttachmentFileInfo_v2	This table contains attachment file names and their attributes such as valid download period and download URL.
chatLogs	This table contains all chat activities performed in a chatting room. Each chat activity was recorded with the message itself, its creation time and message senders.
chatMembers	This table contains the information about users who are participating in the chatting.
chatMetas	This table contains the information about notices for each chat room.
chatRoomList	This table contains the information (the number of participants, the most recently delivered message and the notification alerts setting) about each chat room.

repeatedly concatenated until the length of S_u is greater than or equal to 512 bytes. If S_u is greater than 512 bytes, S_u must be truncated to 512 bytes. For example, when PW'_u = "`abcd1234`" and N_u = `1000`, S_u finally becomes "`abcd12341000abcd12341000` $\cdots$ `abcd1234`".

5) The encryption key K_u is generated by calculating the MD5 hash [15] of the string S_u.
6) To obtain the IV used for the encryption algorithm, K_u is converted to the hex string. The length of the converted string is 32 bytes. The hex string is then encoded according to the Base64 algorithm in order to obtain the value that is placed for the MD5 hash algorithm. Again, the IV is generated by calculating the MD5 hash.

Using the key K_u and IV which are obtained through the above procedure, the encrypted backup database file can be decrypted.

B. Analysis of database structure

As shown in Figure 1, KakaoTalk application supports a variety of message types such as text messages, emoticon messages, and files. Those contents can be stored as a database file through the backup feature. We carefully examined the structure of the database file used for the backup.

The database file is composed of six tables: `backupInfo`, `chatAttachmentFileInfo_v2`, `chatLogs`, `chatMembers`, `chatMetas`, and `chatRoomList`. Table I describes the purposes of those tables, respectively.

Basically, all activities in a chat room are stored into the `chatLogs` table. We can see that all the messages shown in Figure 1 were stored in the `chatLogs` database table (see Figure 4). In the `chatLogs` table, the type information for each message (`1`: text message, `18`: file attachment, `20`: emoticon) is also stored.

When a file was delivered in a chat room, the related records were created at the `chatLogs` and `chatAttachmentFileInfo_v2` tables at the same time. In the `chatLogs` table, the basic file information (e.g., URL, file size, and expired data) is recorded in the `attachment` field while some additional information (e.g., local file path), is stored in the `chatAttachmentFileInfo_v2` table (see Figure 5).

In the backup database file, time and date information is represented in Unix epoch time format.

VI. ATTACK SCENARIO

As you can see in Section V, the backup database file for chat messages is encrypted by using the user's password (PW'_u) and a unique number (N_u) that is assigned by the KakaoTalk server. If we are concerned about online guessing attack, current implementation might be acceptable with a policy of limiting the number of attempts to restore the backup database file with a typed user password. However, if we consider offline guessing attack [13] for the encrypted backup file on an external storage, a significant portion of user chosen passwords are not helpful to prevent offline guessing attacks. Using the password cracking tool called *Hashcat* (https://hashcat.net/hashcat/), a group of white-hat hackers successfully cracked about 90 percent of 16,000 passwords within a few hours [11].

If we assume that a target user chose a weak password, the last remaining barrier in generating the encryption key is the knowledge about N_u alone. However, N_u used for KakaoTalk is also highly guessable because N_u is not only a 9 digit number but also assigned in the registration order. According to our observations, the number of KakaoTalk users is currently less than 3 hundred million users that could be searched in a brute force manner.

To make matters worse, N_u can easily be obtained; if an attacker knows a victim's phone number or KakaoTalk ID, the attacker can add the victim as a friend without the victim's agreement [12] and then obtain the victim's N_u by searching it in the chat history database for the chat messages with all friends.

