

세션 데이터베이스 복호화 및 주요 아티팩트 분류

2조

20232106 정유진 / 20212052 이동훈

alice2002@kookmin.ac.kr / dhsama51@kookmin.ac.kr

25.05.15

목차

1. 환경 세팅
2. 아티팩트 쌓기 / 추출
3. 아티팩트 분석
4. 어플리케이션 분석 - 기초 조사
5. 어플리케이션 분석 - Android Keystore에서 키 추출
6. 어플리케이션 분석 - DB 암호화 흐름
7. 어플리케이션 분석 - DB 복호화 흐름
8. 어플리케이션 분석 - DB 생성, open, close, 조작 흐름
9. 어플리케이션 분석 정리 / 최종 복호화
10. 아티팩트 최종 분석

환경 세팅

Nox 가상환경에서 분석

Android 12 기준

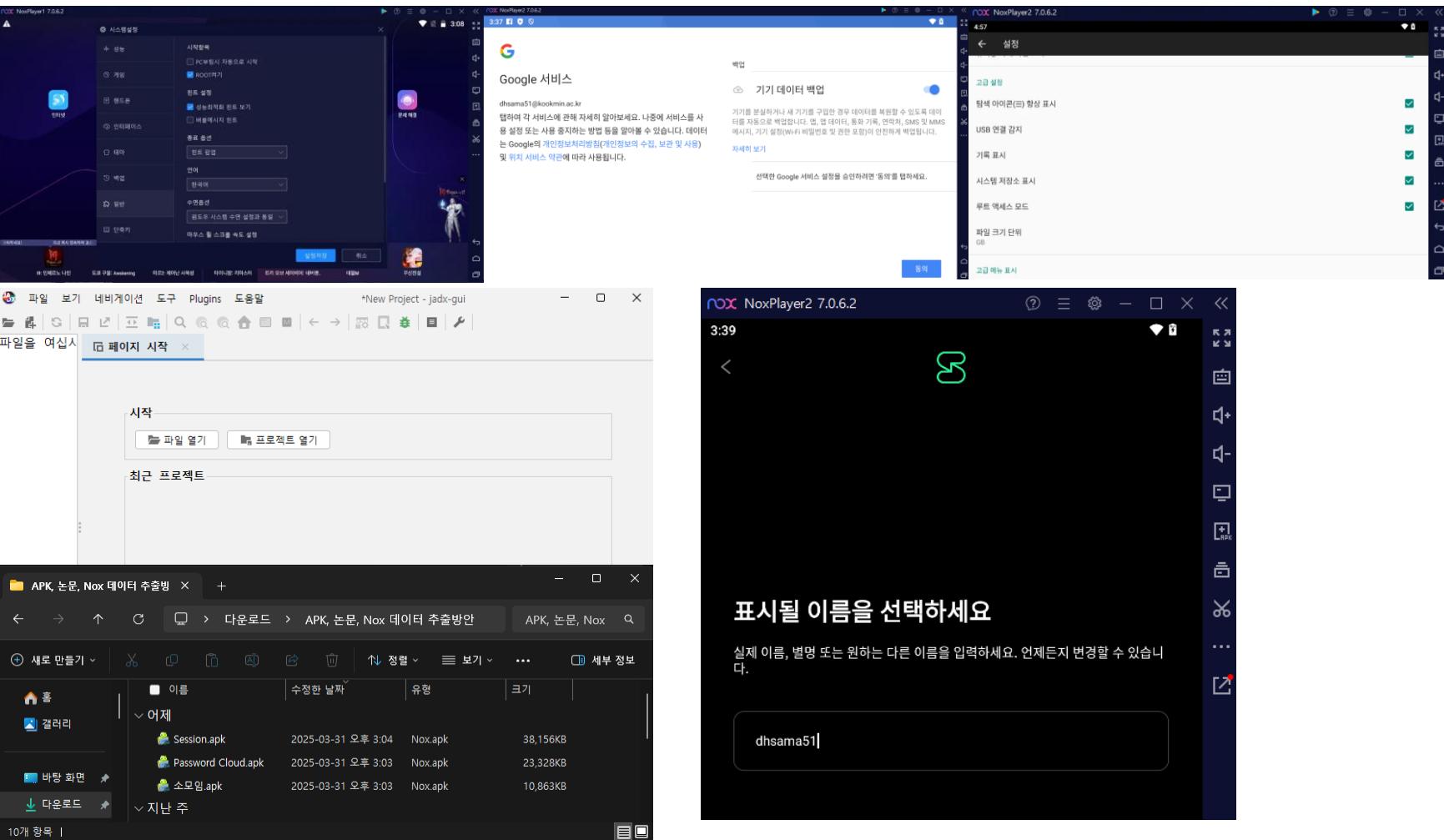
Nox, jadx, Python 사용

Nox - Root, 백업 설정

파일 탐색기 설정

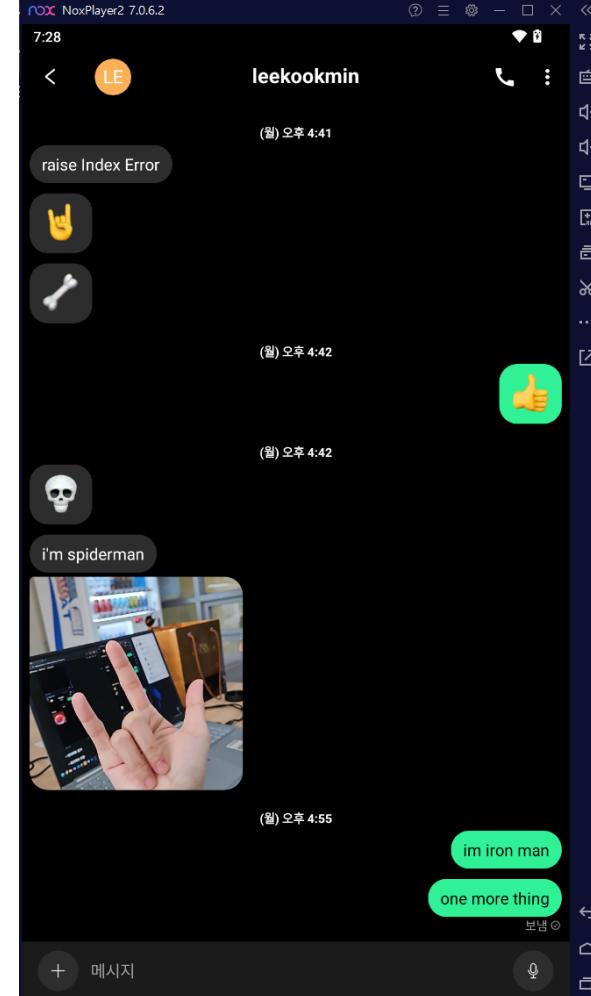
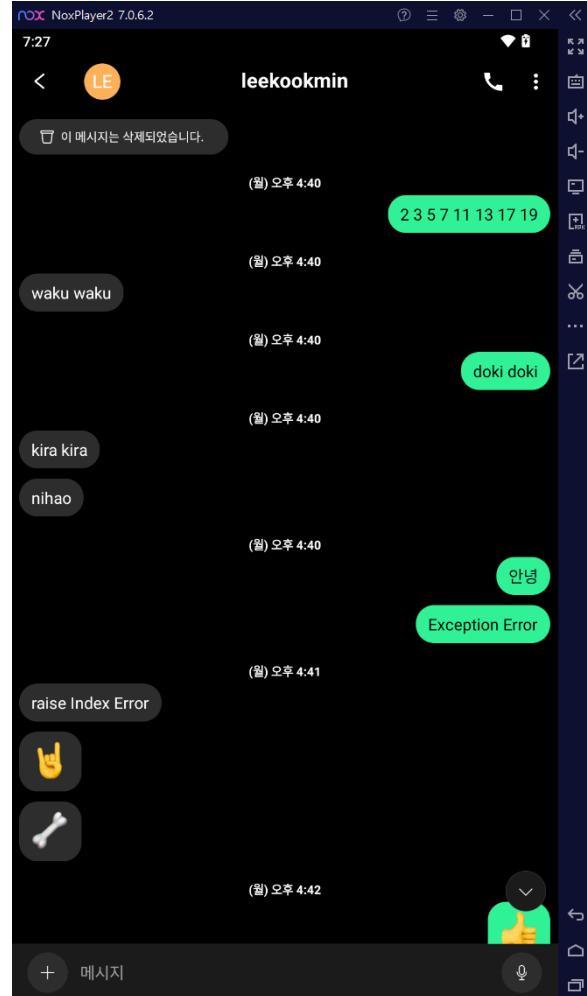
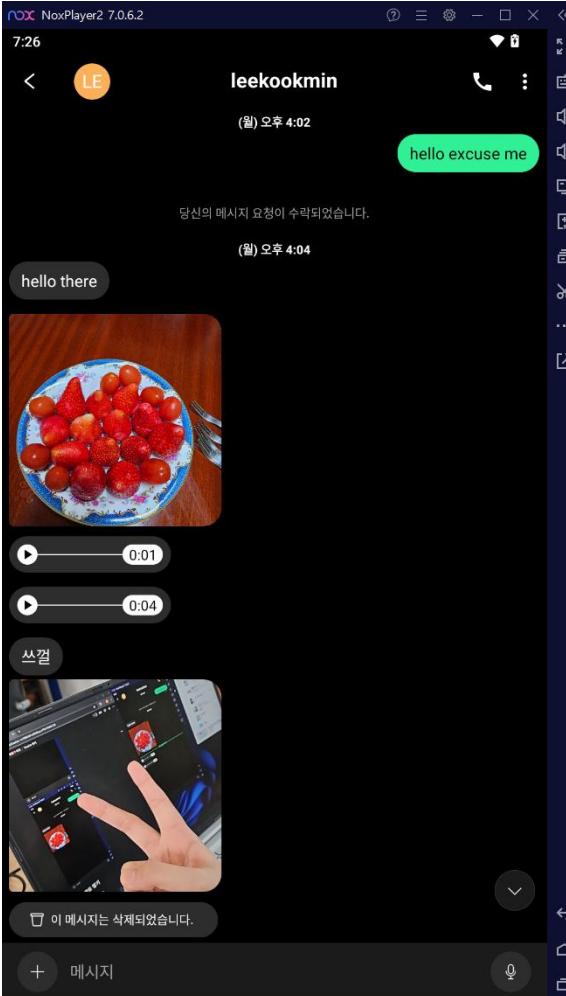
jadx - Session.apk 준비

