ISSN : 1976-5304
https://kdfs.jams.or.kr

보안 메신저 SureSpot 애플리케이션에 대한 포렌식 분석

김 기 윤, 허 욱, 이 세 훈, 김 종 성*
국민대학교 금융정보보안학과, 국민대학교 정보보안암호수학과*

Forensic Analysis of the Secure Instant Messenger SureSpot

Giyoon Kim, Uk Hur, Sehoon Lee, Jongsung Kim*
Dept. of Financial Information Security, Kookmin University
Dept. of Information Security, Cryptology and Mathematics, Kookmin University*

요 약

스마트폰 보급률이 늘어남에 따라 현대 디지털 포렌식 수사에서 모바일 포렌식의 중요성이 대두되고 있다. 특히 모바일 인스턴트 메신저의 데이터 분석은 주요 증거 획득에 용이하므로 필수적으로 이루어진다. 그러나 상당수의 메신저가 사용자의 개인정보보호를 목적으로 데이터를 암호화하여 저장하며, 그중 일부는 종단간 암호화(End-to-End Encryption)까지 적용하고 있어서 데이터 획득 후에도 메신저 대화 내용 식별 및 의미 파악이 어렵다. 따라서 디지털 포렌식 수사 관점에서 주요 정보를 암호화하는 메신저에 대한 복호화 방법 개발과 분석은 매우 중요하며 지속적인 연구가 필요하다. 본 논문에서는 주요 개인정보 및 대화 정보를 암호화하는 보안 메신저인 SureSpot의 암호화된 데이터를 Android와 iOS 환경에서 복호화 할 수 있는 방안을 제안하고 검증하였다. 또한, 서버 내에 저장된 데이터를 획득하기 위해 사용자의 로그인 패스워드를 복구하는 방법을 개발 하였으며, 안티포렌식에 대응하기 위하여 삭제된 메시지 중 일부 데이터 복원 방안을 제시한다.

주제어 : 모바일 포렌식, 인스턴트 메신저, 패스워드 복구, 데이터 복호화, 슈어스팟

ABSTRACT

Mobile forensic has become a necessity to the digital forensic investigators as the smartphone usage has increased signifi cantly. Since data generated through mobile instant messengers (IMs) can be core evidence, the analysis of mobile IMs is essential for the digital forensic investigation. However, it is challenging to obtain evidence from the mobile IMs, because they generally use encryption to their data and apply end-to-end encryption to protect user's private information. Therefore, decryption and analysis of the encrypted data is very important and must be performed before the digital forensic investigation process. In this paper, we have decrypted and analyzed various encrypted data of the Secure Instant Messenger, SureSpot. Based on this process, we have also developed a method to restore the login password, and a method to restore some of deleted messages.

Key Words : Mobile Forensics, Mobile Instant Messenger, SureSpot, Data Decryption, Password Recovery

Ⅰ. 서 론

디지털 기술이 발전하고 인터넷 보급률이 늘어남에 따라 2018년 기준으로 전 세계 인구의 39%(30억 명) 이상이 스마트폰을 사용하고 있다[1]. 사용률의 상당부분을 차지하는 것이 모바일 메신저(이하 메신저)이며, 20.1억 명의 사용자가 메신저를 이용한다[2]. 메신저는 편의성을 위하여 사용자의 다양한 개인정보를 수집하는데, 이때 수집되는 개인정보에 대한 침해사고를 예방하기 위하여 데이터를 암호화하여 저장한다. 또한, 메신저 통신 시 데이터 노출을 막기 위해 종단간 암호화를 적용한 보안 채팅을 제공하기도 한다. 종단간 암호화란 메시지를 전송하는 곳부터 받는 곳까지 모든 과정에서 메시지를 암호화된 상태로 전송하는 방식이다. 종단간 암호화를 적용하는 경우 메시지의 송/수신자만이 가지고 있는 키가 유출되지 않는 한, 메신저 서버에서조차 메시지의 원본데이터를 알 수 없다. 따라서 이를 악용하는 사례가 발생하기도 한다. 실제로 본 논문에서 분석한 SureSpot 메신저는 이슬람 극단주의 무장단체 IS와 접선하기 위하여 사용된 적이 있으며, 또 다른 종단

※ 이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 작성되었습니다.(No.2017-0-00344, 최신 모바일 기기에 대한 암호해독 및 포렌식 분석)
▪ Received 18 July 2019, Revised 20 September 2019, Accepted 23 September 2019
▪ 제1저자(First Author) : Giyoon Kim (Email : gi0412@kookmin.ac.kr)
▪ 교신저자(Corresponding Author) : Jongsung Kim (Email : jskim@kookmin.ac.kr)

간 암호화가 적용된 메신저는 마약 거래 등 불법적인 용도로 사용된 사례가 다수 존재한다[3]. 메신저 데이터는 디지털 포렌식 관점에서 매우 중요한 증거로 사용될 수 있으나, 복호화 및 분석 기술이 없다면 데이터를 수집하더라도 증거로써 사용이 불가능하다. 또한, 본 논문의 SureSpot과 같이 사용률이 저조하고 잘 알려지지는 않지만 높은 보안성을 갖는 메신저는 언제든 악용이 될 가능성이 있으므로 해당 메신저의 데이터 수집 및 분석, 복호화에 대한 선행연구가 필요하다.

본 논문은 2장에서 관련 연구와 본 논문의 연구 성과에 대해 다루며, 3장에서는 OS별 SureSpot 데이터 획득 방법을 다룬다. 4장에서는 SureSpot의 암호화 프로세스를 분석하고 복호화 방안을 제시하며, 5장에서는 복호화한 데이터를 토대로 주요 아티팩트를 분석한다. 6장에서는 로그인 패스워드와 일부 삭제된 데이터의 복구 방안을 제시하고 7장에서 결론으로 마무리한다.

Ⅱ. 관련 연구

모바일 메신저에 대한 디지털 포렌식 수사는 필수적으로 이루어지기 때문에 다양한 연구가 진행되고 있다. 일반적으로 모바일 메신저의 주요 데이터는 스마트폰의 루트 권한 없이는 획득이 어려운 영역에 존재한다. 또한, 루트 권한을 얻는 경우 스마트폰이 초기화되는 경우가 많아 문제가 된다. 그러므로 루트 권한 없이 데이터를 획득하기 위한 연구들이 수행되었다. N.Y.P. Lukito 등은 루트권한을 획득하지 않은 안드로이드 기기에서 데이터를 추출하는 가장 효율적인 방법은 데이터 백업 기능을 활용하는 것임을 보였다[4]. 스마트폰의 백업 기능은 Android의 Android Backup Manager[5]와 iOS의 iTunes(아이튠즈) 백업과 같이 OS에서 지원하는 백업과 스마트폰의 제조사에서 제공하는 백업 기능이 대표적이다. 하지만 상당수의 백업프로그램은 사용자 데이터를 보호하기 위해 백업데이터를 암호화하여 저장한다. 이러한 문제점을 극복하기 위해서 박명서 등은 사용자가 입력한 패스워드를 기반으로 백업데이터를 암호화하는 삼성, 화웨이 스마트폰의 백업데이터를 복호화하고, 고정된 키를 사용하는 LG 스마트폰의 백업데이터 복호화 방안을 제시한 바가 있다[6,7,8]. 본 논문의 SureSpot의 경우 백업 기능이 허용되어있기 때문에 암호화되어 있는 주요 데이터를 다양한 백업 프로토콜을 통해 획득할 수 있다.

