Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.30, NO.4, Aug. 2020 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.4.583

SNS 애플리케이션의 데이터 복호화 및 아티팩트 연구*

신 수 민,[1†] 강 수 진,[1] 김 기 윤,[1] 김 종 성[2‡]
[1,2]국민대학교(대학원생, 교수)

Study on SNS Application Data Decryption and Artifact*

Sumin Shin,[1†] Soojin Kang,[1] Giyoon Kim,[1] Jongsung Kim[2‡]
[1,2]Kookmin University(Graduate student, Professor)

요 약

스마트폰의 대중화로 현대인의 의사소통 수단으로 Social Networking Service(SNS)가 자리 잡았다. 의사소통 수단의 특성상 SNS는 다양한 보관 증거 및 생성 증거를 생성한다. 따라서 디지털 포렌식 수사관점에서 주요 분석대상이다. SNS를 제공하는 애플리케이션은 데이터를 중앙 서버에 저장하거나 사용자의 편의성을 위해 데이터베이스화하여 스마트폰 내부에 저장한다. 이때 일부 애플리케이션은 개인정보 보호를 위해 암호화 기능을 제공하며 이는 디지털 포렌식 수사관점에서 안티 포렌식으로 작용될 수 있다. 따라서 암호화 방안에 관한 연구는 지속적으로 선행돼야 한다. 본 논문에서는 SQLCipher 모듈을 통해 SQLite 기반의 데이터베이스 암호화 기능을 제공하는 두 종류의 애플리케이션을 분석했다. 각 데이터베이스를 복호화했으며 주요 데이터를 식별했다.

ABSTRACT

With the popularization of smartphones, Social Networking Service (SNS) has become the means of communication for modern people. Due to the nature of the means of communication, SNS generates a variety of archive and preservation evidence. Therefore, it is a major analysis target in terms of digital forensic investigation. An application that provides SNS stores data in a central server or database in a smartphone inside for user convenience. Some applications provide encryption for privacy, which can be anti-forensic in terms of digital forensic investigation. Therefore, the study of the encryption method should be continuously preceded. In this paper, we analyzed two applications that provide SQLite-based database encryption through SQLCipher module. Each database was decrypted and key data was identified.

Keywords: SNS, Database, Digital Forensic Investigation, SQLCipher Module

I. 서 론

한국미디어패널조사 결과에 따르면 2019년 기준 91.7%가 스마트폰을 보유하고 있으며, 2011년 이후로 꾸준히 상승하고 있다[1]. 이로 인해 현대인들의 연락수단이 전화, 문자에서 Social Networking Service를 제공하는 애플리케이션(이하 SNS 애플리케이션)으로 대체됐다. 이용자 간 교류 수단으로 자리 잡은 SNS 애플리케이션은 그 특성상 다양한 보관 증거 및 생성 증거가 생성되고 저장된다. 따라서 SNS 애플리케이션은 디지털 포렌식 수사관점에서 주요 분석대상이다.

일부 SNS 애플리케이션은 사용자의 개인정보 보호를 위해 각종 데이터를 암호화하여 저장하며, 이는 디지털 포렌식 수사관점에서 안티 포렌식으로 작용된

Received(03. 25. 2020), Modified(1st: 06. 29. 2020, 2nd: 07. 23. 2020), Accepted(07. 26. 2020)
* 본 연구는 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2017-0-00344, 최신 모바일 기기에 대한 암호해독 및 디지털 포렌식 분석)
† 주저자, tnals523@kookmin.ac.kr
‡ 교신저자, jskim@kookmin.ac.kr(Corresponding author)

다. 실제로 SNS 애플리케이션의 데이터가 암호화되어 수사가 어렵다는 점을 악용하여 마약이나 아기를 불법 거래하거나, 단체 채팅방을 통해 불법 촬영물을 공유하는 등의 사례가 존재한다[2, 3]. 따라서 안티 포렌식 행위에 대한 대응 방안은 지속적으로 연구돼야 한다.

본 논문에서는 데이터베이스를 암호화하여 사용자의 개인정보를 보호하는 애플리케이션 L사 애플리케이션과 B사 애플리케이션을 분석했다. 두 애플리케이이션은 각각 2018년, 2014년에 출시되었고 2020년 7월을 기준으로 Google Play Store에서 다운로드 수는 50만 이상이며 사용률과 영향력은 높지 않다. 하지만, 안티 포렌식 관점에서의 악용 가능성은 열려있다. 그러므로 디지털 포렌식 수사관점에서 선행연구는 필요하다.