If we successfully guess PW'_u and N_u, we can generate the encryption key for the backup database file using the key generation procedure (see Figure 3). To show the feasibility of our attack scenario, we tried to decrypt a real backup database

Table: chatLogs

New Record | Delete Record

	logId	chatId	authorId	type	clientMsgId	sendAt	message	attachement	pc_logout	ev_msg_missir	ext_msg_miss
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
140	12760372213···	13623690417···		1	733953713	1473052690	This		0	1	0
141	12760372339···	13623690417···		1	733953714	1473052692	is		0	0	0
142	12760372402···	13623690417···		1	733953715	1473052692	the		0	0	0
143	12760372565···	13623690417···		1	733953716	1473052694	test		0	0	0
144	12760372857···	13623690417···		1	733953717	1473052698	message.		0	0	0
145	12760373563···	13623690417···		20	733953718	1473052706		{"type":"anim···	0	0	0
146	12761599116···	13623690417···		20	1472971663	1473067316		{"path":"4400···	0	0	0
147	12770946216···	13623690417···		18	736532157	1473178742	Test.zip	{"name":"Te···	0	0	0

Fig. 4. Example of `chatLogs` table.

Table: chatAttachmentFileInfo_v2

New Record | Delete Record

	expireDate	localFilePath	astModifiedDate	authorId	sendAt	fileName	fileSize	url	status	tempFilePath	token
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1474388342526	C:₩Users ₩Administra···	13117543592···	[redacted]	1473178742	Test.zip	8583508	https:// dn.talk.kakao···	3	NULL	1/ oWXbytdoA4···

Fig. 5. Example of `chatAttachmentFileInfo_v2` table.

file with our prototype implementation. SQLite Database browser (http://sqlitebrowser.org/) was used to view records in the database tables. Figure 4 shows the `chatLogs` table for the chat messages with friends. We can see that each message shown in Figure 1 was created as a record in the database table. Figure shows the `chatAttachmentFileInfo_v2` database table for the path and URL of the delivered file via KakaoTalk.

Finally, we evaluated the performance of offline password guessing attacks. For evaluation, we created the *encrypted* backup database files with the 100 passwords that were randomly chosen from the 0.5 million real passwords leaked from Yahoo. Because we can easily obtain N_u for a target victim with his or her phone number, we used a fixed constant for N_u. For offline password guessing attacks, we used publicly available password dictionaries consist of 32.6 million passwords. As a result, we successfully cracked 41 out of 100 passwords within a few hours (the mean cracking time was about 30.8 seconds with the standard deviation of about 50.2 seconds).

VII. Countermeasures

In this section, we discuss two possible defense mechanisms to mitigate offline password guessing attacks to decrypt the encrypted backup database file in IM applications.

A. Using a random secret number to generate the encryption key

In Section VI, we demonstrated that the encryption key for the backup database file in KakaoTalk could be generated by an attacker since a user u's password (PW'_u) and his or her user sequence number (N_u) (used for the encryption key) could be successfully guessed with a high chance.

Algorithm 1 Key and initial vector (IV) generation using a user u's random secret number R_u. $||$ represents the concatenation operator.

1: **function** KEY AND INITIAL VECTOR GENERATION(`string`)
2: **Input:** PW_u from user u
3: **Output:** K_u and IV
4: $PW_u \rightarrow$ server
5: **if** PW_u is correct **then**
6: $R_u \leftarrow$ server
7: $K_u = Hash(PW_u||R_u)$
8: $IV = Hash(K_u||PW_u||R_u)$
9: **return** K_u and IV
10: **end if**
11: **end function**

Without compromising usability of the key generation mechanism, our recommendation would be to use a random secret number with a high entropy instead of user sequence number assigned in the registration order. Algorithm 1 presents a possible implementation using a random secret number R_u for u to generate the encryption key and initial vector (IV). Here, R_u should be randomly chosen and sufficiently long. In practice, a secure hash function such as SHA-256 can be used for $Hash()$; the resulting 256-bit hash value is truncated to 128 bits for generating 128-bit key and IV without the step 8 in Algorithm 1.

B. Increasing the analysis complexity for identifying the key generation method

To generate the encryption key used for the backup database file, an attacker has to exactly identify the key generation procedure from machine codes in order to analyze how the encryption key is generated. Although this analysis needs to be performed only once, we can considerably increase the difficulty of the analysis by applying obfuscation techniques [7] in order to hide critical codes and control flows to be protected. Intuitively, this would make the IM application harder to analyze in practice.

However, software-based code obfuscation mechanisms have inherent limitations since program codes themselves can always be reviewed and then compromised by sophisticated attackers, before or during runtime. Therefore, we need to consider a secure environment for isolating the key generation procedure from other processes including a debugger. Hardware assisted solutions (e.g., TrustZone [5]) might be used to achieve this security goal.