암호화된 메신저 데이터에 대한 분석연구는 다음과 같이 소개된 바 있다. K. Rathi 등은 Telegram, Viber, WeChat, WhatsApp 4개의 모바일 메신저의 데이터를 추출하여 분석한 결과 장치 및 사용자의 정보를 암호화하여 저장하고 있음을 확인하였다[9]. C. Anglano 등은 강력한 암호화 기능을 제공하는 ChatSecure 모바일 메신저를 분석하여 사용자가 입력한 패스워드를 기반으로 데이터를 암호화함을 확인하였다[10]. 이 외에도 C. Anglano 등은 Telegram 메신저를 분석하여 Telegram의 데이터가 일반적인 데이터베이스(이하 DB)형태가 아닌 자체 형식임을 확인하였다[11]. 이러한 연구들은 본 논문에서 연구된 SureSpot과 부분적으로 매우 유사하다. SureSpot 애플리케이션은 256-bit의 강도를 갖는 강력한 암호 알고리즘을 사용하여 사용자의 주요 데이터를 암호화하는 것으로 알려져 있다[12]. 또한, 연구결과 일반적으로 사용하는 SQlite와 같은 DB를 사용하지 않고 JSON (JavaScript Object Notation)과 Plist (Property List)를 사용해 데이터를 관리하는 것을 확인했다.

본 논문에서는 디지털 포렌식 수사 관점에서 Android와 iOS에서 사용 가능한 SureSpot의 암호화된 데이터를 복호화하였으며 주요 아티팩트를 분석했다. 또한, 분석된 결과와 복호화 프로세스를 활용하여 사용자 로그인 패스워드를 복구하는 방법을 제시하고, 삭제된 메시지의 타입을 추측하고 SureSpot 데이터 추출 및 분석, 일부 삭제된 메시지를 복구하였다.

Ⅲ. 운영체제별 데이터 획득 방안

본 장에서는 운영체제별 SureSpot 애플리케이션의 데이터 획득 방법과 획득한 데이터에 관한 설명을 한다.

1. Android 기반 스마트폰의 SureSpot 애플리케이션 데이터 구조 및 획득

Android의 애플리케이션 데이터는 일반 사용자 권한에서 접근할 수 없다. 따라서 애플리케이션의 데이터를 획득하기 위해서는 백업 기능을 지원하거나 루트 권한이 요구된다. Android 애플리케이션의 백업은 OS에서 지원하는 Android Backup과 스마트폰 제조사에서 제공하는 백업 기능 등을 통해 가능하다. 이중 OS에서 지원하는 Android Backup 기능이 허용되어있는 경우 가장 손쉽게 데이터 획득이 가능하다. Android Backup은 애플리케이션이 백업 및 복원 서비스를 제공하도록 설정된 경우에만 사용할 수 있으며, 이는 애플리케이션의 AndroidManifest.xml 내 표기된 Android:allowBackup="true"를 통해 기능 제공 여부를 알 수 있다. 서비스가 제공될 시 일반 사용자의 권한으로도 Android Debug Bridge Backup (ADB Backup) 명령어를 통해 손쉽게 데이터 획득이 가능하다. ADB Backup은 Android Backup Manager를 간편하게 사용할 수 있게 해주는 도구로써 본 논문의 SureSpot 애플리케이션의 경우 Android Backup 기능이 허용돼있으므로 손쉽게 데이터 획득이 가능하다.

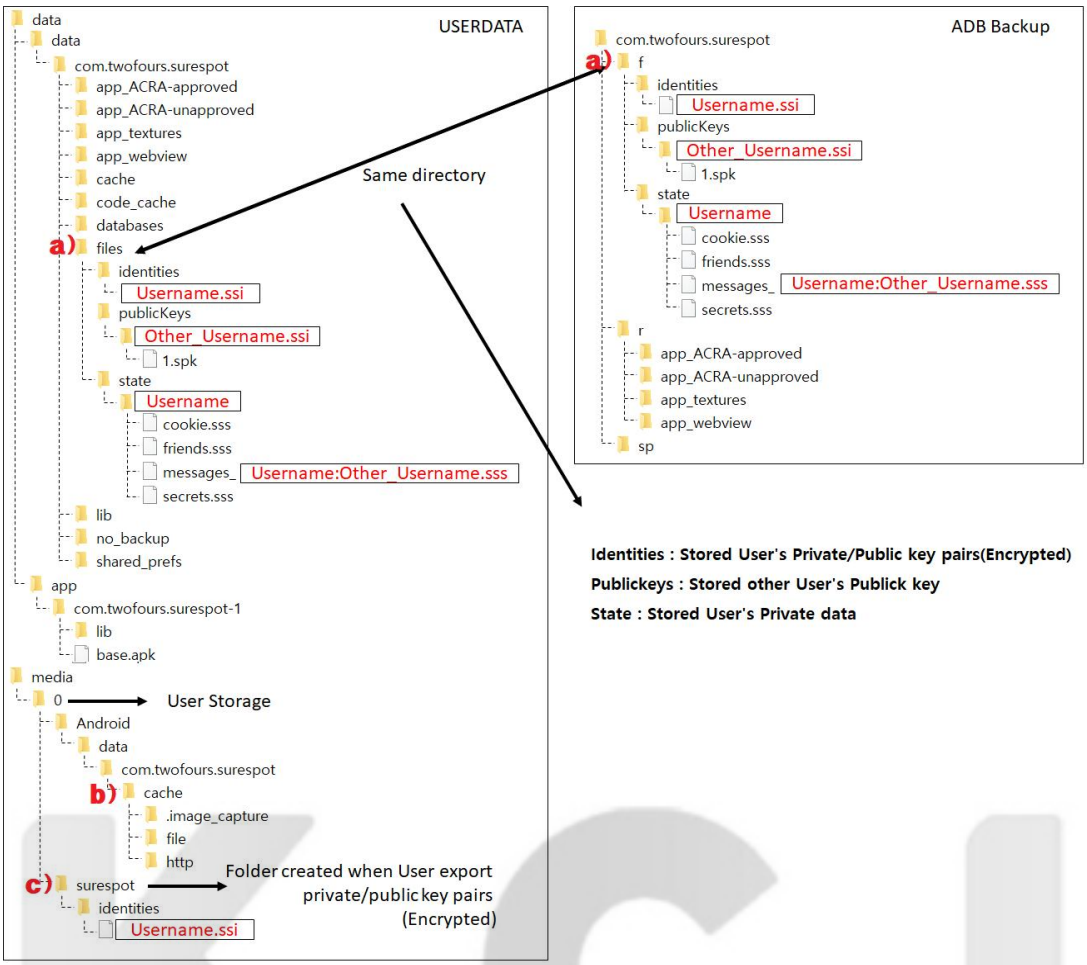

그림 1. 안드로이드 SureSpot 애플리케이션 데이터의 저장 경로
Fig. 1. Android SureSpot app data directory path

(그림 1)은 USERDATA영역에 존재하는 SureSpot 데이터와 ADB backup을 통해 획득한 데이터 구조를 보여주고 있으며, 주요 데이터에 대한 설명은 다음과 같다.

o File 폴더(a) : SureSpot 애플리케이션의 주요 정보가 모두 저장되어 있는 메인 저장공간으로 대화 내역, 친구목록 등 주요 정보들이 저장된다. ADB backup으로 데이터를 추출할 시 폴더의 이름이 'f'로 변경되며, 폴더 내부에는 속성에 따라 3종류의 데이터가 JSON 형식으로 저장된다. 종류별 특징은 다음과 같다.