2장에서는 관련 연구를 설명하며, 3장에서는 L사 애플리케이션을 분석하고 주요 아티팩트를 정리한다. 4장에서는 B사 애플리케이션을 분석하고 주요 아티팩트를 정리했으며. 마지막 5장에서는 결론으로 마무리한다. 다음은 본 논문의 연구 환경 및 연구 결과의 정리다[표 1, 2].

Table 1. Analysis Environment and Information of Tools

Devices and Software	Name and Version
PC	Windows 10
Smart Phone	Samsung Galaxy S5/Android 7.1.2 Samsung Galaxy A30/Android 9
Decompiler	JEB ver. 3.13.0.202002181912
Debugger	IDA Pro ver. 7.3
DB Browser	DB Browser for SQLite ver. 3.11.2
Hex Viewer	HxD ver. 2.3.0.0

II. 관련 연구

디지털 포렌식 관점에서 안티 포렌식으로 작용하는 데이터베이스 암호화의 대응 방안은 여러 방면에서 연구되고 있다. 세계적으로 많이 사용되는 인스턴트 메신저 WeChat은 SQLCipher를 사용하여 데이터베이스를 암호화하였으며 이에 대한 복호화 방안이 제시된 바 있다[4]. WeChat에서는 IMEI (International Mobile Equipment Identity)와 uin (unique identification number)을 사용하여 Passphrase를 생성하였으며, SQLCipher를 활용해 데이터베이스를 복호화했다. 개인정보 보호를 위해 강력한 암호화를 제공하는 ChatSecure의 데이터베이스를 복호화하고 주요 아티팩트를 분석한 연구도 존재한다[5]. ChatSecure의 데이터베이스 암/복호화 Passphrase는 사용자 입력 패스워드 기반으로 암호화되어 preference 파일에 저장되어있었다. 중국과 한국에서 많이 사용되는 메신저 카카오톡, 네이트온 그리고 QQ 대한 데이터베이스 복호화 방안도 연구된 바 있다[6]. 사용자 입력 패스워드를 기반으로 데이터를 암호화한 보안 메신저 SureSpot의 암호화된 데이터를 Android와 iOS 환경에서 복호화 방안 연구도 존재한다[7].

III. L사 애플리케이션

L사 애플리케이션은 관심사를 기반으로 소통하는

Table 2. Analysis Results of Study

Application	APK Version	Key Generation Algorithm	Key Generation Algorithm Parameter	Algorithm	Algorithm Parameter
L Company's Application	1.1.7	SHA-256	android_id, string1, string2, string3	SQLCipher (C, Passphrase, Pagesize, Iteration)	C : Database File Password : Output of Key Generation Algorithm Pagesize : 1,024 Iteration : 64,000
B Company's Application	2.1.28	nextInt()	-	SQLCipher (C, Passphrase, Pagesize, Iteration)	C : Database File Password : Output of Key Generation Algorithm Pagesize : 1,024 Iteration : 64,000

애플리케이션이다. 일대일 채팅, 그룹 채팅, 오픈 채팅과 일정을 정리할 수 있는 캘린더 기능을 제공한다. L사 애플리케이션의 데이터 구조는 다음과 같다(그림 1). databases 폴더 내에 존재하는 user.db는 사용자 관련 데이터를 저장하며, talk.db는 채팅 정보와 채팅방 정보를 저장한다. 두 파일은 모두 SQLCipher 모듈에 의해 암호화되어 있다. 본 장에서는 애플리케이션 분석을 통해 SQLCipher 모듈에 사용된 Passphrase 생성방안을 분석하고 실제 복호화한 뒤, 각각의 데이터베이스의 아티팩트를 정리한다.

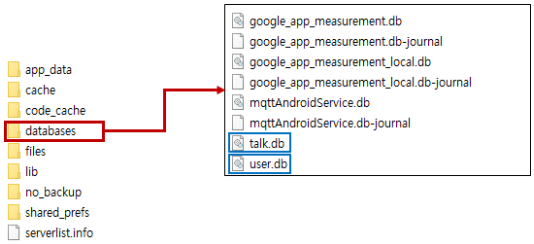

Fig. 1. Data Structure of L Company's Application

3.1 암호화 과정 분석

L사 애플리케이션은 데이터베이스 암호화를 위해 SQLCipher 모듈을 사용한다(그림 2).