Session 회원가입



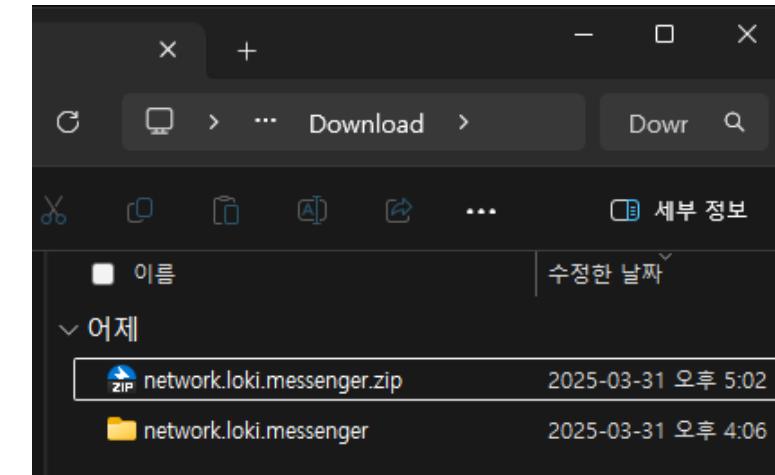
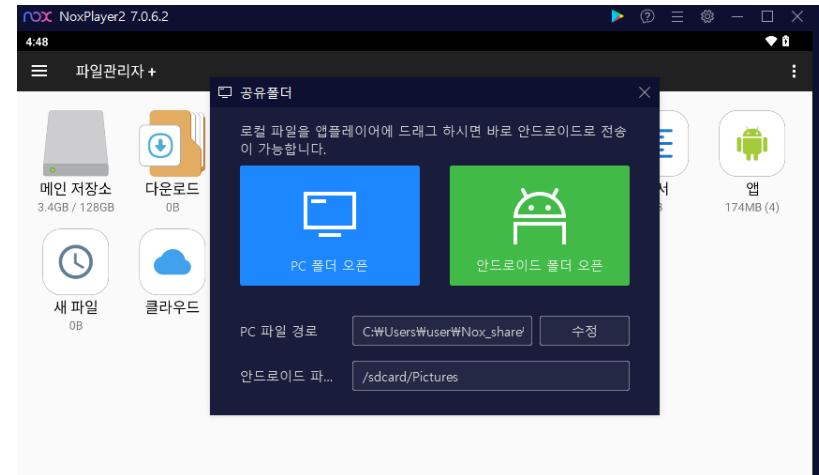
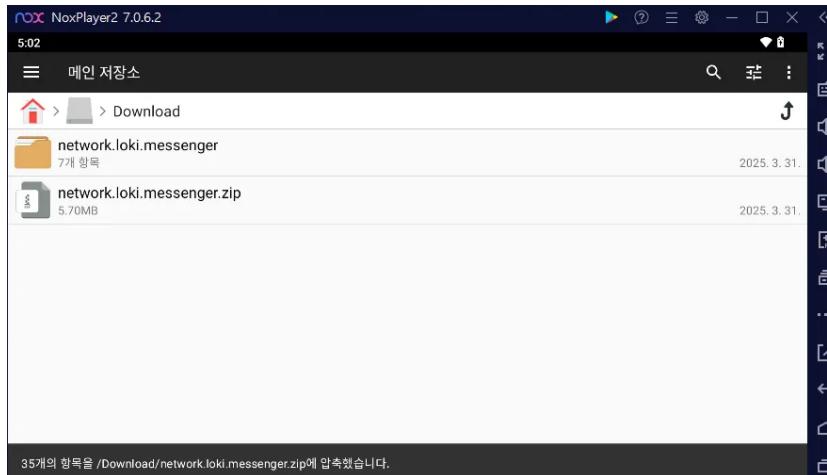
아티팩트 쌓기

Nox - 핸드폰끼리 일반 메시지, 사진, 음성 파일, 이모티콘 등 전송, 수신 후 삭제한 메시지 생성



아티팩트 추출

network.loki.messenger 폴더 Download로 복사, Windows로 이동

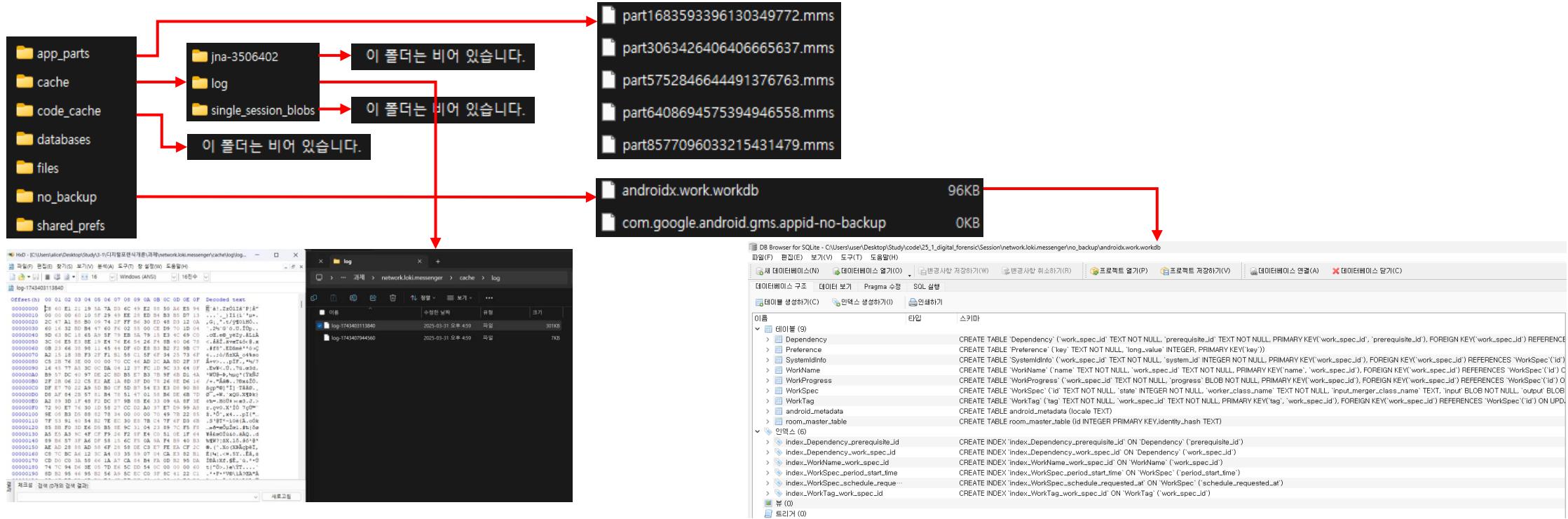


아티팩트 분석 (1)

app_parts: mms 파일 → 멀티미디어 파일(사진, 음성 파일), 암호화되어 있음

cache/log: HxD로 log 파일 암호화되어 있음을 확인

no_backup/android.work.workdb: 앱 설정과 관련된 내용. 대부분 비어 있을 것이다.



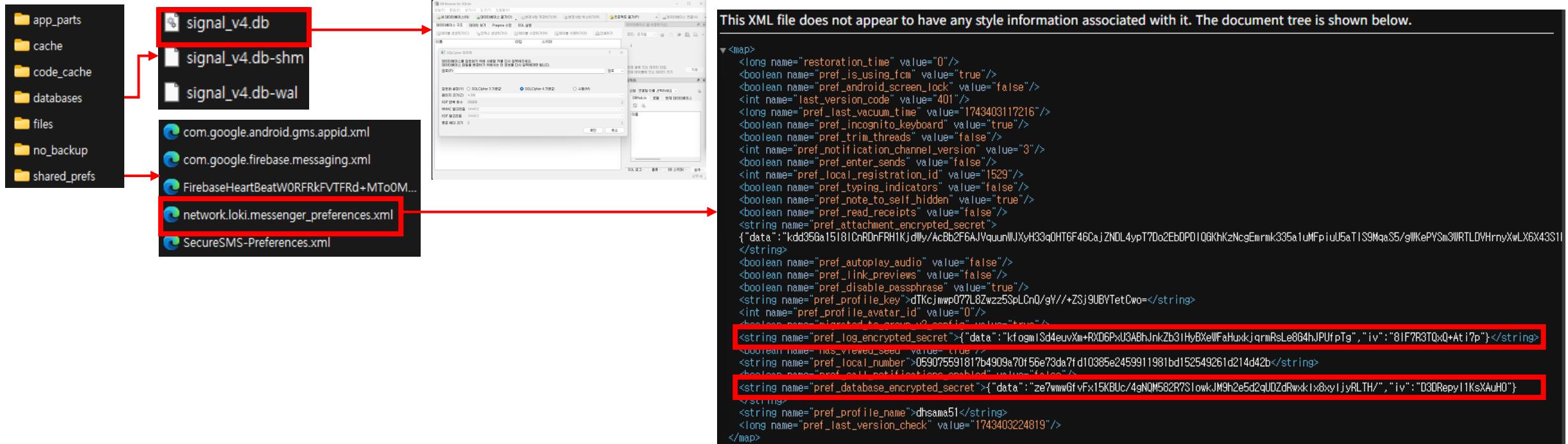
```
<string name="pref_log_encrypted_secret">
    {"data": "kfgomlSd4euVxm+RXD6PxU3ABhJnkZb3IHByXeWFaHuxkjqrnPsLe8G4hJPUpfTg", "iv": "8IF7R3T0xQ+At7p"}</string>
```