- 'Identities' : 애플리케이션 로그인 권한이 존재하는 사용자의 정보가 저장되는 폴더이다.
  - Username.ssi: 사용자의 Private/Public key pairs가 암호화되어 저장되어 있다. 이에 대한 원본 획득방법은 다음 4장에서 다룬다.

- 'PublicKeys' : 대화를 한 번 이상 한 유저의 public key가 저장되어 있다. 'other username/1.spk'의 형태로 저장되며 GNU zip에 의해 압축되어 있다.

- 'State' : 로그인 권한이 존재하는 유저의 메인 아티팩트가 저장되어 있다.
  - Friends.sss : 대화 친구에 대한 데이터를 저장하고 있다.
  - Message 'username':'other username'.sss (이하 message.sss) : 대화 기록을 저장하고 있다.
  - Secrets.sss : 대화내역을 암/복호화하기 위한 비밀 키를 저장하고 있다(암호화되어있음).

o Cache 폴더(b) : SureSpot 애플리케이션은 상대방에게 송신 혹은 수신한 미디어 파일을 본 폴더에 암호화하여 저장하고 있다. 암호화된 파일 원본파일 획득 방법에 대해서는 다음 4장에서 다룬다.

o SureSpot 폴더(c) : 데이터 백업 기능을 사용 시 생성되는 폴더로 암호화된 사용자의 개인정보가 저장되어 있다.

2. iOS 기반 스마트폰의 SureSpot 애플리케이션 데이터 구조 및 획득

iOS는 아이튠즈의 백업기능을 통해 데이터 획득이 가능하다. 하지만 iOS의 백업기능을 이용하는 경우 Android와 달리 'username.ssi' 파일만 획득이 가능하다. 이는 아이튠즈의 백업기능을 사용하는 경우, 애플리케이션 패키지 내에 존재하는 Documents 폴더내의 파일만 백업이 되기 때문이다. 따라서, Android와 동일하게 전체 데이터를 획득하기 위해서는 탈옥이 선행되어야 한다. 탈옥이란 Apple에서 적용한 제한 사항을 제거하여 애플리케이션 스토어 이외의 제3사 소프트웨어를 설치할 수 있게 하는 방법이다. 탈옥이 된 iOS 기기에 제3사 소프트웨어 설치 시 최상위 권한 획득이 가능하다. 본 논문에서는 iPhone 6s iOS 11.4.1를 이용하였다. 탈옥을 통해 획득 가능한 데이터 및 경로 그리고 이에 관한 설명은 다음과 같다(그림 2).

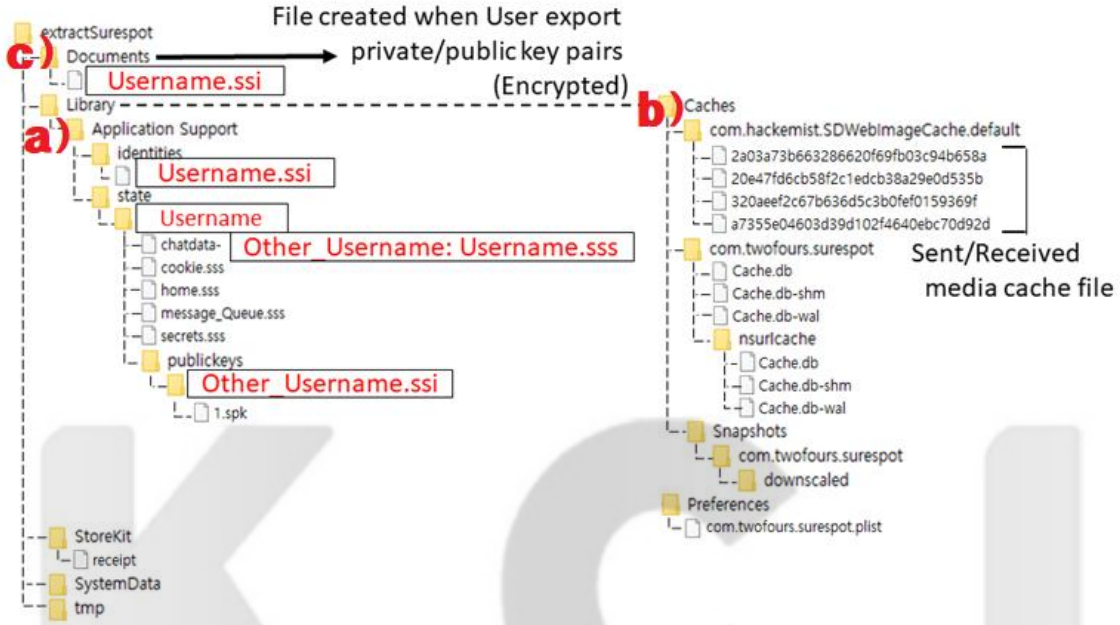

그림 2. iOS SureSpot 애플리케이션 데이터의 저장 경로
Fig. 2. iOS SureSpot app data directory path

iOS에 저장되는 주요 데이터 a), b), c)는 Android와 같다. 단, 데이터의 저장 형식이 Plist로 안드로이드 버전에서 저장되는 형식인 JSON과 다르며, 대화내역을 저장하고 있는 파일의 이름(chatdata-'username':'other name'.sss)과 일부 폴더의 경로가 다르다. 단, 탈옥이 불가능한 버전의 iOS의 경우 데이터 추출에 제약이 걸리나 백업을 통해 추출되는 'username.ssi'를 통해 패스워드를 복구하여 로그인하는 방식으로 데이터 획득이 가능하다.

Ⅳ. SureSpot 암호화 프로세스 분석

암호화 프로세스 분석은 메신저 내에서 사용되는 인자들을 리버스 엔지니어링을 통해 획득 후 재구성함으로 분석되었으며 오픈소스와 비교하여 검증하였다. 암호화 과정은 크게 키 교환 프로토콜을 통한 비밀 값 생성, 공유된 비밀 값을 토대로 암/복호화 키 생성, 메시지 암/복호화의 3단계로 나뉜다. 키 교환 프로토콜에는 NIST 표준 타원곡선중 하나인 Secp521r1을 사용하고 있으며 키 생성에는 SHA256 해쉬함수를 사용한다. (그림 3)은 SureSpot 애플리케이션의 전반적인 메시지 암호화 과정이다.