SQLCipher는 SQLite 데이터베이스를 암호화하기 위한 확장 모듈이다. 사용자 입력 패스워드를 기반으로 암호화키를 생성하며, AES-256-CBC모드로 데이터를 암호화한다. 암호화키 생성 시 사용할 해시 알고리즘, 키 생성 함수의 반복횟수, 검증에 사용할 HMAC 알고리즘, 데이터베이스의 페이지 크기를 파라미터로 받아 조절할 수 있으며 잘못된 파라미터를 사용 시 옳은 패스워드를 입력해도 정상적인 복호화가 불가능하다[8].

각각의 데이터베이스별 Passphrase 생성과정은 다음과 같다.

```
public static void loadLibs(Context arg0, File arg1, LibraryLoader arg2) {
    Class v0 = SQLiteDatabase.class;
    synchronized(v0) {
        arg2.loadLibraries(new String[]{"sqlcipher"});
    }
}
```

Fig. 2. Load "SQLCipher" Library in L Company's Application

3.1.1 user.db 암호화 키 생성과정

user.db의 Passphrase 생성과정은 다음과 같다(그림 3).

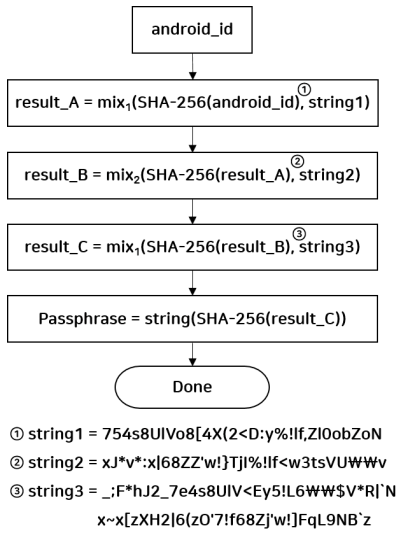

Fig. 3. Generating Passphrase of user.db

android_id는 디바이스를 구분하는 64-bit의 고유번호다. android_id는 안드로이드 버전에 따라 획득 방법이 상이하다. 안드로이드 7.0 이하에서는 ADB 명령어를 통해 획득할 수 있다(그림 4).

```
C:₩Windows₩system32>adb shell content query --uri content://settings/secure
--where "name=₩'android_id₩'"
Row: 0 _id=31, name=android_id, value=c5b9cf5153cfb8f2
```

Fig. 4. Getting android_id from ADB Command

또는, 핸드폰 내의 개발자 옵션에서 장치 호스트명 항목을 통해 획득할 수 있다(그림 5).

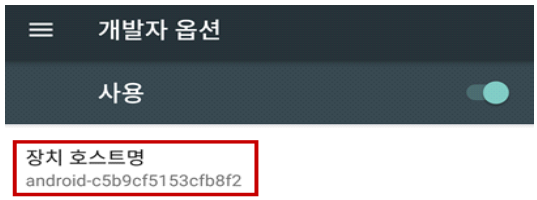

Fig. 5. Getting android_id from Developer Option

안드로이드 8.0 이상에서는 '/data/system/users/0' 경로에 저장된 settings_ssaid.xml 파일에서 애플

리케이션의 android_id를 획득할 수 있다(그림 6).

```
<setting id="116" name="10300" value="b65bd51c368e771d"
package="com.everysing.lysn" defaultValue="b65bd51c368e771d"
defaultSysSet="false" tag="null"/>
```

Fig. 6. Getting android_id from settings_ssaid.xml

획득한 android_id는 SHA-256 해시함수로 해싱한 뒤 고정된 문자열 string1, string2, string3과 mix 연산을 3회 반복 후 한 번 더 해싱하는 것으로 Passphrase가 된다.