아티팩트 분석 (2)

shared_prefs/network.loki.messenger_preferences.xml: 암호 정보, 민감한 정보 포함

pref_last_vacuum_time, pref_attachment_encrypted_secret, pref_profile_key,
pref_log_encrypted_secret, pref_database_encrypted_secret, pref_profile_name

databases/signal_v4.db: 데이터베이스가 암호화되어 있음



아티팩트 분석 (3)

shared_prefs/network.loki.messenger_preferences.xml의 **pref_last_vacuum_time**과

shared_prefs/ com.google.android.gms.appid.xml의 **timestamp** 비교

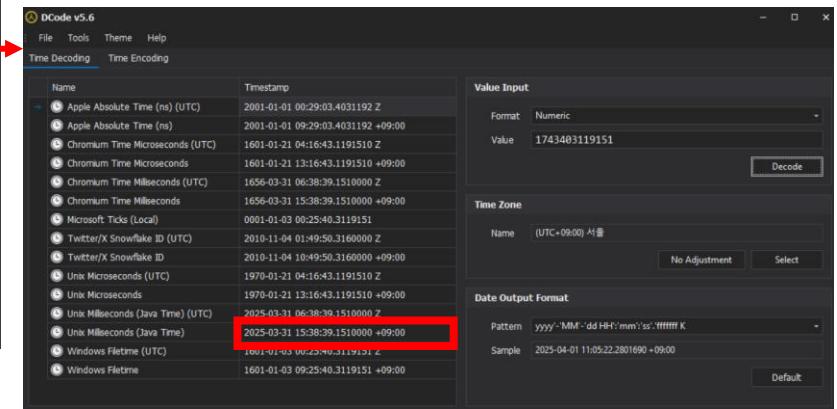
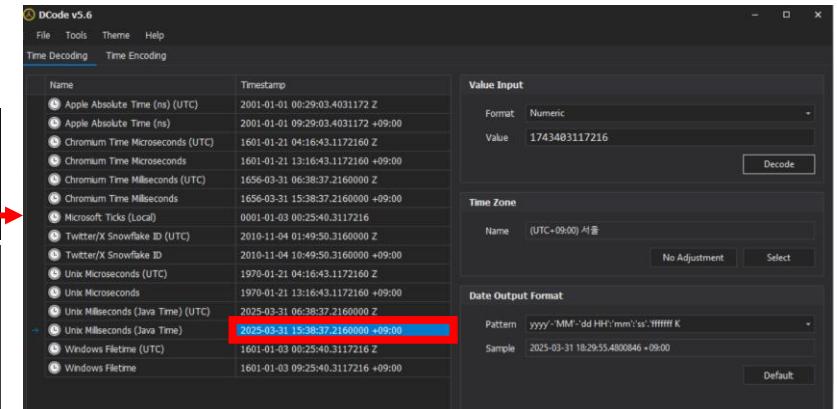
Vacuum time은 vacuum 작업(로컬 DB 최적화 관리) 시간

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<map>
    <string name="T|435124674901*">
        {"token": "cvOeFp0rTd65RZA80aCv5:APA91bHu0kCh2FrreHQ...3coll4EEPhB7t-jl-Pa24qjSxTAIIusF3b8ptWV3J88AD12XGh346y9FZzhEXBucLPLvCe_3x-63U4XZjeZjsJrxLXpDixvKEoY", "appVersion": "4015", "timestamp": "1743403119151"}</map>
```

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<map>
    <long name="restoration_time" value="0"/>
    <boolean name="pref_is_using_icm" value="true"/>
    <boolean name="pref_android_screen_lock" value="false"/>
    <int name="last_version_code" value="4015"/>
    <long name="pref_last_vacuum_time" value="1743403117216"/>
    <boolean name="pref_incorrect_scanner" value="true"/>
    <boolean name="pref_trim_threads" value="false"/>
    <int name="pref_notification_channel_version" value="3"/>
    <boolean name="pref_enter_sends" value="false"/>
    <int name="pref_local_registration_id" value="1529"/>
    <boolean name="pref_typering_indicators" value="false"/>
    <boolean name="pref_note_to_self_hidden" value="true"/>
    <boolean name="pref_read_receipt" value="false"/>
    <string name="pref_attachment_encrypted_secret">
        {"data": "kdd356a15181Cr0NfRH1Kjdly/AcB6z2FA6IAvuurWJX/H33a0HT6F46cajZNDL4ypT7D0e2Eb0PD0I06KhKzNcaEunrk335a1uMfpjuJ5aTS9Mqa35/silKePYS3iIRLDvHrryXwLX6X43S11", "v2": true}
    </string>
    <boolean name="pref_autoplay_audio" value="false"/>
    <boolean name="pref_link_preview" value="false"/>
    <boolean name="pref_disable_passphrase" value="true"/>
    <string name="pref_profile_key">dKcijmp077L0Bwzz5SpCnQ/gV//+ZSJ9UBYTetCwo=</string>
    <int name="pref_profile_avatar_id" value="0"/>
    <boolean name="migrated_to_group_v2_config" value="true"/>
    <string name="pref_log_encrypted_secret"><data>fkfom1Sd4euVm+RY06PxJ3AbhJnkZb3IHv8xlfHuxkjqrnRsLe864hJPUpTs</data>, "iv": "8IF7R3TQx+Ait1?p"</string>
    <boolean name="has_viewed_seed" value="true"/>
    <string name="pref_local_number">0507559181764909a70f56e73da7fd10385e2459911961bd152549261d214d42b</string>
    <boolean name="pref_call_notifications_enabled" value="false"/>
    <string name="pref_database_encrypted_secret"><data>:2e7wwwGvFx15kBUC/4gNDM582R7SIovkJW9h2e5d2aJ02dRwxk1x8xyjjyRLTH/</data>, "iv": "D30RepI1KsXAUh0"</string>
    <string name="pref_profile_name">dhsm51</string>
    <long name="pref_last_version_check" value="1743403224819"/>
</map>
```



어플리케이션 분석 – 기초 조사 (1)

사전 정보

주요 데이터를 데이터베이스에 보관하고, 이를 암호화
AES256/GCM/NoPadding 사용

Android Keystore의 구조를 분석하여 암호키 추출

Keymaster: Android Keystore의 모든 키를 관리하는 키.

Blob 형태로 저장되어 있으며 다음과 같은 정보가 들어있음

- ①Android Keystore 버전 ②Nonce
 - ③암호키 ④암호키 인증을 위한 Tag 및 속성 정보
- Keymaster Blob이라고 불림

보안 인스턴트 메신저 Session 데이터베이스 복호화 방안 연구

박지수*, 박세준*, 김기윤**, 김종성***

*, **, ***국민대학교(학생, 대학원생, 교수)

Decryption Methods for Secure Instant Messenger Application
Session Database

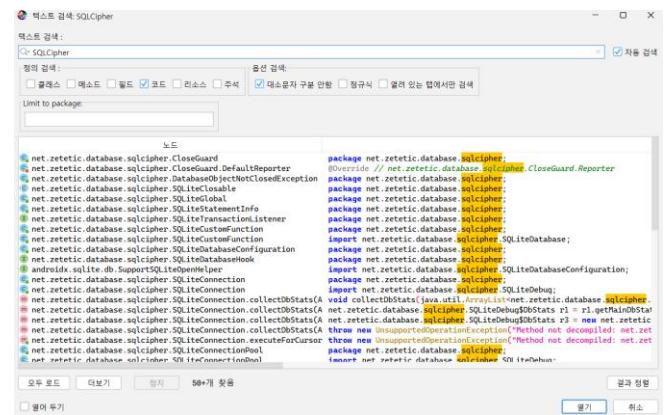
JiSu Park*, SeJun Park*, GiYoon Kim**, JongSung Kim***

*, **, ***Kookmin University(Student, Graduate student, Professor)

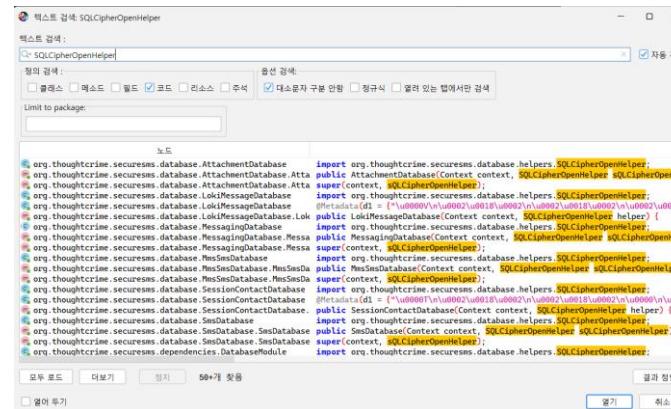
어플리케이션 분석 - 기초 조사 (2)

데이터베이스 암호화 정보 확인

SQLCipher로 암호화되어 있음 → 관련 코드 검색
암호화된 데이터베이스 이름인 signal_v4.db 확인



```
/* loaded from: classes4.dex */
public class SQLCipherOpenHelper extends SQLiteOpenHelper {
    private static final String CIPHER3_DATABASE_NAME = "signal.db";
    public static final String DATABASE_NAME = "signal_v4.db"; // This line is highlighted with a red box
    private static final int DATABASE_VERSION = 69;
    private static final int MIN_DATABASE_VERSION = 28;
    private static final String TAG = "SQLCipherOpenHelper";
    private static final int lokiV10 = 31;
    private static final int lokiV11 = 32;
    private static final int lokiV12 = 33;
    private static final int lokiV13 = 34;
    private static final int lokiV14_BACKUP_FILES = 35;
    private static final int lokiV15 = 36;
    private static final int lokiV16 = 37;
    private static final int lokiV17 = 38;
    private static final int lokiV18_CLEAR_BG_POLL_JOBS = 39;
    private static final int lokiV19 = 40;
    private static final int lokiV20 = 41;
    private static final int lokiV21 = 42;
    private static final int lokiV22 = 43;
    private static final int lokiV23 = 44;
    private static final int lokiV24 = 45;
    private static final int lokiV25 = 46;
}
```



Jadx decompiler를 통해 분석해본 결과 signal_v4.db 데이터베이스는 SQLCipher로 암호화되었다. 다음은 실제 소스코드 상에서 확인 가능한 SQLCipher를 활용한 데이터베이스 오픈 과정이다(그림 2).