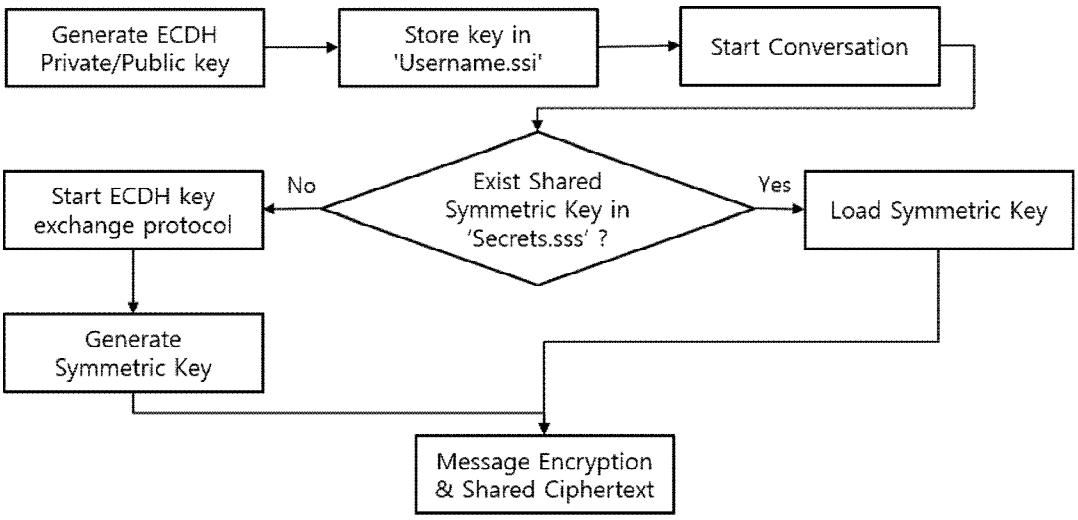

그림 3. SureSpot 메시지 암호화 과정
Fig. 3. Encryption process of SureSpot message

SureSpot은 메시지 암호화 과정에서 세 가지의 데이터를 암호화하여 저장한다. 첫째는 종단간 암호화를 위한 대칭키를 생성하는데 사용되는 개인키/공개키 쌍을 저장하고 있는 'Username.ssi' 파일이고, 둘째는 종단간 암호화에 사용된 대칭키를 저장하고 있는 'Secrets.sss' 파일이며 마지막은 주고받은 메시지이다. 각 파일 및 데이터를 암/복호화하기 위해 사용하는 알고리즘은 AES-256-GCM으로 동일하나 알고리즘에 사용되는 키의 생성 과정에서는 차이점이 존재한다. GCM은 Galois/Counter Mode의 약자로 기밀성과 무결성을 동시에 제공하는 암/복호화 모드이다. 본 장에서는 암호화된 주요 데이터들을 암호화하는 과정을 제시하며 이를 토대로 복호화 방법을 제시한다. 복호화된 데이터들에 대한 상세 내용은 다음 5장에서 다룬다.

1. Username.ssi

'Username.ssi'는 타원곡선을 이용한 Diffie-Hellman 키 교환 방식과 디지털 서명 알고리즘을 사용하기 위한 private/public key pairs를 저장하고 있다. 'Username.ssi' 파일은 암호화 과정은 동일하지만 서로 다른 두 개의 비밀 키(Key A, Key B)에 의해 서로 다른 두 가지 데이터(Data A, Data B)로 존재한다. 각 데이터를 수식으로 나타내면 다음과 같다.

$$Data\,A = Encrypt_{keyA}('Username.ssi') \quad (1)$$
$$Data\,B = Encrypt_{keyB}('Username.ssi') \quad (2)$$

다음은 Android와 iOS에서 두 데이터(Data A, Data B)를 획득하는 방법이다.

o Android
- Data A : 스마트폰 USERDATA영역에서 획득 혹은 ADB Backup을 통해 획득 가능
- Data B : 애플리케이션 내부 백업 기능을 이용하여 구글 드라이브 혹은 로컬 Storage에 저장하여 획득 가능

o iOS
- Data A : 탈옥을 통해 관리자 권한을 획득하여 데이터 추출 가능
- Data B : 애플리케이션 내부 백업 및 아이튠즈 백업을 통해 획득 가능

수식 (1)과 (2)에 사용된 암호화의 상세 과정은 다음과 같다.

$$Pwd = (Password \| Identity) \quad (3)$$
$$Key = PBKDF\,2(Pwd, Salt, iter = 1000, HMAC-SHA256) \quad (4)$$
$$CipherText = AES-256-GCM(PlainText, IV, KEY) \quad (5)$$

수식 (3)의 패스워드(Password)는 SureSpot 애플리케이션에 로그인 시 사용자가 입력하는 패스워드이며 identity는 대상이 Data A인지 Data B인지에 따라 달라진다. Data A의 경우 "_cache_identity" 문자열을 identity로 사용하며, Data B의 경우"_export_identity" 문자열을 identity로 사용한다. Salt와 IV (Initial Vector)는 랜덤하게 생성된 16

bytes의 난수이며, 평문(Plaintext)은 'Username.ssi'의 데이터를 의미한다. 이외의 암호화 과정은 동일하며 암호화가 종료되면 IV, salt, 암호문(Ciphertext)의 순서대로 연접하여 데이터를 저장한다[표 1].

표 1. 암호화된 파일 구조
Table 1. Structure of encrypted files

Offset(Bytes)	Content
0~15	IV
16~31	Salt
32~	CipherText

따라서 암호화된 파일 데이터 획득 후 수식 데이터에 맞게 identity를 설정해준 후, (3)과 (4)의 과정을 통해 키를 생성해 AES-256-GCM로 복호화하면 사용자의 개인키/공개키 쌍이 저장되어 있는 JSON 파일을 획득할 수 있다. 해당 JSON파일에는 X.509 인증서 기반의 DH 개인키/공개키와 DSA 개인키/공개키를 저장하고 있다(그림 4).

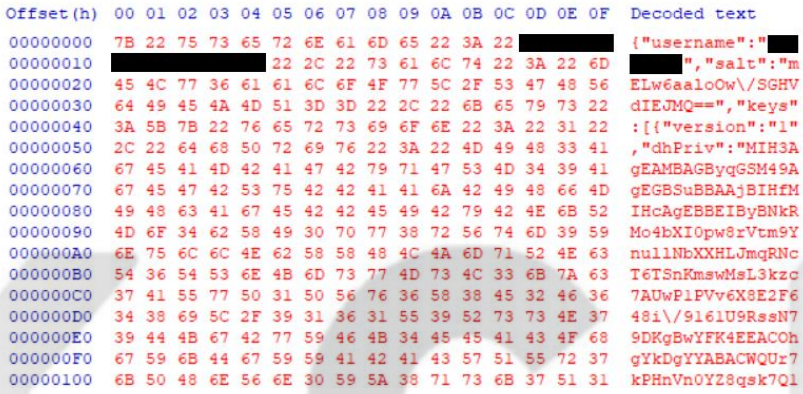

그림 4. Username.ssi 복호화 결과
Fig. 4. Decrypted result of Username.ssi

'Username.ssi'에 저장되어 있는 사용자의 개인키와 대화 상대방에게 받은 공개키는 메시지를 암/복호화하기 위한 대칭키(비밀키)를 공유하기 위해 사용된다. 공유된 비밀키의 상세한 생성과정은 다음과 같다.

$$SharedSecret = (MyPrivKey * Other'sPublicKey) \bmod p \quad (6)$$

$$MessageEncryptKey = SHA256(SharedSecret) \quad (7)$$

이때, 공유된 비밀키는 개인키/공개키 쌍을 새로 생성하기 전까지 반영구적으로 사용한다. Android의 경우 SureSpot 사용 여부에 상관없이 공유된 비밀키를 'Secrets.sss'에 저장하지만, iOS의 경우 SureSpot을 사용하고 있는 동안에만 'Secrets.sss'에 저장하며, 애플리케이션 종료 시 삭제한다.