본 논문에서 사용되는 mix_n 함수는 총 3가지로 다음과 같다.

$$A = Hash\ Value$$
$$B = Fixed\ String$$

$$\text{def}\ mix_1(A,B):$$
$$\quad \text{for}\ i\ in\ range\ (0, \max(len(A), len(B))):$$
$$\quad\quad result = result \| A[i*2 : i*2+2] \| B[i]$$
$$\quad result = result \| A[i:] \| B[i:]$$

$$\text{def}\ mix_2(A,B):$$
$$\quad \text{for}\ i\ in\ range\ (0, \max(len(A), len(B))):$$
$$\quad\quad result = result \| A[i*1 : i*1+1] \| B[i]$$
$$\quad result = result \| A[i:] \| B[i:]$$

$$\text{def}\ mix_3(A,B):$$
$$\quad result = B \| A$$

mix_1 함수는 해시값 2-byte당 고정된 문자열 1-byte를 조합하는 함수이다.

mix_2 함수는 해시값 1-byte당 고정된 문자열 1-byte를 조합하는 함수이다.

mix_3 함수는 고정된 문자열 뒤에 해시값을 연접하는 함수이다.

mix_1, mix_2 함수는 조합 후 해시값 또는 문자열이 남는 경우 남은 부분만큼을 조합된 문자열 뒤에 연접한다.

user.db 생성 시 사용되는 mix_n 함수는 순서대로 mix_1, mix_2, mix_1다. 다음은 user.db의 문자열 조합 과정을 시각화한 것이다(그림 7).

mix_n 함수를 모두 거친 결과물을 한 번 더 SHA-256 해시함수로 해싱하고 나온 데이터를 Hex-String으로 인코딩(소문자)하는 것으로 user.db의 Passphrase가 생성된다.

A = SHA-256(android_id) =	a_1	a_2	a_3	a_4	a_5	a_6	···	a_{62}	a_{63}	a_{64}
string1 =	7	5	4	s	8	U	···	Z	o	N
result_A = mix_1(A, string1) =	a_1	a_2	7	a_3	a_4	5	···	a_{62}	a_{63}	N
B = SHA-256(result_A) =	b_1	b_2	b_3	b_4	b_5	b_6	···	b_{62}	b_{63}	b_{64}
string2 =	x	J	*	v	*	:	···	U	₩	v
result_B = mix_2(B, string2) =	b_1	x	b_2	J	b_3	*	···	b_{62}	b_{63}	b_{64}
C = SHA-256(result_B) =	c_1	c_2	c_3	c_4	c_5	c_6	···	c_{62}	c_{63}	c_{64}
string3 =	_	;	F	*	h	J	···	B	`	z
result_C = mix_1(C, string3) =	c_1	c_2	_	c_3	c_4	;	···	B	`	z

Fig. 7. Mixing Rule for Generating user.db Passphrase

3.1.2 talk.db 암호화 키 생성과정

talk.db의 Passphrase 생성과정은 다음과 같다(그림 8).

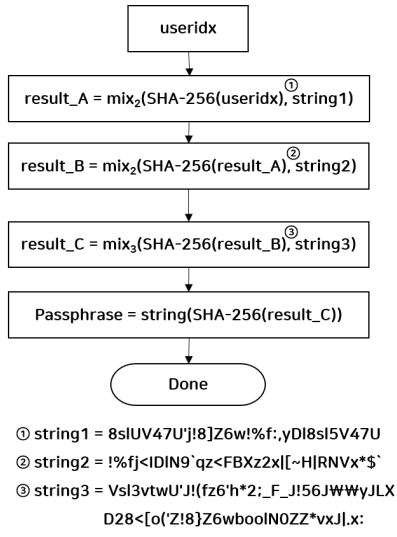

Fig. 8. Generating Passphrase of talk.db

useridx는 user.db의 users 테이블에 저장되어 있는 사용자 고유 index다. 따라서 user.db의 복호화가 선행돼야 한다.

talk.db의 Passphrase 생성과정은 user.db와 모두 동일하지만 고정된 문자열이 다르고 mix_n 함수의 사용 순서가 mix_2, mix_2, mix_3로 차이가 있다. 다음은 talk.db의 문자열 조합 과정을 시각화한 것이다(그림 9).

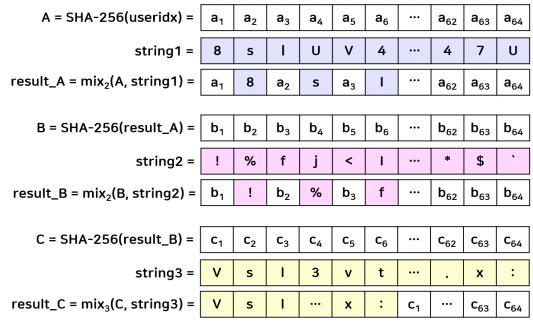

Fig. 9. Mixing Rule for Generating talk.db Passphrase

분석한 암호화 과정을 기반으로 재구성한 데이터베이스의 복호화 방안은 다음과 같다(그림 10).