SQLCipher로 암호화되었다.

어플리케이션 분석 – 기초 조사 (3)

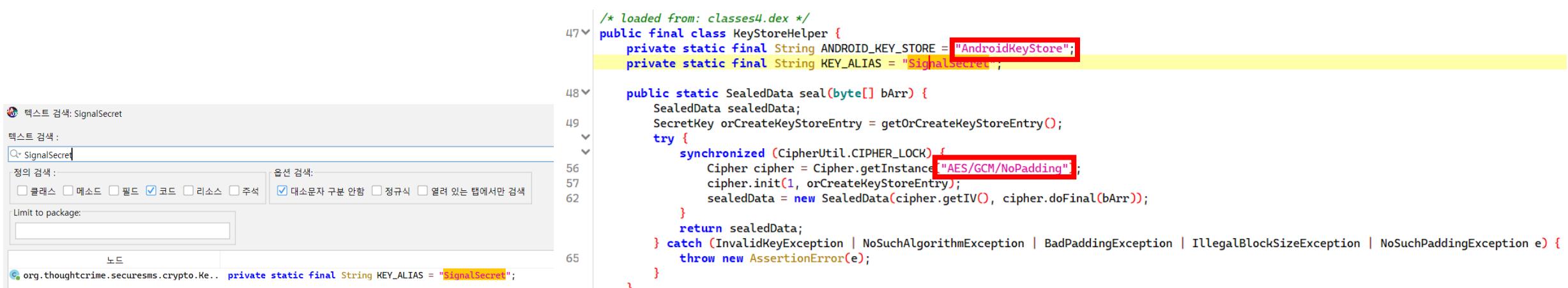
Session 암호키 정보

논문에 언급된 SignalSecret 검색 → 결과 1건

SignalSecret과 AndroidKeyStore이 단 1건 있고,

AES/GCM/NoPadding 사용을 직접 확인

SQLCipher에 사용된 패스프레이즈는 암호화되어 Shared Preference에 저장된다. 이때 암호 알고리즘은 AES/GCM/NoPadding 이었으며, 암호키는 Android Keystore에 저장되었다. Android Keystore에서 Session의 암호키를 구분하기 위한 별칭은 SignalSecret이다(그림 3).



```
/* loaded from: classes4.dex */
public final class KeyStoreHelper {
    private static final String ANDROID_KEY_STORE = "AndroidKeyStore";
    private static final String KEY_ALIAS = 'SignalSecret';

    public static SealedData seal(byte[] bArr) {
        SealedData sealedData;
        SecretKey orCreateKeyStoreEntry = getOrCreateKeyStoreEntry();
        try {
            synchronized (CipherUtil.CIPHER_LOCK) {
                Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
                cipher.init(1, orCreateKeyStoreEntry);
                sealedData = new SealedData(cipher.getIV(), cipher.doFinal(bArr));
            }
            return sealedData;
        } catch (InvalidKeyException | NoSuchAlgorithmException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
            throw new AssertionError(e);
        }
    }
}
```

텍스트 검색: SignalSecret
정의 검색 : 클래스 메소드 필드 코드 리소스 주석
옵션 검색: 대소문자 구분 안함 정규식 열려 있는 탭에서만 검색
Limit to package:
노드
org.thoughtcrime.securesms.crypto.KeyStoreHelper

어플리케이션 분석 – 기초 조사 (4)

암호키를 안전하게 관리하는 시스템

키를 생성할 때 하드웨어 보안 모듈에 저장, 없다면 소프트웨어로 구현

KeyStore.getEntry(KEY_ALIAS, null)로 접근

→ KeyEntry 내부적으로 제공(Key를 뽑아낼 수는 없음)

cipher.init(1, keyStoreEntry, ...)로 keyEntry를 이용한 실제 암호화 진행,
cipher.init(2, keyStoreEntry, ...)로 keyEntry를 이용한 실제 복호화 진행



Nox에서는 /data/misc/keystore/ 경로에 persist.sqlite를 생성하여 키 저장, 관리

→ persist.sqlite와 관련된 코드는 앱 코드에 존재하지 않고, 직접 접근도 불가

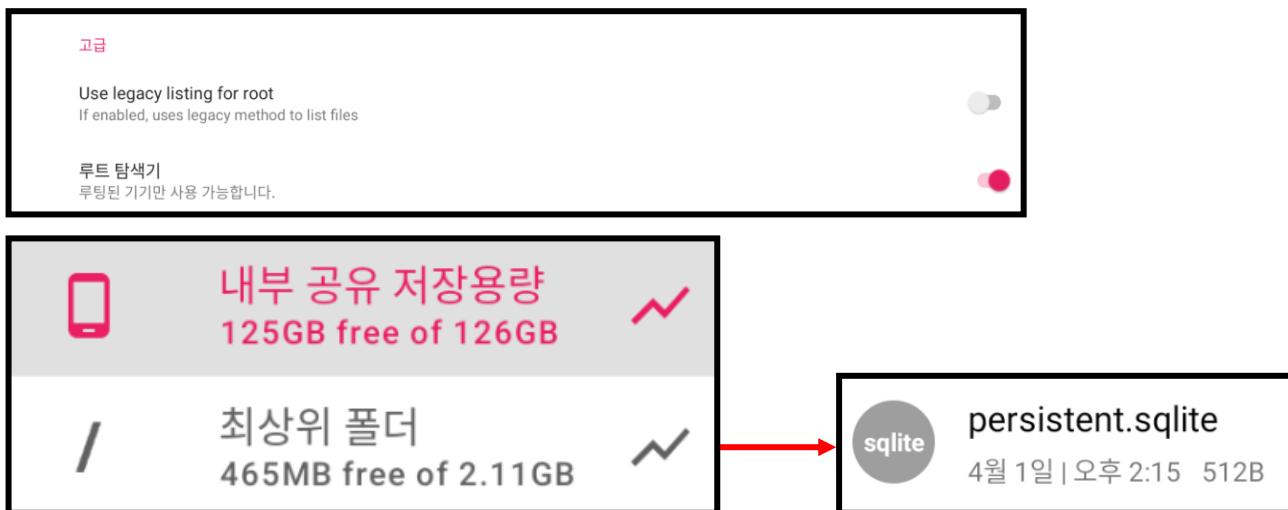
내부적으로는 keyentry와 blobentry 테이블이 생성되어 있음

지원되는 알고리즘

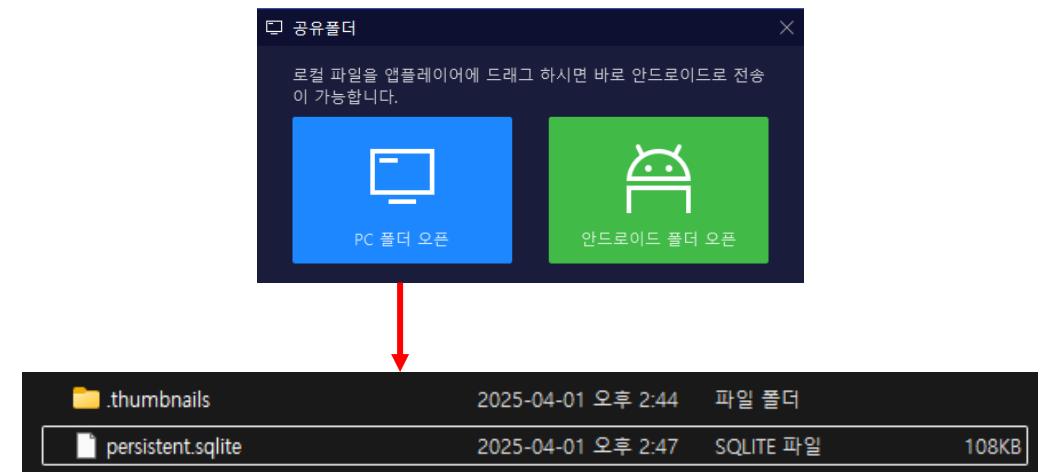
- Cipher
- KeyGenerator
- KeyFactory
- KeyStore (KeyGenerator 및 KeyPairGenerator 와 동일한 키 유형 지원)
- KeyPairGenerator
- Mac
- Signature
- SecretKeyFactory

어플리케이션 분석 – Android Keystore에서 키 추출 (1)