2. Secrets.sss

'Secrets.sss'는 메시지 암/복호화에 사용되는 비밀키들을 저장하고 있다. 대화를 진행할 때, 상대방과 공유한 비밀키가 저장되어 있는 경우 'Secrets.sss'로부터 키를 추출하여 메시지를 암/복호화한다. 상대방과 공유한 비밀키가 없는 경우 'Username.ssi'에 저장되어 있는 개인키와 상대방으로부터 받는 공개키를 사용하여 비밀키를 생성하여 파일에 업데이트 한다. 'Secrets.sss'의 암호화 과정은 'Username.ssi'와 암호화 과정과 동일하나 암호화에 사용되는 키 생성 시 identity의 연접이 없고, PBKDF2의 반복횟수가 다르다는 차이점이 있다. 자세한 과정은 다음과 같다.

$$Key = PBKDF2(Password, Salt, iter = N, SHA256 - HMAC) \quad (8)$$

$$CipherText = AES-256-GCM(PlainText, IV, Key) \quad (9)$$

이때, 수식 (8)의 반복횟수 N은 Android와 iOS에서 차이가 존재하며, 복호화된 데이터의 형식에도 차이가 있다.

o Android

Android의 경우, (8)의 N이 5000회이며, 복호화 시 JAVA의 HashMap Object를 획득할 수 있다. HashMap Object의 key는 '사용자의 아이디:사용자의 버전:친구의 아이디:친구의 버전:해쉬 관련 플래그 값' 이며 이에 대응하는 value의 정보는 32 bytes의 공유된 비밀 값이다. 따라서 (그림 5)의 friendname과의 대화 내역을 복호화하기 위해서는 value의 시작을 알리는 '0x00000020' 이후의 32 bytes를 획득하여 비밀키로 사용하면 된다.

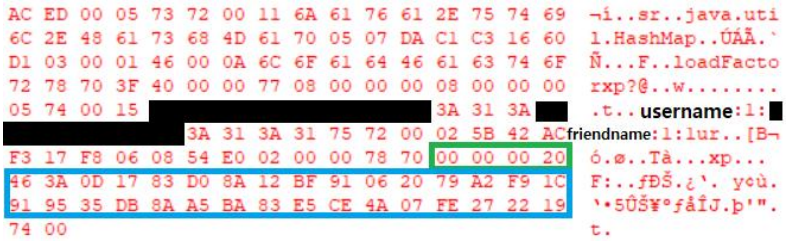

그림 5. 안드로이드 'Secrets.sss'복호화 결과
Fig. 5. Decrypted Android 'Secrets.sss'

o iOS

iOS의 경우 N이 20000회 이며, 복호화 시 plist형태로 저장된 Android와 동일한 데이터를 획득할 수 있다(그림 6).

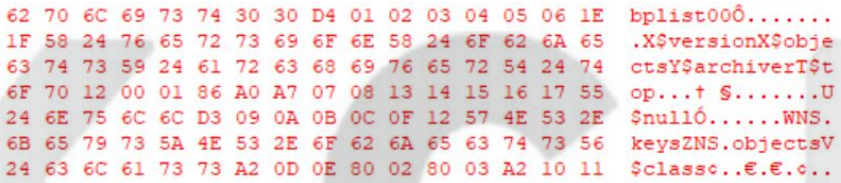

그림 6. iOS 'Secrets.sss'복호화 결과
Fig. 6. Decrypted iOS 'Secrets.sss'

단, iOS의 경우 안드로이드 '0x00000020'과 달리 키의 구분자가 0x4F1020이며, 매번 동일한 순서로 저장되어있는 Android와 다르게 SureSpot 사용 시 대화한 상대방 순서대로 키가 저장되어있으므로 유의해서 키를 추출해야 한다. 즉, 'Secrets.sss' 획득이 가능하고 복호화가 가능하다면, 사용자 간의 비밀키와 메시지의 원본 데이터 획득이 수월해진다. 만약, 'Secrets.sss'의 획득이 어려운 경우, 수식(6),(7)을 통해 복호화 키를 생성해야 한다.

3. Message(text/media data)

사용자와 상대방간의 대화내용은 message 'username' : 'other username'.sss 혹은 chatdata-'username' : 'other username'.sss (이하 'Message.sss')에 암호화 및 Base64 인코딩되어 저장되어있다(그림 7).

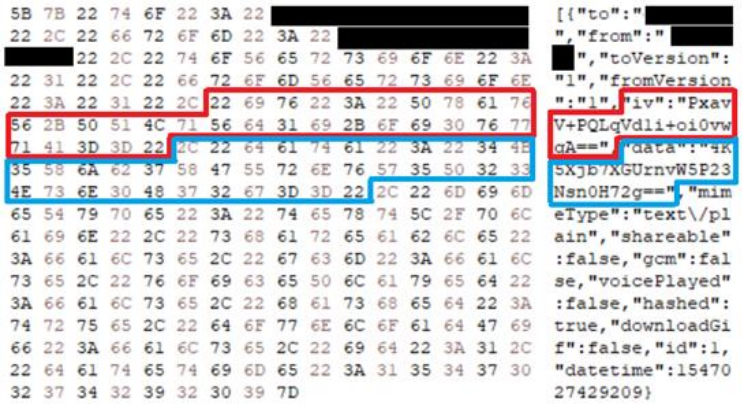

그림 7. Message.sss에 저장되어있는 데이터
Fig. 7. Data of Message.sss

메시지 암호화에 사용된 비밀키는 'Secrets.sss'에 저장된 값이며 암호화 알고리즘은 AES-256-GCM이다. IV의 경우 'Message.sss'에 Base64 인코딩되어 같이 저장되어 있다. SureSpot에서 수/발신이 가능한 데이터('Message.sss'에 저장 되는 데이터)의 종류는 텍스트 메시지, 사진, 음성 그리고 GIF 이미지로 네 가지다. 종류별 암호화된 데이터의 복호화 방법은 동일하나 데이터 획득방법은 다르다. 다음은 종류별 데이터의 저장되어 있는 방식과 원본데이터의 획득방법이다.

- o text message : 암호화하여 저장
- o picture/voice record : 데이터를 암호화하여 서버에 업로드 후, 다운로드 URL을 평문으로 저장
- o GIF image : 원본데이터가 저장되어 있는 링크를 암호화하여 저장

Text message와 GIF 이미지는 Base64로 인코딩된 IV와 Data를 디코딩 후 AES-256-GCM로 복호화 시 원본데이터 획득이 가능하다. 단, GIF 이미지의 경우 복호화된 결과가 링크 정보이므로 링크를 통해 데이터를 내려받는 추가작업이 필요하다. Picture/voice record의 경우 URL로부터 데이터를 내려받은 후 AES-256-GCM 복호화를 통해 원본 파일을 획득할 수 있다. 다음은 실제로 복호화 도구를 제작하여 'Message.sss'를 복호화한 결과이다(그림 8).