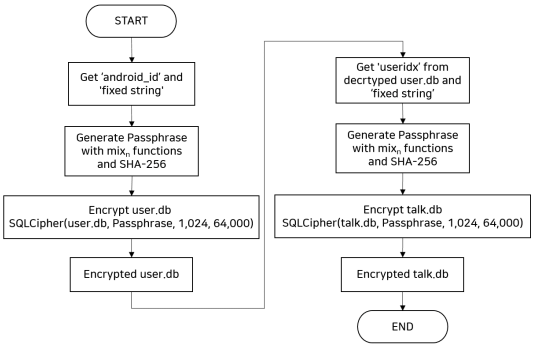

Fig. 10. Full Decryption Process for L Company's Application Database

3.2 데이터베이스 복호화 결과

L사 애플리케이션은 데이터베이스 암호화에 SQLCipher3의 default 파라미터를 사용했다. SQLCipher3의 default 파라미터는 다음과 같다.

Pagesize = 1,024
Key Derivation Function Iteration = 64,000
HMAC/KDF algorithm = SHA-1

실제 Passphrase 생성과정을 구현 후 생성된 Passphrase와 SQLCipher3 파라미터로 복호화 시도 결과 데이터베이스가 정상적으로 복호화됐다(그림 11).

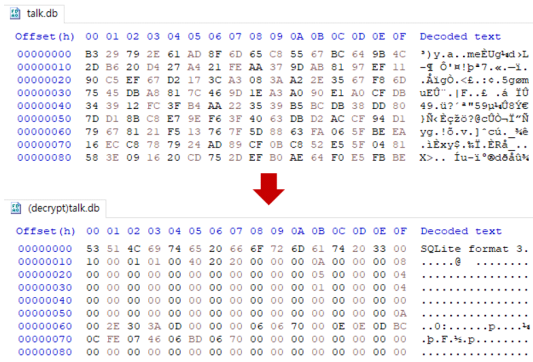

Fig. 11. L Company's Application Database Decryption Result

3.3 아티팩트 분석

user.db 내에 저장된 주요 아티팩트는 다음과 같다[표 3].

useridx 컬럼은 1자리 이상의 정수로 이루어진 데이터로 사용자의 고유 index 값이다. container 컬럼은 사용자의 유형을 의미하며, 로그인되어있는 사용자 계정은 1, 나머지는 0으로 표현된다. userId 컬럼은 사용자가 가입 시 입력한 아이디인 이메일 주소가 저장된다. profileMessage 컬럼은 사용자가 입력한 상태 메시지를 의미하고 phoneNo 컬럼은 사용자의 핸드폰 번호가 저장된다. updateTime 컬

Table 3. The Data of user.db

Table Name	Columns	Content	Remarks
users	useridx	user index	
	container	type of user	0 : user's friend 1 : user
	userId	user ID	email address
	username	user name	
	profile Message	profile message of user status	
	phoneNo	user phone number	
	update Time	user update time	Unix Time

럼은 Unix Time으로 저장된 사용자의 업데이트 시간이다.

talk.db 내에 저장된 주요 아티팩트는 다음과 같다[표 4].

Table 4. The Data of talk.db

Table Name	Columns	Content	Remarks
chats	roomidx	chatroom index	
	type	type of message	text, video, image, place, file, audio etc
	text	message content	
	sender	user index which sent message	
	time	message send time	
rooms	roomidx	chatroom index	
	name	user indexes in chat rooms	
	lastChat Text	message content which is sent lasty	
	lastChat Time	message time which is sent lastly	
	lastChat Type	message type which is sent lastly	text, video, image, place, file, audio etc
	lastChat Idx	message index which is sent lastly	
	unread Count	the number of unread message	
	mqttHost	mqtt host address	
	mqttPort	mqtt port number	ex. 1009
	openchat_ name	openchat room name	

chats 테이블은 메시지에 대한 정보를 저장한다. roomidx 컬럼은 메시지를 수/발신한 채팅방의 index 값을 의미한다. type 컬럼은 메시지의 유형을 나타내며, text, video, image, place, file, audio 등으로 표현된다. text 컬럼에는 수/발신한 메시지 내용이 저장된다. sender 컬럼은 메시지를 보낸 사용자의 index 값을 의미한다.

rooms 테이블에는 채팅방에 대한 정보를 저장한다. name 컬럼에는 채팅방에 포함된 모든 사용자의 index 값이 저장된다. unreadCount 컬럼은 읽지 않은 메시지의 개수를 의미한다. 오픈 채팅일 경우에는 openchat_name 컬럼에 오픈 채팅방의 이름이 저장된다.