Amaze 파일 매니저 이용해 persist.sqlite 추출



안드로이드 12 이상 버전에서의 암호키는 USERDATA/misc/keystore내 persist.sqlite 데이터베이스에 저장된다. 해당 데이터베이스에서



어플리케이션 분석 - Android Keystore에서 키 추출 (2)

keyentry에서 SignalSecret 발견,
blobentry에서 같은 id를 가진 것 발견
0xD~0x1C에서 암호키 추출

→ '79 FA 7C 4A 90 A5 53 95 B1 1A 2B 49 7A 82 57 8D'

이름	타입	테이블(T): keyentry	테이블(T): blobentry																																																																																									
blobentry	테이블(7)	<table border="1"> <thead> <tr> <th></th> <th>id</th> <th>key_type</th> <th>domain</th> <th>namespace</th> <th>alias</th> </tr> </thead> <tbody> <tr><td>필터</td><td>필터</td><td>필터</td><td>필터</td><td>필터</td><td></td></tr> <tr><td>1</td><td>6926690968309298972</td><td>0</td><td>0</td><td>1017</td><td>boot_level_key</td></tr> <tr><td>2</td><td>-585827675954575484</td><td>0</td><td>0</td><td>10046</td><td>c5635988b7157a3a</td></tr> <tr><td>3</td><td>-1154276977243341612</td><td>0</td><td>0</td><td>10046</td><td>mobstore_encrypt</td></tr> <tr><td>4</td><td>6044086939184787821</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>5</td><td>-6262722488817596635</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>6</td><td>-1994549063571697783</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>7</td><td>-3169231379557241686</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>8</td><td>3392414123250361534</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>9</td><td>-7823036944918506034</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>10</td><td>-5423159458800941720</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>11</td><td>6940908571265112464</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>12</td><td>-9167606571973839702</td><td>0</td><td>0</td><td>10046</td><td>nearby_presence_connection_sec</td></tr> <tr><td>13</td><td>4036893587965478311</td><td>0</td><td>0</td><td>10062</td><td>SignalSecret</td></tr> </tbody> </table>		id	key_type	domain	namespace	alias	필터	필터	필터	필터	필터		1	6926690968309298972	0	0	1017	boot_level_key	2	-585827675954575484	0	0	10046	c5635988b7157a3a	3	-1154276977243341612	0	0	10046	mobstore_encrypt	4	6044086939184787821	0	0	10046	nearby_presence_connection_sec	5	-6262722488817596635	0	0	10046	nearby_presence_connection_sec	6	-1994549063571697783	0	0	10046	nearby_presence_connection_sec	7	-3169231379557241686	0	0	10046	nearby_presence_connection_sec	8	3392414123250361534	0	0	10046	nearby_presence_connection_sec	9	-7823036944918506034	0	0	10046	nearby_presence_connection_sec	10	-5423159458800941720	0	0	10046	nearby_presence_connection_sec	11	6940908571265112464	0	0	10046	nearby_presence_connection_sec	12	-9167606571973839702	0	0	10046	nearby_presence_connection_sec	13	4036893587965478311	0	0	10062	SignalSecret
	id	key_type	domain	namespace	alias																																																																																							
필터	필터	필터	필터	필터																																																																																								
1	6926690968309298972	0	0	1017	boot_level_key																																																																																							
2	-585827675954575484	0	0	10046	c5635988b7157a3a																																																																																							
3	-1154276977243341612	0	0	10046	mobstore_encrypt																																																																																							
4	6044086939184787821	0	0	10046	nearby_presence_connection_sec																																																																																							
5	-6262722488817596635	0	0	10046	nearby_presence_connection_sec																																																																																							
6	-1994549063571697783	0	0	10046	nearby_presence_connection_sec																																																																																							
7	-3169231379557241686	0	0	10046	nearby_presence_connection_sec																																																																																							
8	3392414123250361534	0	0	10046	nearby_presence_connection_sec																																																																																							
9	-7823036944918506034	0	0	10046	nearby_presence_connection_sec																																																																																							
10	-5423159458800941720	0	0	10046	nearby_presence_connection_sec																																																																																							
11	6940908571265112464	0	0	10046	nearby_presence_connection_sec																																																																																							
12	-9167606571973839702	0	0	10046	nearby_presence_connection_sec																																																																																							
13	4036893587965478311	0	0	10062	SignalSecret																																																																																							
blobentry	테이블(6)	<table border="1"> <thead> <tr> <th></th> <th>id</th> <th>subcomponent_type</th> <th>keyentryid</th> <th>blob</th> </tr> </thead> <tbody> <tr><td>필터</td><td>필터</td><td>필터</td><td>필터</td><td>필터</td></tr> <tr><td>4</td><td>4</td><td></td><td>0 -1154276977243341612</td><td>BLOB</td></tr> <tr><td>5</td><td>7</td><td></td><td>0 6044086939184787821</td><td>BLOB</td></tr> <tr><td>6</td><td>10</td><td></td><td>0 -6262722488817596635</td><td>BLOB</td></tr> <tr><td>7</td><td>13</td><td></td><td>0 -1994549063571697783</td><td>BLOB</td></tr> <tr><td>8</td><td>16</td><td></td><td>0 -3169231379557241686</td><td>BLOB</td></tr> <tr><td>9</td><td>19</td><td></td><td>0 3392414123250361534</td><td>BLOB</td></tr> <tr><td>10</td><td>22</td><td></td><td>0 -7823036944918506034</td><td>BLOB</td></tr> <tr><td>11</td><td>25</td><td></td><td>0 -5423159458800941720</td><td>BLOB</td></tr> <tr><td>12</td><td>28</td><td></td><td>0 6940908571265112464</td><td>BLOB</td></tr> <tr><td>13</td><td>31</td><td></td><td>0 -9167606571973839702</td><td>BLOB</td></tr> <tr><td>14</td><td>32</td><td></td><td>0 -4036893587965478311</td><td>BLOB</td></tr> </tbody> </table>		id	subcomponent_type	keyentryid	blob	필터	필터	필터	필터	필터	4	4		0 -1154276977243341612	BLOB	5	7		0 6044086939184787821	BLOB	6	10		0 -6262722488817596635	BLOB	7	13		0 -1994549063571697783	BLOB	8	16		0 -3169231379557241686	BLOB	9	19		0 3392414123250361534	BLOB	10	22		0 -7823036944918506034	BLOB	11	25		0 -5423159458800941720	BLOB	12	28		0 6940908571265112464	BLOB	13	31		0 -9167606571973839702	BLOB	14	32		0 -4036893587965478311	BLOB																									
	id	subcomponent_type	keyentryid	blob																																																																																								
필터	필터	필터	필터	필터																																																																																								
4	4		0 -1154276977243341612	BLOB																																																																																								
5	7		0 6044086939184787821	BLOB																																																																																								
6	10		0 -6262722488817596635	BLOB																																																																																								
7	13		0 -1994549063571697783	BLOB																																																																																								
8	16		0 -3169231379557241686	BLOB																																																																																								
9	19		0 3392414123250361534	BLOB																																																																																								
10	22		0 -7823036944918506034	BLOB																																																																																								
11	25		0 -5423159458800941720	BLOB																																																																																								
12	28		0 6940908571265112464	BLOB																																																																																								
13	31		0 -9167606571973839702	BLOB																																																																																								
14	32		0 -4036893587965478311	BLOB																																																																																								
blobmetadata	테이블(0)																																																																																											
grant	테이블(0)																																																																																											
keyentry	테이블(0)																																																																																											
keymetadata	테이블(0)																																																																																											
keyparameter	테이블(0)																																																																																											
version	테이블(0)																																																																																											
인덱스(6)	테이블(0)																																																																																											
blobentry_keyentryid_index	인덱스(1)																																																																																											
blobmetadata_blobentryid_index	인덱스(1)																																																																																											
keyentry_domain_namespace_index	인덱스(1)																																																																																											
keyentry_id_index	인덱스(1)																																																																																											
keymetadata_keyentryid_index	인덱스(1)																																																																																											
keyparameter_keyentryid_index	인덱스(1)																																																																																											

데이터베이스 셀 설정하기(C)

모드: 바이너리

0000	70	4b	4d	62	6c	6f	62	00	00	10	00	00	00	79	fa	7c	pKMblob.....v.1
0010	4a	90	a5	53	95	b1	1a	2b	49	7a	82	57	8d	00	00	00	J..S...+Iz.W...
0020	00	00	00	00	00	00	00	00	00	00	00	00	00	0c	00	00
0030	00	61	00	00	03	00	00	30	80	00	00	00	02	00	00	00	.a.....0.....
0040	10	20	00	00	01	00	00	20	00	00	00	00	01	00	00	00
0050	20	01	00	00	08	00	00	30	60	00	00	00	04	00	00	00`.....
0060	20	20	00	00	06	00	00	20	01	00	00	00	f7	01	00	00`.....
0070	70	01	bd	02	00	60	43	5c	ec	ea	95	01	00	00	be	02	p....`C\.....
0080	00	10	00	00	00	00	c1	02	00	30	c0	d4	01	00	c2	020.....
0090	00	30	dd	15	03	00	90	9b	4c	f8	38	9a	10	68		L..h.....

Keymaster Blob을 획득하기 위해서는 keyentry 테이블과 blobentry 테이블을 활용해야 한다. 먼저, 두 테이블을 연결해주는 고윳값이 필요하다. keyentry 테이블에는 alias 컬럼과 id 컬럼이 존재한다. alias 컬럼 내 Session 암호키의 별칭이 존재하는 열의 id 값이 두 테이블을 연결해주는 고윳값이 된다. 해당 값을 blobentry 테이블의 keyentryid 컬럼에서 매칭한 뒤 같은 열에 존재하는 blob 데이터가 Keymaster Blob이 된다. 해당 Keymaster Blob의 offset 0xD~0x1C에는 암호키가 평문 형태로 저장되어 있다(그림 5).