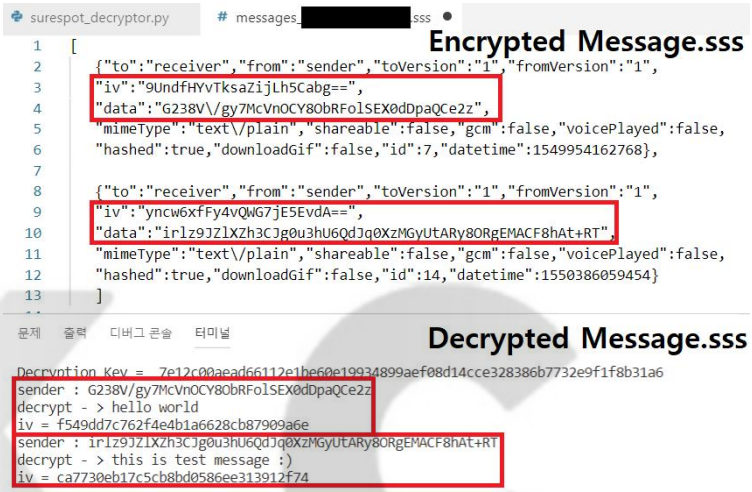

그림 8. Message.sss의 암호화된 데이터 복호화 결과
Fig. 8. Result of decrypted message in Message.sss

Ⅴ. SureSpot 복호화 데이터 분석

본 장에서는 SureSpot 애플리케이션 주요 속성 값에 대한 분석결과를 설명한다. 'Username.ssi'의 경우 각종 개인키와 공개키의 집합체이고, 'Secrets.sss'은 앞서 복호화 과정에서 서술한 대로 비밀키를 저장하는 용도로만 사용하기 때문에 본 장에서는 다루지 않는다. 따라서 본 장에서는 디지털 포렌식 수사에 활용하기 위해 주요 정보가 저장된 'Friends.sss'와 'Message.sss'의 주요 키 값과 그 속성 값의 분석결과를 설명한다.

1. Friends.sss

'Friends.sss'파일은 GNU Zip에 의해 압축되어있는 파일(혹은 plist로 저장된 파일)로 대화 상대방에 관련된 데이터가 저장되어있다(그림 9).

그림 9. Friends.sss 내의 데이터
Fig. 9. Friends.sss's Data

해당 데이터들은 일반적으로 공란으로 되어있으며(좌측) 설정에 따라 데이터가 변경된다(우측). 'Friends.sss' 파일 내의 주요 Key-Value pairs와 그에 대한 설명은 〔표 2〕와 같다.

표 2. Friends.sss파일 내의 주요 키 값과 속성 값
Table 2. Key-Value Pairs of Friends.sss

Key	Value	Note
User ControlID	Increase the vlaue to 1 each time you allow ·block · delete friend, assign nicknames, and remove alias	
name	Friend's ID	
flag	1 : Deleted friend 2 : Invited friend 4 : New friend 8 : A friend who exchanges messages 16 : Message Activity 32 : Inviter	2 : Before the other party has yet to respond 32 : Before user respond
last Received Message ControlID	Number of messages deleted	
last Available MessageID	Number of messages you have delivered to the other user	
last ViewedMeesageID	Number of messages checked	Never reduced
imageUrl	Address of the image you have set up for the profile	Encrypted
imageIV	IV used to encrypt profile set images	
aliasData	Base64 encoded encrypted alias	
aliasIV	Base64 encoded IV used to decrypt aliasdata	
aliasHashed	Hashing of aliasdata	True/False
muted	Alarm	True/False

2. Message.sss

'Message.sss' 파일은 GNU Zip에 의해 압축되어 있는 파일(혹은 plist로 저장된 파일)로 대화 상대방에 관련된 데이터가 저장되어 있다(그림 8). 'Message.sss'내의 주요 Key-Value pairs와 그에 대한 설명은 〔표 3〕과 같다.

표 3. Message.sss파일 내의 주요 키 값과 속성 값
Table 3. Key-Value Pairs of Message.sss

Key	Value	Note
to	message receiver account	
from	message sender account	
toVersion	receiver version	
fromVersion	sender version	
iv	IV used to encrypt Data	
data	TextMessage or URL	
mimeType	Message type	- text/plain - audio/mp4 - image - gif/https
shareable	Available to saved Image	True/False
vocieplayed	Check Voice-Playing	True/False
downloadGIF	Check GIF-Image Viewing	True/False
id	Message id	Unique number
datetime	Message sent/received time	
datasize	Data length	Not exist, when mimeType is text/plain

Ⅵ. SureSpot 로그인 패스워드 및 삭제된 메시지 복구 방안

용의자들은 자신의 범죄에 대한 증거를 삭제하려 하므로 디지털 포렌식 수사 관점에서 삭제된 데이터를 복구하는 방안 연구는 매우 중요하다. 본 장에서는 SureSpot의 고정정보를 이용하여 사용자의 로그인 패스워드와 삭제된 메시지 일부에 대한 복구 방안을 제시한다.

1. 로그인 패스워드 복구

앞선 암호화 프로세스에 의하면 'Username.ssi'와 'Secrets.sss' 파일은 사용자의 로그인 패스워드에 기반을 두어 암호화된다. 즉, 패스워드가 동일하다면 암호화에 사용되는 키 역시 동일하다. 또한, 두 파일은 특정 데이터 포맷을 가지고 있으므로 고정된 값이 존재한다. 본 논문에서는 해당 고정 값을 이용하여 로그인 패스워드를 추정하고 복구한다. 다음 〔표 4〕는 두 파일의 평문으로부터 획득한 고정 값이다.

표 4. Username.ssi와 Secrets.sss의 고정 값
Table 4. Fixed Data of 'Username.ssi' and Secrets.sss'

File Name	Fixed Data(Offset 0 ~)	Decoded Text
Username.ssi	7B 22 75 73 65 72 6E 61 6D 65 22 3A 22	{"username":"
Secrets.sss(Android)	AC ED 00 05 73 72 00 11 6A 61 76 61 2E 75 74 69	¬í■|sr■java.uti
Secrets.sss(iOS)	62 70 6C 69 73 74 30 30 D4 01 02 03 04 05	bplist00Ô┌┘|

각 파일의 고정 값은 Offset 0부터 시작하므로 패스워드 추측 후 생성된 키로 복호화를 진행하고, 복호화된 첫 16 bytes가 〔표 4〕와 일치한다면 해당 패스워드가 유효함을 알 수 있다. 'Username.ssi' 파일을 암호화하기 위해 생성되는 키는 키 생성과정에서 PBKDF2 반복횟수가 'Secrets.sss'보다 더 적으므로 패스워드 추측은 'Username.ssi' 파일을 이용하는 것이 효율적이다. 단, 'Username.ssi'의 고정데이터는 16 bytes가 아니므로 낮은 확률로 잘못된 패스워드를 유효하다고 추측할 수 있다. 따라서, 추측된 키로 'Secrets.sss'를 복호화하여 패스워드를 검증해보는 것이 필요하다. 다음은 실제 'Username.ssi' 파일을 대상으로 자주 사용되는 ASCII 코드 72자 (A-Z,a-z,0-9 외 특수문자 일부)를 선정하여 패스워드 전수조사 수행 시간을 측정한 결과이다〔표 5〕.