IV. B사 애플리케이션

B사 애플리케이션은 블루투스를 이용해 소통할 수 있는 오프라인 메시지 애플리케이션이다. 블루투스를 통해 사용자와 일대일 채팅을 할 수 있다. Broadcast 기능을 통해 주변에 있는 모든 애플리케이션 사용자에게 메시지를 전송할 수도 있다. 데이터 구조는 다음과 같다(그림 12).

사용자의 정보 및 채팅 정보 등의 데이터는 databases 폴더 내의 암호화된 데이터베이스에 저장되어있다. 데이터베이스는 SQLCipher 모듈에 의해 암호화되어 있다. 본 장에서는 데이터 분석을 통해 Passphrase를 획득하고 데이터를 복호화한 뒤, 데이터베이스의 아티팩트를 정리한다.

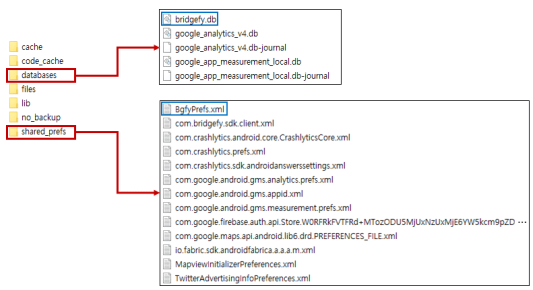

Fig. 12. Data Structure of B Company's Application

4.1 암호화 과정 분석

B사 애플리케이션은 데이터베이스 암호화에 사용한 SQLCipher의 Passphrase를 shared_prefs/

BgfyPrefs.xml 내에 존재하는 “database_password” 항목에 평문 형태로 저장한다(그림 13).

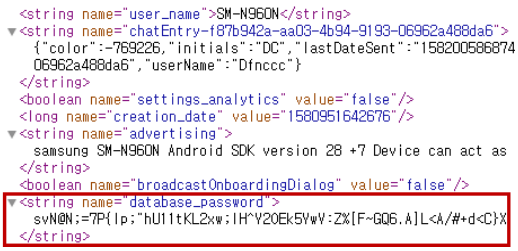

Fig. 13. “database_password” Stored in BgfyPrefs.xml

저장된 “database_password”의 생성과정은 다음과 같다(그림 14).

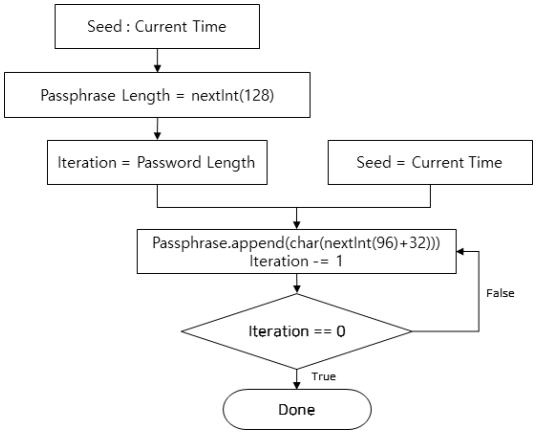

Fig. 14. Generating B Company’s Application Passphrase

nextInt 함수는 정수를 입력으로 받아 0부터 해당 정수 사이 임의의 정수를 반환하는 의사 난수 생성기다. 의사 난수 생성기의 Seed는 현재 시각이다. Passphrase의 길이는 nextInt(128) 함수를 통해 랜덤하게 정해진다. 이후 Passphrase의 길이만큼 반복하여 ASCII 코드 내에 표현 가능한 문자열을 뽑아 연접하여 Passphrase로 사용한다.

분석한 암호화 과정을 기반으로 재구성한 데이터베이스의 복호화 방안은 다음과 같다(그림 15).