이름	타입	테이블(T): keyentry	테이블(T): blobentry																							
blobentry	테이블(6)	<table border="1"> <thead> <tr> <th></th> <th>id</th> <th>key_type</th> <th>domain</th> <th>namespace</th> <th>alias</th> <th>state</th> <th>km_uuid</th> </tr> </thead> <tbody> <tr><td>필터</td><td>필터</td><td>필터</td><td>필터</td><td>필터</td><td>필터</td><td>필터</td><td>필터</td></tr> <tr><td>1313877684582408609</td><td>0</td><td>0</td><td>0</td><td>10063</td><td>SignalSecret</td><td>1</td><td>BLOB</td></tr> </tbody> </table>		id	key_type	domain	namespace	alias	state	km_uuid	필터	1313877684582408609	0	0	0	10063	SignalSecret	1	BLOB							
	id	key_type	domain	namespace	alias	state	km_uuid																			
필터	필터	필터	필터	필터	필터	필터	필터																			
1313877684582408609	0	0	0	10063	SignalSecret	1	BLOB																			
blobmetadata	테이블(0)																									
grant	테이블(0)																									
keyentry	테이블(0)																									
keymetadata	테이블(0)																									
keyparameter	테이블(0)																									
version	테이블(0)																									
인덱스(6)	테이블(0)																									
blobentry_keyentryid_index	인덱스(1)																									
blobmetadata_blobentryid_index	인덱스(1)																									
keyentry_domain_namespace_index	인덱스(1)																									
keyentry_id_index	인덱스(1)																									
keymetadata_keyentryid_index	인덱스(1)																									
keyparameter_keyentryid_index	인덱스(1)																									

(그림 3) Android Keystore Key

어플리케이션 분석 – DB 암호화 흐름 (1)

1. signal_v4.db를 암호화할 DatabaseSecret 생성

```
private DatabaseSecret createAndStoreDatabaseSecret(Context context) {
    byte[] bArr = new byte[32];
    Util.SECURE_RANDOM.nextBytes(bArr);
    DatabaseSecret databaseSecret = new DatabaseSecret(bArr);
    TextSecurePreferences.setDatabaseEncryptedSecret(context, KeyStoreHelper.seal(databaseSecret.asBytes()).serialize());
    return databaseSecret;
}
```

SECRET_RANDOM으로 32byte(=256bit) 난수를 생성하고, DatabaseSecret에 저장

```
public class DatabaseSecret {
    private final String encoded;
    private final byte[] key;

    public DatabaseSecret(byte[] bArr) {
        this.key = bArr;
        this.encoded = Hex.toStringCondensed(bArr);
    }

    public DatabaseSecret(String str) throws IOException {
        this.key = Hex.fromStringCondensed(str);
        this.encoded = str;
    }

    public String asString() {
        return this.encoded;
    }

    public byte[] asBytes() {
        return this.key;
    }
}
```

난수를 key와 encoded로 중복 저장

어플리케이션 분석 – DB 암호화 흐름 (2)

2. DatabaseSecret의 data를 AES/GCM/NoPadding 방식으로 암호화

```
private DatabaseSecret createAndStoreDatabaseSecret(Context context) {
    byte[] bArr = new byte[32];
    Util.SECURE_RANDOM.nextBytes(bArr);
    DatabaseSecret databaseSecret = new DatabaseSecret(bArr);
    TextSecurePreferences.setDatabaseEncryptedSecret(context, KeyStoreHelper.seal(databaseSecret.asBytes(), serialize()));
    return databaseSecret;
}

/* loaded from: classes4.dex */
public final class KeyStoreHelper {
    private static final String ANDROID_KEY_STORE = "AndroidKeyStore";
    private static final String KEY_ALIAS = "SignalSecret";

    public static SealedData seal(byte[] bArr) {
        SealedData sealedData;
        SecretKey orCreateKeyStoreEntry = getOrCreateKeyStoreEntry();
        try {
            synchronized (CipherUtil.CIPHER_LOCK) {
                Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
                cipher.init(1, orCreateKeyStoreEntry);
                sealedData = new SealedData(cipher.getIV(), cipher.doFinal(bArr));
            }
            return sealedData;
        } catch (InvalidKeyException | NoSuchAlgorithmException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
            throw new AssertionError(e);
        }
    }

    private static SecretKey getOrCreateKeyStoreEntry() {
        return hasKeyStoreEntry() ? getKeyStoreEntry() : createKeyStoreEntry();
    }

    private static SecretKey createKeyStoreEntry() {
        try {
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES", ANDROID_KEY_STORE);
            keyGenerator.init(new KeyGenParameterSpec.Builder(KEY_ALIAS, 3).setBlockModes(CipherPackage.GCM).setEncryptionPaddings("NoPadding").build());
            return keyGenerator.generateKey();
        } catch (InvalidAlgorithmParameterException | NoSuchAlgorithmException | NoSuchProviderException e) {
            throw new AssertionError(e);
        }
    }

    private static SecretKey getKeyStoreEntry() throws UnrecoverableKeyException {
        KeyStore keyStore = getKeyStore();
        try {
            try {
                return (KeyStore.SecretKeyEntry) keyStore.getEntry(KEY_ALIAS, null).getSecretKey();
            } catch (KeyStoreException e) {
                e = e;
                throw new AssertionError(e);
            } catch (NoSuchAlgorithmException e2) {
                e = e2;
                throw new AssertionError(e);
            } catch (UnrecoverableKeyException e3) {
                throw e3;
            } catch (UnrecoverableEntryException e4) {
                e = e4;
                throw new AssertionError(e);
            }
        } catch (UnrecoverableKeyException unused) {
            return getSecretKey(keyStore);
        }
    }
}
```

DatabaseSecret을 암호화할 키(=KEY_ALIAS=SignalSecret)가 이미 있으면 가져오고, 없으면 Android Keystore에서 새로 생성함
Cipher라는 java 표준 라이브러리를 이용하며, Android Keystore에서 연산 수행

어플리케이션 분석 – DB 암호화 흐름 (3)

3. DatabaseSecret을 serialize함

```
private DatabaseSecret createAndStoreDatabaseSecret(Context context) {
    byte[] bArr = new byte[32];
    Util.Secure_RANDOM.nextBytes(bArr);
    DatabaseSecret databaseSecret = new DatabaseSecret(bArr);
    TextSecurePreferences.setDatabaseEncryptedSecret(context, KeyStoreHelper.seal(databaseSecret.asBytes(), serialize());
    return databaseSecret;
}
```

iv와 암호화된 data를 serialize하여 JSON 형태로 변환

```
public String serialize() {
    try {
        return JsonUtil.toJsonThrows(this);
    } catch (IOException e) {
        throw new AssertionError(e);
    }
}

SealedData(bvte[] bArr, byte[] bArr2) {
    this.iv = bArr;
    this.data = bArr2;
}
```

SealedData를 serialize하여 반환

```
private static class ByteArraySerializer extends JsonSerializer<byte[]> {
    private ByteArraySerializer() {
    }

    @Override // com.fasterxml.jackson.databind.JsonSerializer
    public void serialize(byte[] bArr, JsonGenerator jsonGenerator, SerializerProvider serializerProvider) throws IOException {
        jsonGenerator.writeString(Base64.encodeToString(bArr, 3));
    }
}
```

str을 감지하면 호출되어 Base64 인코딩을 진행함

어플리케이션 분석 – DB 암호화 흐름 (4)

4. serialize 결과인 JSON 문자열을 SharedPreferences에 저장

```
private DatabaseSecret createAndStoreDatabaseSecret(Context context) {
    byte[] bArr = new byte[32];
    Util.SECURE_RANDOM.nextBytes(bArr);
    DatabaseSecret databaseSecret = new DatabaseSecret(bArr);
    TextSecurePreferences.setDatabaseEncryptedSecret(context, KeyStoreHelper.seal(databaseSecret.asBytes()).serialize());
    return databaseSecret;
}
```

```
public void setDatabaseEncryptedSecret(String secret) {
    Intrinsiccs.checkNotNullParameter(secret, "secret");
    setStringPreference("pref_database_encrypted_secret", secret);
}
```

```
public final void setStringPreference(Context context, String key, String value) {
    Intrinsiccs.checkNotNullParameter(context, "context");
    PreferenceManager.getDefaultSharedPreferences(context).edit().putString(key, value).apply();
}
```

매개변수 secret의 값을 pref_database_encrypted_secret에 저장

```
<string name="pref_database_encrypted_secret">
{"data":"ze7wmwGfvFx15KBUC/4gNQM582R7SIowkJM9h2e5d2qUDZdRwxkIx8xyIjyRLTH/","iv":"D3DRepyI1KsXAuH0"}</string>
```

어플리케이션 분석 - DB 복호화 흐름 (1)