VIII. Ethical and Legal Issues

The main motivation of our experiments is to analyze potential risks of the encrypted databases for chat messages in KakaoTalk and to suggest reasonable countermeasures to mitigate such risks. We did not collect any personal data without users' agreement.

We recently found that KakaoTalk already changed the storage location of the backup database file to their server. Therefore, we confirmed that the backup database file cannot be leaked anymore by the attack reported here.

IX. Conclusion

In this paper, we presented a forensic analysis of the backup feature in KakaoTalk, revealing the key generation procedure and the structure of the chat database used. Our findings on the key generation procedure demonstrated that the encrypted backup file could be leaked to unauthorized attackers.

Our study shows how difficult it is to securely protect the user's sensitive data files with only pure software technologies against reverse engineering analysis. Although we currently limited our security analysis to KakaoTalk messenger service alone, we believe this type of attack can also be applicable to other IM applications.

In future work, we plan to develop security mechanisms to protect encrypted databases against reverse engineering techniques.

X. Acknowledgements

This work was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-R0992-16-1006) supervised by the IITP (Institute for Information & communications Technology Promotion). This research was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (No. 2014R1A1A1003707). This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (No. R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

References

[1] M. S. Dahiya Aditya Mahajan and H. P. Sanghvi. Forensic Analysis of Instant Messenger Applications on Android Devices. *International Journal of Computer Applications*, 68(8):38–44, 2013.

[2] Nedaa B Al Barghuthi and Huwida Said. Social networks IM forensics: Encryption analysis. *Journal of Communications*, 8(11):708–715, 2013.

[3] Bertolt Meyer Alfred Kobsa, Sameer Patil. Privacy in instant messaging: an impression management model. *Behaviour & Information Technology*, 31:355–370, 2012.

[4] Cosimo Anglano. Forensic analysis of WhatsApp Messenger on Android smartphones. *Digital Investigation*, 11(3):201–213, 2014.

[5] Ahmed M. Azab, Peng Ning, Jitesh Shah, Quan Chen, Rohan Bhutkar, Guruprasad Ganesh, Jia Ma, and Wenbo Shen. Hypervision Across Worlds: Real-time Kernel Protection from the ARM TrustZone Secure World. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[6] Eunjeong Choi. KakaoTalk, a mobile social platform pioneer. *SERI Quarterly*, 6(1):63, 2013.

[7] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation-tools for software protection. *IEEE Transactions on software engineering*, 28(8):735–746, 2002.

[8] Kakao Corp. 2Q FY2016 Earnings Conference Call, 2016. http://www.kakaocorp.com/upload_resources/ir/event/ir_event_20160825012832.pdf (accessed on 2016-09-07).

[9] Alex Fitzpatrick. Here's How Police Get a Suspect's Facebook Information, 2016. http://mashable.com/2012/12/18/police-facebook/#ERE6rnihaPqw (accessed on 2016-09-14).

[10] Feng Gao and Ying Zhang. Analysis of WeChat on iPhone. In *2nd International Symposium on Computer, Communication, Control and Automation*, 2013.

[11] Dan Goodin. Anatomy of a hack: How crackers ransack passwords like "qeadzcwrsfxv1331". http://arstechnica.com/security/2013/05/how-crackers-make-minced-meat-out-of-your-passwords/ (accessed on 2016-11-02), May 2013.

[12] Eunhyun Kim, Kyungwon Park, Hyoungshick Kim, and Jaeseung Song. Design and analysis of enumeration attacks on finding friends with phone numbers: A case study with KakaoTalk. *Computers & Security*, 52:267–275, 2015.

[13] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications security*, 2005.

[14] Kyungwon Park and Hyoungshick Kim. Encryption Is Not Enough: Inferring user activities on KakaoTalk with traffic analysis. In *International Workshop on Information Security Applications*, 2015.

[15] Ronald Rivest. The MD5 message-digest algorithm. RFC 1321, MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[16] Neha S. Thakur. Forensic Analysis of WhatsApp on Android Smartphones. Master's thesis, University of New Orleans, 2013.