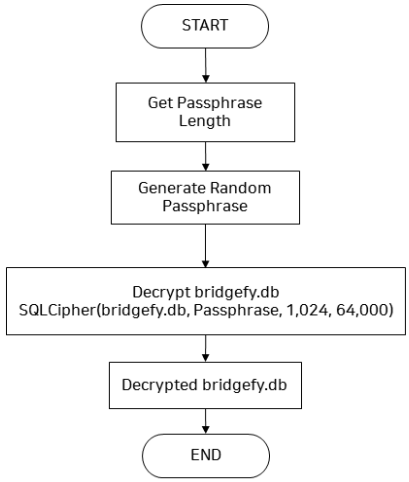

Fig. 15. Full Decryption Process for B Company’s Application Database

4.2 데이터베이스 복호화

B사 애플리케이션은 데이터베이스 암호화에 SQLCipher3의 default 파라미터를 사용한다. 실재 BgfyPrefs.xml에 저장된 database_password를 사용하여 데이터베이스를 복호화하여 정상적으로 복호화된 데이터베이스를 획득했다(그림 16).

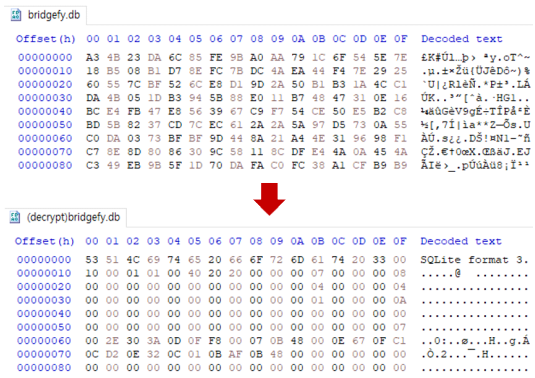

Fig. 16, B Company’s Application Database Decryption Result

4.3 아티팩트 분석

B사 애플리케이션의 데이터베이스 내에 저장된 주요 아티팩트는 다음과 같다〔표 5〕.

사용자와 사용자의 친구에 관한 정보는 bgf_friend 테이블에 저장되며 메시지에 관한 정보는 bgf_message

Table 5. The Data of B Company's Application Database

Table Name	Columns	Content	Remarks
bgf_ friend	id	user ID	
	user_ name	user name	
bgf_ message	dataSent	message send time	Unix Time
	image	image name	
	friend_id	friend ID	
	message_ type	message type	0 : text 1 : image (jpg, png, gif etc) 2 : location
	receiver	receiver ID	
	sender	sender ID	

테이블에 저장된다. id 컬럼은 사용자마다 부여되는 고유한 값이며, user_name 컬럼은 사용자가 설정한 이름이다. dataSent 컬럼은 메시지의 전송 시간을 의미하고, Unix Time으로 표현된다. 이미지를 전송했을 때 image 컬럼에 파일명이 저장된다. 텍스트, 사진과 사용자의 위치 좌표마다 메시지의 유형이 다르게 나타난다.

V. 결 론

SNS 애플리케이션에 저장된 데이터는 채팅 내용과 시간 정보 등 사용자의 행위를 파악하는 데 도움이 된다. 다양한 SNS 애플리케이션에서는 해당 정보를 중앙 서버에 보관하거나 암호화하여 디바이스 내부에 저장한다. 암호화 기술은 안티 포렌식으로 작용될 수 있다. 암호화된 데이터에 관한 복호화 연구는 선행되지 않는 한 디지털 포렌식 수사에 차질을 빚는다. 따라서 디지털 포렌식 수사관점에서 복호화 방안을 분석하고 가치 있는 아티팩트를 분류하는 등 선행 연구가 필요하다.

본 논문에서는 데이터베이스를 암호화하여 저장하는 두 종류의 SNS 애플리케이션을 분석했다. L사 애플리케이션은 android_id와 useridx를 기반으로 해시함수와 문자열 mix 함수를 사용하여 Passphrase를 생성했다. 생성된 Passphrase는 SQLCipher3와 함께 데이터베이스를 암/복호화했다. B사 애플리케이션은 현재 시각을 기반으로 랜덤 Passphrase를 생성하고 SQLCipher3와 함께 데이터베이스를 암/복호화했다. 랜덤하게 생성된 Passphrase는 shared_prefs 내에 존재하는 BgfyPrefs.xml 파일에 저장됐다. 애플리케이션 분석 후 Passphrase를 생성 혹은 획득했으며 실제 데이터베이스를 복호화했다.