1. SharedPreferences에 저장된 암호화된 패스프레이즈(JSON 문자열에 포함)를 로드

```
public DatabaseSecret getOrCreateDatabaseSecret() {
    String databaseUnencryptedSecret = TextSecurePreferences.getDatabaseUnencryptedSecret(this.context);
    String databaseEncryptedSecret = TextSecurePreferences.getDatabaseEncryptedSecret(this.context);
    if (databaseUnencryptedSecret != null) {
        return getUnencryptedDatabaseSecret(this.context, databaseUnencryptedSecret);
    }
    if (databaseEncryptedSecret != null) {
        return getEncryptedDatabaseSecret(databaseEncryptedSecret);
    }
    return createAndStoreDatabaseSecret(this.context);
}

private DatabaseSecret getUnencryptedDatabaseSecret(Context context, String str) {
    try {
        DatabaseSecret databaseSecret = new DatabaseSecret(str);
        TextSecurePreferences.setDatabaseEncryptedSecret(context, KeyStoreHelper.seal(databaseSecret.asBytes()).serialize());
        TextSecurePreferences.setDatabaseUnencryptedSecret(context, null);
        return databaseSecret;
    } catch (IOException e) {
        throw new AssertionError(e);
    }
}

private DatabaseSecret getEncryptedDatabaseSecret(String str) {
    return new DatabaseSecret(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str)));
}

private DatabaseSecret createAndStoreDatabaseSecret(Context context) {
    byte[] bArr = new byte[32];
    Util.SECURE_RANDOM.nextBytes(bArr);
    DatabaseSecret databaseSecret = new DatabaseSecret(bArr);
    TextSecurePreferences.setDatabaseEncryptedSecret(context, KeyStoreHelper.seal(databaseSecret.asBytes()).serialize());
    return databaseSecret;
}
```

```
<string name="pref_database_encrypted_secret">
    {"data": "ze7wmwGfvFx15KBUC/4gNQM582R7StowkJM9h2e5d2qUDZdRwxkIx8xyIjyRLTH/", "iv": "D3DRepyI1KsXAuHO"}</string>
```

Input : pref_database_encrypted_secret, Android_KeyStore_Key
Output : Passphrase

```
1 : IV ← Base64_Decode(pref_database_encrypted_secret.iv)
2 : Data ← Base64_Decode(pref_database_encrypted_secret.data)
3 : Tag ← Data[-16:]
4 : Encrypted_key ← Data[0:48]
5 : Passphrase ← AES256/GCM/NoPadding_Decrypt
    (Android_Keystore_Key, IV, Encrypted_key, Tag)
6 : return Passphrase
```

(그림 5) Getting PassPhrase of signal_v4.db

```
public String getDatabaseUnencryptedSecret() {
    return getStringPreference("pref_database_unencrypted_secret", null);
}
```

pref_database_unencrypted_secret이 없으므로 실패

```
public String getDatabaseEncryptedSecret() {
    return getStringPreference("pref_database_encrypted_secret", null);
}

public final String getStringPreference(Context context, String key, String defaultValue) {
    Intrinsic.checkNotNullParameter(context, "context");
    Intrinsic.checkNotNullParameter(key, "key");
    return PreferenceManager.getDefaultSharedPreferences(context).getString(key, defaultValue);
}
```

pref_database_encrypted_secret에서 JSON 문자열 로드

어플리케이션 분석 – DB 복호화 흐름 (2)

2. JSON 문자열(Base64 형태)을 SealedData(str 형태)로 deserialize

```
private DatabaseSecret getEncryptedDatabaseSecret(String str) {
    return new DatabaseSecret(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str)));
}
```

```
public static SealedData fromString(String str) {
    try {
        return (SealedData) JsonUtil.fromJson(str, SealedData.class);
    } catch (IOException e) {
        throw new AssertionError(e);
    }
}
```

JSON 문자열 → SealedData의 data, iv로 값을 넣어줌
data는 암호화된 패스프레이즈, iv는 data 암호화에 사용된 값

Input : pref_database_encrypted_secret, Android_KeyStore_Key
OutPut : Passphrase

- 1 : IV ← Base64_Decode(pref_database_encrypted_secret.iv)
- 2 : Data ← Base64_Decode(pref_database_encrypted_secret.data)
- 3 : Tag ← Data[-16:]
- 4 : Encrypted_key ← Data[0:48]
- 5 : Passphrase ← AES256/GCM/NoPadding_Decrypt
(Android_Keystore_Key, IV, Encrypted_key, Tag)
- 6 : return Passphrase

```
SealedData(byte[] bArr, byte[] bArr2) {
    this.iv = bArr;
    this.data = bArr2;
}
```

```
private static class ByteArrayDeserializer extends JsonDeserializer<byte[]> {
    private ByteArrayDeserializer() {
    }

    @Override // com.fasterxml.jackson.databind.JsonDeserializer
    public byte[] deserialize(JsonParser jsonParser, DeserializationContext deserializationContext) throws IOException {
        return Base64.decode(jsonParser.getValueAsString(), 3);
    }
}
```

byte[]임을 감지하면 호출되어 Base64 디코딩을 통해 str로 바꿔 줌

어플리케이션 분석 - DB 복호화 흐름 (3)

3. SealedData 내 iv와 data를 AES/GCM/NoPadding 방식으로 복호화

```
private DatabaseSecret getEncryptedDatabaseSecret(String str) {  
    return new DatabaseSecret(KeyStoreHelper.unseal(KeyStoreHelper.SealedData.fromString(str)));  
}
```

```
public static byte[] unseal(SealedData sealedData) {  
    byte[] doFinal;  
    SecretKey keyStoreEntry = getKeyStoreEntry();  
    try {  
        synchronized (CipherUtil.CIPHER_LOCK) {  
            Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");  
            cipher.init(2, keyStoreEntry, new GCMParameterSpec(128, sealedData.iv));  
            doFinal = cipher.doFinal(sealedData.data);  
        }  
        return doFinal;  
    } catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException |  
    throw new AssertionError(e);  
}
```

```
private static SecretKey getKeyStoreEntry() {  
    KeyStore keyStore = getKeyStore();  
    try {  
        try {  
            return getSecretKey(keyStore);  
        } catch (UnrecoverableKeyException e) {  
            throw new AssertionError(e);  
        }  
    } catch (UnrecoverableKeyException unused) {  
        return getSecretKey(keyStore);  
    }  
}
```

Input : pref_database_encrypted_secret, Android_KeyStore_Key
OutPut : Passphrase
1 : IV ← Base64_Decode(pref_database_encrypted_secret.iv)
2 : Data ← Base64_Decode(pref_database_encrypted_secret.data)
3 : Tag ← Data[-16:]
4 : Encrypted_key ← Data[0:48]
5 : Passphrase ← AES256/GCM/NoPadding_Decrypt
(Android_Keystore_Key, IV, Encrypted_key, Tag)
6 : return Passphrase

```
private static SecretKey getSecretKey(KeyStore keyStore) throws UnrecoverableKeyException {  
    try {  
        return ((KeyStore.SecretKeyEntry) keyStore.getEntry(KEY_ALIAS, null)).getSecretKey();  
    } catch (KeyStoreException e) {  
        e = e;  
        throw new AssertionError(e);  
    } catch (NoSuchAlgorithmException e2) {  
        e = e2;  
        throw new AssertionError(e);  
    } catch (UnrecoverableKeyException e3) {  
        throw e3;  
    } catch (UnrecoverableEntryException e4) {  
        e = e4;  
        throw new AssertionError(e);  
    }  
}
```

```
/* loaded from: classes4.dex */  
public final class KeyStoreHelper {  
    private static final String ANDROID_KEY_STORE = "AndroidKeyStore";  
    private static final String KEY_ALIAS = "SignalSecret";  
}
```

SealedData를 복호화, new DatabaseSecret로 선언한 변수에 저장
getKeyStoreEntry()에서 복호화할 키(=KEY_ALIAS=SignalSecret)를 가져옴
Cipher라는 java 표준 라이브러리를 이용하여, Android Keystore에서 연산을 수행

어플리케이션 분석 – DB 생성, open, close, 조작 (1)

1. signal_v4.db를 생성해서 open하기

```
public final SQLCipherOpenHelper provideOpenHelper(@ApplicationContext Context context) {
    Intrinsic.checkNotNullParameter(context, "context");
    DatabaseSecret orCreateDatabaseSecret = new DatabaseSecretProvider(context).getOrCreateDatabaseSecret();
    SQLCipherOpenHelper.migrateSqlCipher3To4IfNeeded(context, orCreateDatabaseSecret);
    return new SQLCipherOpenHelper(context, orCreateDatabaseSecret);
}

public SQLCipherOpenHelper(Context context, DatabaseSecret databaseSecret) {
    super(context, DATABASE_NAME, databaseSecret.asString(), (SQLiteDatabase.CursorFactory) null, 69, 28, (SQLiteDatabase.CursorFactory) null);
    @Override // net.zetetic.database.sqlcipher.SQLiteDatabaseHook
    public void preKey(SQLiteConnection sQLiteConnection) {
        SQLCipherOpenHelper.applySQLCipherPragmas(sQLiteConnection, true);
    }

    @Override // net.zetetic.database.sqlcipher.SQLiteDatabaseHook
    public void postKey(SQLiteConnection sQLiteConnection) {
        SQLCipherOpenHelper.applySQLCipherPragmas(sQLiteConnection, true);
        if (System.currentTimeMillis() - TextSecurePreferences.getLastVacuumTime(context) > 604800000) {
            sQLiteConnection.execute("VACUUM", null, null);
            TextSecurePreferences.setLastVacuumNow(context);
        }
    }
    this.context = context.getApplicationContext();
    this.databaseSecret = databaseSecret;
}

public SQLiteOpenHelper(Context context, String str, byte[] bArr, SQLiteDatabase.CursorFactory cursorFactory, int i, DatabaseErrorHandler databaseErrorHandler, SQLiteDatabaseHook sQLiteDatabaseHook) {
    if (i < 1) {
        throw new IllegalArgumentException("Version must be >= 1, was " + i);
    }
    this.mContext = context;
    this mName = str;
    this.mPassword = bArr;
    this.mFactory = cursorFactory;
    this.mNewVersion = i;
    this.mErrorHandler = databaseErrorHandler;
    this.mDatabaseHook = sQLiteDatabaseHook;
    this.mEnableWriteAheadLogging = z;
    this.mMinimumSupportedVersion = Math.max(0, i2);
}

public static SQLiteDatabase openDatabase(String str, byte[] bArr, CursorFactory cursorFactory, int i, DatabaseErrorHandler databaseErrorHandler, SQLiteDatabaseHook sQLiteDatabaseHook) {
    SQLiteDatabase sQLiteDatabase = new SQLiteDatabase(str, bArr, i, cursorFactory, databaseErrorHandler, sQLiteDatabaseHook);
    sQLiteDatabase.open();
    return sQLiteDatabase;
}
```