표 5. ASCII 코드 72자 기반 패스워드 복구 시간 측정
Table 5. Password Recovery Times Based on 72 Characteristic of ASCII Code

Test Spec[1]			
CPU	Intel cpu i7-8700K @3.70GHz		
Thread Num	Single		
Language	C / 64bit Release mode (Optimiziation option : O2)		
File Name	Password Length	Time	Note
Username.ssi	2	2.52 seconds	
	3	3 minutes 1 seconds	
	4	3 hours 37 minutes	
	5	10 days	
	6	2 years	Estimation

*Password length is unlimited

2. 로그인을 통한 메시지 복구

SureSpot 애플리케이션은 메시지를 1,000개까지 저장하며 그 이후부터는 가장 오래된 메시지를 삭제한다〔10〕. 이때 1,000개의 메시지는 모두 암호화되어 서버에 저장되며 로그인 시 서버로부터 암호화된 데이터를 내려받는다. 따라서 로그인을 위한 identity가 존재한다면 다른 기기에서 로그인하여 서버에 저장된 대화 내용을 내려받을 수 있다. 이때 identity는 애플리케이션에서 백업 기능을 통해 백업한 데이터(3장의 Data B)다. Data B를 획득한 후 앞선 로그인 패스워드 복구 알고리즘을 통해 패스워드를 복구해 로그인을 하면 된다. 만일 Data A만 획득이 가능한 하다면 패스워드 복구 후, 수식 (3)의 identity를 "_export_identity"로 적용하여 암호화한 후 Data B를 생성하고 이를 통해 로그인하면 된다. 단, 사용자가 의도적으로 삭제한 데이터는 서버에서도 삭제되므로 로그인을 통한 데이터 복구는 애플리케이션을 삭제하거나 의도적으로 데이터 영역을 제거한 경우에만 유효한 데이터 복구 방안이다.

3. 삭제된 메시지 복구

메시지가 삭제되는 경우 'Message.sss' 파일 내에서 해당 속성 값이 완전히 삭제되고 파일 크기에 변형이 일어나기에 일반적으로 복구는 불가능하다. 하지만, 메시지의 id에는 변화가 일어나지 않기 때문에 이를 이용해 메시지가 삭제됐음을 추측할 수 있다. 이는 고의로 메시지를 삭제한 경우에도 동일하다. 예를 들어 'Message.sss' 파일내의 id가 7, 8, 9인 메

1) GPU(Nvidia RTX 2080 Ti 10대) 사용 시 초당 약 1900만 회의 조사가 가능하며 8자리 패스워드 조사에 약 1년 3개월이 걸린다[13].

시지가 있을 때 9번 메시지를 삭제하고 새로운 메시지를 작성하면 기록되어 있는 id는 7, 8, 10이다. 따라서 불규칙한 메시지 id는 해당 데이터를 은닉 또는 증거인멸의 시도로 볼 수 있다. 삭제된 메시지의 복구는 어려우나 삭제된 메시지가 텍스트 메시지가 아니고, 캐시 파일이 존재하는 경우에는 복구가 가능하다[표 6]. 또한, 이를 이용하여 삭제된 메시지의 타입을 유추할 수 있다. SureSpot은 앞 장에서 설명하였듯 텍스트 메시지를 제외한 모든 메시지를 암호화된 데이터 혹은 원본 데이터(GIF)를 서버에 업로드하며, 메신저 수/발신 시 해당 링크를 주고받는다. 이때 관련된 캐시 정보가 디바이스에 자동으로 저장되며 의도적으로 삭제하지 않는 한 반영구적으로 보존되는데, 이 정보들을 이용하면 데이터를 복구할 수 있다. 단, 캐시정보를 이용하여 데이터를 복구하는 방안은 Android와 iOS가 서로 상이하다.

표 6. 복구 가능한 데이터
Table 6. Recoverable Data List

mimeType	Recoverability	Note
text/plain	×	
audio/mp4	○	Need to Cache File
image	○	
gif/https	○	

Android의 경우 SureSpot의 Cache 폴더에는 다양한 폴더가 존재한다. 이중 디지털 포렌식 관점에서 가치있는 데이터는 file과 http 폴더에 존재한다. 두 폴더에는 최대 16bytes 길이의 16진수 문자열의 이름으로 갖는 '.0' 또는 '.1' 확장자의 파일이 존재한다. 다음은 두 폴더의 경로와 설명이다.

o /storage/emulated/0/Android/data/com.twofours.surespot/cache/file

- file 폴더에는 '.0'파일이 존재한다. 해당 파일은 사용자가 서버와의 통신을 통해 데이터를 전송할 때 생긴 파일이다. 따라서 사용자가 미디어 파일을 전송한 경우 해당 폴더에 서버에 업로드한 파일과 동일한 파일이 저장되어있다.

o /storage/emulated/0/Android/data/com.twofours.surespot/cache/http

- http 폴더에는 '.0'과 '.1' 파일이 존재한다. '.0'의 경우 서버와의 통신 과정에서 데이터를 수신하기 위해 생성된 데이터이다. 해당 파일에는 수신해야 하는 데이터정보(미디어데이터인 경우 해당 파일의 URL), 타임스탬프, TLS 정보와 같은 통신데이터가 저장되어 있다.
- '.1' 파일은 '.0' 파일 내부에 데이터 정보가 URL인 경우 해당 URL로부터 내려받은 데이터이다. 즉, '.0' 파일이 상대방으로부터 받은 미디어 데이터에 대한 정보인 경우, '.1' 파일은 상대방이 전송한 미디어 데이터와 동일한 데이터를 저장하고 있다.

즉, 해당 폴더들에 존재하는 Cache 파일은 메신저에서 주고받은 미디어 파일이 암호화된 것이다. 따라서 앞선 'Message.sss'내의 id가 비어있는 시간대와 캐시폴더 내부의 파일 생성된 시간, 서버 통신을 위한 타임스탬프의 정보 등을 토대로 삭제된 데이터의 종류를 특정 지을 수 있다.

iOS의 경우 안드로이드와 달리 하나의 폴더만 존재한다.

o 〈Application home〉/Library/Cache

iOS의 캐시 폴더에는 통신한 서버의 도메인 정보와 주고받은 데이터가 존재한다. 해당 데이터들은 서버로부터 주고받은 데이터이므로 암호화되어 있으며, Android와 다르게 통신정보를 확인하기가 어려우므로 저장되어 있는 데이터의 파일시스템상의 메타데이터를 참조하여 생성시간을 토대로 'Message.sss'와 비교분석하여 데이터를 유추할 수 있다.