두 애플리케이션은 데이터베이스의 암호화 키를 안전하지 못한 방법으로 생성하거나 안전하게 보관하지 않는다. 디지털 포렌식 수사관점에서는 증거 획득을 위한 부분이 되었지만, 개인정보 보호의 관점에서는 보완할 요소다. 보안 방법으로는 NIST의 권고사항인 패스워드를 저장한 파일을 암호화한다거나, 패스워드를 직접적으로 저장하는 것이 아닌 해시값만 저장, 그리고 고정된 암호키가 아닌 사용자의 입력값을 기반으로 암호키를 생성해야 하는 것 등이 존재한다[9].

마지막으로 본 논문에서는 복호화된 데이터베이스에서 주요 아티팩트를 선별 및 정리했다.

References

[1] Korea Information Society Development Institute, "KISDI STAT Report," http://www.kisdi.re.kr/kisdi/fp/kr/publication/selectResearch.do?cmd=fpSelectResearch&sMenuType=2&curPage=1&searchKey=TITLE&searchValue=&sSDate=&sEDate=&controlNo=14669&langdiv=1

[2] "Illegal trading of drugs and babies through SNS," KBS News, https://news.kbs.co.kr/news/view.do?ncd=4384756, Feb. 2020

[3] "Nth room: A digital prison of sexual slavery," Korea JoongAng Daily, https://koreajoongangdaily.joins.com/2020/03/29/features/DEBRIEFING-Nth-room-A-digital-prison-of-sexual-slavery/3075441.html, Mar. 2020

[4] Songyang Wu, Yong Zhang, Xupeng Wang, Xiong Xiong, and Lin Du, "Forensic analysis of WeChat on Android smartphones," *Digital Investigation*, vol.

25, pp. 5-23, Dec. 2018.

[5] Cosimo Anglano, Massimo Canonico, and Marco Guazzone, "Forensic analysis of the ChatSecure instant messaging application on android smartphones," *Digital Investigation*, vol. 19, pp. 44-59, Dec. 2016.

[6] Jusop Choi, Jaegwan Yu. Sangwon Hyun, and Hyoungshick Kim, "Digital forensic analysis of encrypted database files in instant messaging applications on Windows operating systems: Case study with KakaoTalk, NateOn and QQ messenger," *Digital Investigation*, vol. 28, pp. 550-559, Apr. 2019.

[7] Giyoon Kim, Uk Hur, Sehoon Lee, and Jongsung Kim, "Forensic Analysis of the Secure Instant Messenger Surespot," *Journal of Digital Forensics*, 13(3), pp. 175-188, Sep. 2019.

[8] Zetetic, "SQLCipher," https://www.zetetic.net/sqlcipher/

[9] NIST Computer Security Resource Center, "Draft NIST SP 800-118," https://csrc.nist.gov/csrc/media/publications/sp/800-118/archive/2009-04-21/documents/draft-sp800-118.pdf

〈저자소개〉

신 수 민 (Sumin Shin) 학생회원
2020년 2월: 국민대학교 정보보안암호수학과 졸업
2020년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 디지털 포렌식, 정보보호

강 수 진 (Soojin Kang) 학생회원
2018년 2월: 국민대학교 정보보안암호수학과 졸업
2020년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 디지털 포렌식, 정보보호

김 기 윤 (Giyoon Kim) 학생회원
2019년 2월: 국민대학교 정보보안암호수학과 졸업
2019년 3월~현재: 국민대학교 금융정보보안학과 석사과정
〈관심분야〉 디지털 포렌식, 암호학

김 종 성 (Jongsung Kim) 종신회원
2000년 8월/2002년 8월: 고려대학교 수학 학사/이학석사
2006년 11월: K.U.Leuven, ESAT/SCD-COSIC 정보보호 공학박사
2007년 2월: 고려대학교 정보보호대학원 공학박사
2007년 3월~2009년 8월: 고려대학교 정보보호기술연구센터 연구교수
2009년 9월~2013년 2월: 경남대학교 e-비즈니스학과 조교수
2013년 3월~2017년 2월: 국민대학교 수학과 부교수
2014년 3월~현재: 국민대학교 일반대학원 금융정보보안학과 부교수
2017년 3월~현재: 국민대학교 정보보안암호수학과 부교수
〈관심분야〉 정보보호, 암호 알고리즘, 디지털 포렌식