획득한 데이터는 복호화 과정을 거쳐야 그 의미를 알 수 있으며, 이때 복호화에 사용되는 비밀 키는 'Secrets.sss'에 저장된 Key를 사용하면 된다. 하지만 IV는 'message.sss'에서 삭제되었으므로 값을 복구해야만 데이터의 복원이 가능하다.
AES-256-GCM 에서 데이터를 암호화하는 실질적인 부분은 AES-256-CTR모드와 동일하므로 다음 수식 (10)에 의하여 암호화에 사용된 IV(이하 counter)정보는 복구가 가능하다.

$$Decrypt_{Key}(CipherText \oplus PlainText) = IV(Counter) \tag{10}$$

이때 counter를 복구하기 위해서는 암호문 1블록에 대응하는 평문 1블록(16 bytes)이 필요하다. SureSpot은 이미지 전송 시 원본 이미지의 확장자와 무관하게 JPEG 포맷으로 변형하여 이미지를 전송하기 때문에 평문의 9블록이 고정 값이다. 음성 메시지의 경우 MP4의 형식으로 전송되기 때문에 평문의 첫 블록은 고정된 데이터이다. 〔표 7〕은 미디어 데이터의 타입별 고정된 평문의 값이다. 따라서 이를 이용하면 counter는 손쉽게 복원해 낼 수 있다. 복원된 counter와 AES-256-CTR 모드를 사용하면 삭제 미디어 데이터를 복구할 수 있다.

표 7. 미디어 메시지별 고정 값
Table 7. Fixed Data of Media Message

mimeType	Data	Bytes Offset	Decoded Text
image	FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01 00 01 00 00	0x00 ~ 0x14	ÿØÿà■┼JFIF■┌┌■■ ┌■┌■■
audio/mp4	00 00 00 18 66 74 79 70 6D 70 34 32 00 00 00 00 69 73 6F 6D 6D 70 34 32 00 00 02	0x00 ~ 0x1A	■■■ftypmp42■■ ■■isommp42■┐

Ⅶ. 결론

디지털 포렌식 수사에서 다양한 정보가 저장된 모바일 메신저는 주요 증거로 활용될 수 있다. 하지만 최근 대다수의 모바일 메신저는 개인정보보호를 위하여 데이터 암호화 기능을 지원하고 있다. 특히 종단간 암호화가 적용된 메신저의 경우 메신저 서버로부터 데이터를 획득하여도 데이터를 복호화하기 위한 키가 스마트폰 기기에만 존재하므로 원본데이터의 정보를 얻기 어렵다. 이러한 보안 기술은 범죄자에 의해 악용되는 사례가 증가함에 따라 디지털 포렌식 수사 관점에서 안티포렌식으로 작용한다. 따라서 안티포렌식 대응을 위해 암호화된 메신저 데이터에 관한 복호화 기술연구는 지속적인 관심이 필요하다.

본 논문에서는 보안 메신저 SureSpot의 암호화된 데이터에 대한 복호화 방안을 제시했다. 또한, 복호화한 데이터를 토대로 패스워드 검증 방안을 제시하였으며 이를 토대로 사용자 로그인 패스워드를 복구하는 기술을 개발했다. SureSpot의 경우 키 생성을 위해 사용되는 PBKDF2 알고리즘의 반복횟수가 1,000회로 NIST 권장 횟수 10,000회〔14〕에 비해 많이 부족했다. 따라서 충분한 컴퓨팅파워가 존재한다면 〔표 5〕 이상으로 키 복구 및 데이터 획득이 가능하다. 단, SureSpot의 경우 1,000개의 메시지만 클라이언트에 저장하고 있으므로 최근 메시지를 제외하곤 데이터 획득에 어려움이 존재했다. 하지만, 미디어 데이터의 경우 서버를 통해 주고받으면서 생성되는 캐시 데이터를 관리하지 않았기에 이를 토대로 삭제된 메시지 일부를 복구할 수 있다. 본 논문에서 제시한 방법들은 암호화된 데이터를 복호화하여 유효한 증거로 활용할 수 있을 것으로 기대된다.

참 고 문 헌 (References)

[1] “Mobile market trends”, https://resources.newzoo.com/hubfs/Reports/Newzoo_2018_Global_Mobile_Market_Report_Free.pdf.

[2] “Number of mobile messaging users worldwide,” https://www.statista.com/statistics/483255/number-of-mobile-messaging-users-worldwide/.

[3] “Encryption Technology Embraced By Terrorist”, http://cjlab.memri.org/335latest-reports/encryption-technology-embraced-by-isis-al-qaeda-other-jihadis-reaches-new-level-with-increased-dependence-on-apps-software-kik-surespot-telegram-wickr-detekt-tor-part-iv-f/.

[4] Novelino Yona Pribadi Lukito, Fazmah Arif Yulianto, Erwid Jadied, Comparison of data acquisition technique using logical extraction method on unrooted android device,in: 2016 4th International Conference on Information and CommunicationTechnology (ICoICT), IEEE, pp. 1 – 6, 2016.

[5] “Android Backup Manager”, https://developer.android.com/reference/android/app/backup/BackupManager.

[6] Myungseo Park, Hangi Kim, Jongsung Kim, How to Decrypt PIN-Based Encrypted Backup Data of Samsung Smartphones, Digital Investigation (I.F 1.771), Vol. 26, pp. 63-71, 2018.

[7] Myungseo Park, Giyoon Kim, Jongsung Kim, Decrypting password-based encrypted backupdata for Huawei smartphones, Digital Investigation 28 (2018) 119 – 125.

[8] Myungseo Park, Hangi Kim, Jongsung Kim, Study on Improved Recovery Method of LG Smartphone Backup Data, Journal of Digital Forensics, Vol. 12, pp. 1-7, 2018.

[9] Khushboo Rathi, Umit Karabiyik, Temilola Aderibigbe, Hongmei Chi, Forensic analysis of encrypted instant messaging applications on android, in: 2018 6th Interna-tional Symposium on Digital Forensic and Security (ISDFS), IEEE, pp. 1 – 6, 2018.

[10] Cosimo Anglano, Massimo Canonico, Marco Guazzone, Forensic analysis of the chatsecure instant messaging application on android smartphones, Digital investigation 19, pp. 44 – 59, 2016.

[11] Cosimo Anglano, Massimo Canonico, Marco Guazzone, Forensic analysis of telegram messenger on android smartphones, Digital Investigation 23, pp. 31 – 49, 2017.

[12] “SureSpot Encrypted Messenger”, https://www.surespot.me.

[13] “Benchmark Hashcat v5.1.0 on 10 * GTX 2080 Ti”, https://www.onlinehashcrack.com/tools-benchmark-hashcat-gtx-1080-ti-1070-ti.php

[14] NIST, “Digital Identity Guidelines-authentication and Lifecycle Management”, SP 800 – 863B, June 2017.

저 자 소 개

김 기 윤 (Giyoon Kim)
학생회원
2019년 2월: 국민대학교 수학과 졸업
2019년 3월 ~ 현재 : 국민대학교 금융정보보안학과 석사과정
관심분야 : 디지털 포렌식, 정보보호 등

허 욱 (Uk Hur)
학생회원
2019년 2월: 건국대 신소재공학과 졸업
2019년 3월 ~ 현재 : 국민대학교 금융정보보안학과 석사과정
관심분야 : 디지털 포렌식, 정보보호 등

이 세 훈 (Sehoon Lee)
학생회원
2019년 2월: 경북대 전자공학과 졸업
2019년 3월 ~ 현재 : 국민대학교 금융정보보안학과 석사과정
관심분야 : 디지털 포렌식, 정보보호 등

김 종 성 (Jongsung Kim)
정회원
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 : 디지털 포렌식, 암호 알고리즘, 정보보호 등