The diagram illustrates the flow of code execution:

- The process starts with the `provideOpenHelper` method, which creates a `DatabaseSecret` and calls `migrateSqlCipher3To4IfNeeded`.
- It then returns a new instance of `SQLCipherOpenHelper` with the provided context and secret.
- This instance is passed to the constructor of `SQLCipherOpenHelper`, which performs several setup steps:
 - Registers hooks for `preKey` and `postKey` events.
 - Updates `TextSecurePreferences` to set the last vacuum time if it's been 60 days since the last vacuum.
 - Sets the application context and database secret.
- The instance is then passed to the `SQLiteOpenHelper` constructor, which initializes the context, database name, password, cursor factory, version, error handler, and database hook.
- Finally, the `openDatabase` static method is called, creating a new `SQLiteDatabase` object with the given parameters and opening it.

어플리케이션 분석 – DB 생성, open, close, 조작 (2)

2. signal_v4.db close하기

```
void endTransaction(SQLiteDatabase SQLiteDatabase) {  
    SQLiteDatabase.setTransactionSuccessful();  
    SQLiteDatabase.endTransaction();  
}
```

```
public void endTransaction() {  
    acquireReference();  
    try {  
        getThreadSession().endTransaction(null);  
    } finally {  
        releaseReference();  
    }  
}
```

3. signal_v4.db sms 테이블 생성하기

```
public void onCreate(SQLiteDatabase SQLiteDatabase) {  
    SQLiteDatabase.execSQL(SmsDatabase.CREATE_TABLE);  
    SQLiteDatabase.execSQL(MmsDatabase.CREATE_TABLE);  
    SQLiteDatabase.execSQL(AttachmentDatabase.CREATE_TABLE);  
    SQLiteDatabase.execSQL(ThreadDatabase.CREATE_TABLE);  
    SQLiteDatabase.execSQL(DraftDatabase.CREATE_TABLE);  
    SQLiteDatabase.execSQL(PushDatabase.CREATE_TABLE);  
    SQLiteDatabase.execSQL(GroupDatabase.CREATE_TABLE);  
    SQLiteDatabase.execSQL(RecipientDatabase.CREATE_TABLE);  
    SQLiteDatabase.execSQL(GroupReceiptDatabase.CREATE_TABLE);  
    for (String str : SearchDatabase.CREATE_TABLE) {  
        SQLiteDatabase.execSQL(str);  
    }  
}
```

```
public class SmsDatabase extends MessagingDatabase {  
    ...  
    public static final String CREATE_TABLE = "CREATE TABLE sms (_id integer PRIMARY KEY, thread_id INTEGER, address TEXT,  
    address_device_id INTEGER DEFAULT 1, person INTEGER, date INTEGER, date_sent INTEGER,  
    protocol INTEGER, read INTEGER DEFAULT 0, status INTEGER DEFAULT -1, type INTEGER,  
    reply_path_present INTEGER, delivery_receipt_count INTEGER DEFAULT 0, subject TEXT, body TEXT,  
    mismatched_identities TEXT DEFAULT NULL, service_center TEXT, subscription_id INTEGER DEFAULT -1,  
    expires_in INTEGER DEFAULT 0, expire_started INTEGER DEFAULT 0, notified DEFAULT 0,  
    read_receipt_count INTEGER DEFAULT 0, unidentified INTEGER DEFAULT 0,  
    is_deleted GENERATED ALWAYS AS ((type & 31) IN (28, 19)) VIRTUAL";  
}
```

어플리케이션 분석 – DB 생성, open, close, 조작 (3)

4. signal_v4.db sms 테이블에 insert

```
public long insertMessageOutbox(long j, OutgoingTextMessage outgoingTextMessage, boolean z, long j2, InsertListener insertListener, boolean z2) {
    long j3 = outgoingTextMessage.isSecureMessage() ? 10485782L : 22L;
    if (z) {
        j3 |= 64;
    }
    if (outgoingTextMessage.isOpenGroupInvitation()) {
        j3 |= 16384;
    }
    Address address = outgoingTextMessage.getRecipient().getAddress();
    Map<Address, Long> remove = earlyDeliveryReceiptCache.remove(j2);
    Map<Address, Long> remove2 = earlyReadReceiptCache.remove(j2);
    ContentValues contentValues = new ContentValues(6);
    contentValues.put("address", address.getAddress());
    contentValues.put("thread_id", Long.valueOf(j));
    contentValues.put("body", outgoingTextMessage.getMessageBody());
    contentValues.put("date", Long.valueOf(SnodeAPI.getNowWithOffset()));
    contentValues.put("date_sent", Long.valueOf(outgoingTextMessage.getSentTimestampMillis()));
    contentValues.put("read", (Integer) 1);
    contentValues.put("type", Long.valueOf(j3));
    contentValues.put(MmsSmsColumns.SUBSCRIPTION_ID, Integer.valueOf(outgoingTextMessage.getSubscriptionId()));
    contentValues.put("expires_in", Long.valueOf(outgoingTextMessage.getExpiresIn()));
    contentValues.put(MmsSmsColumns.EXPIRE_STARTED, Long.valueOf(outgoingTextMessage.getExpireStartedAt()));
    contentValues.put("delivery_receipt_count", Long.valueOf(Stream.of(remove.values()).mapToLong(new ToLongFunction() { // from class: org.thoughtcrime.securesms
        @Override // com.annimon.stream.function.ToIntFunction
        public final long applyAsLong(Object obj) {
            long longValue;
            longValue = ((Long) obj).longValue();
            return longValue;
        }
    }).sum()));
    contentValues.put("read_receipt_count", Long.valueOf(Stream.of(remove2.values()).mapToLong(new ToLongFunction() { // from class: org.thoughtcrime.securesms
        @Override // com.annimon.stream.function.ToIntFunction
        public final long applyAsLong(Object obj) {
            long longValue;
            longValue = ((Long) obj).longValue();
            return longValue;
        }
    }).sum())));
    if (isDuplicate(outgoingTextMessage, j)) {
        Log.w(TAG, "Duplicate message (" + outgoingTextMessage.getSentTimestampMillis() + "), ignoring...");
        return -1L;
    }
    long insert = this.databaseHelper.getWritableDatabase().insert("sms", "address", contentValues);
    if (insertListener != null) {
        insertListener.onComplete();
    }
    if (z2) {
        DatabaseComponent.get(this.context).threadDatabase().update(j, true);
    }
    if (DatabaseComponent.get(this.context).threadDatabase().getLastSeenAndHasSent(j).first().longValue() < outgoingTextMessage.getSentTimestampMillis()) {
        DatabaseComponent.get(this.context).threadDatabase().setLastSeen(j, outgoingTextMessage.getSentTimestampMillis());
    }
    DatabaseComponent.get(this.context).threadDatabase().setHasSent(j, true);
    notifyConversationListeners(j);
    return insert;
}
```

어플리케이션 분석 – DB 생성, open, close, 조작 (4)

5. signal_v4.db sms 테이블에 update(읽음 표시)

```
public void incrementReceiptCount(MessagingDatabase.SyncMessageId syncMessageId, boolean z, boolean z2) {
    Cursor cursor;
    SQLiteDatabase writableDatabase = this.databaseHelper.getWritableDatabase();
    try {
        Cursor query = writableDatabase.query("sms", new String[]{"_id", "thread_id", "address", "type"}, "date_sent = ?", new String[]{String.valueOf(syncMessageId.getTimestamp())}, null);
        boolean z3 = false;
        while (query.moveToNext()) {
            try {
                if (HmsSmsColumns.Types.isOutgoingMessageType(query.getLong(query.getColumnIndexOrThrow("type")))) {
                    Address address = syncMessageId.getAddress();
                    Address fromSerialized = Address.fromSerialized(query.getString(query.getColumnIndexOrThrow("address")));
                    String str = z ? "delivery_receipt_count" : "read_receipt_count";
                    if (fromSerialized.equals(address)) {
                        long i = query.getLong(query.getColumnIndexOrThrow("thread_id"));
                        writableDatabase.execSQL("UPDATE sms SET " + str + " = " + str + " + 1 WHERE _id = ?", new String[]{String.valueOf(query.getLong(query.getColumnIndexOrThrow("_id")))} );
                        DatabaseComponent.get(this.context).threadDatabase().update(), false);
                        notifyConversationListeners();
                        z3 = true;
                    }
                }
            } catch (Throwable th) {
                th = th;
                cursor = query;
                if (cursor != null) {
                    cursor.close();
                }
                throw th;
            }
        }
        if (!z3) {
            if (z) {
                earlyDeliveryReceiptCache.increment(syncMessageId.getTimestamp(), syncMessageId.getAddress());
            }
            if (z2) {
                earlyReadReceiptCache.increment(syncMessageId.getTimestamp(), syncMessageId.getAddress());
            }
        }
        if (query != null) {
            query.close();
        }
    } catch (Throwable th2) {
        th = th2;
        cursor = null;
    }
}
```

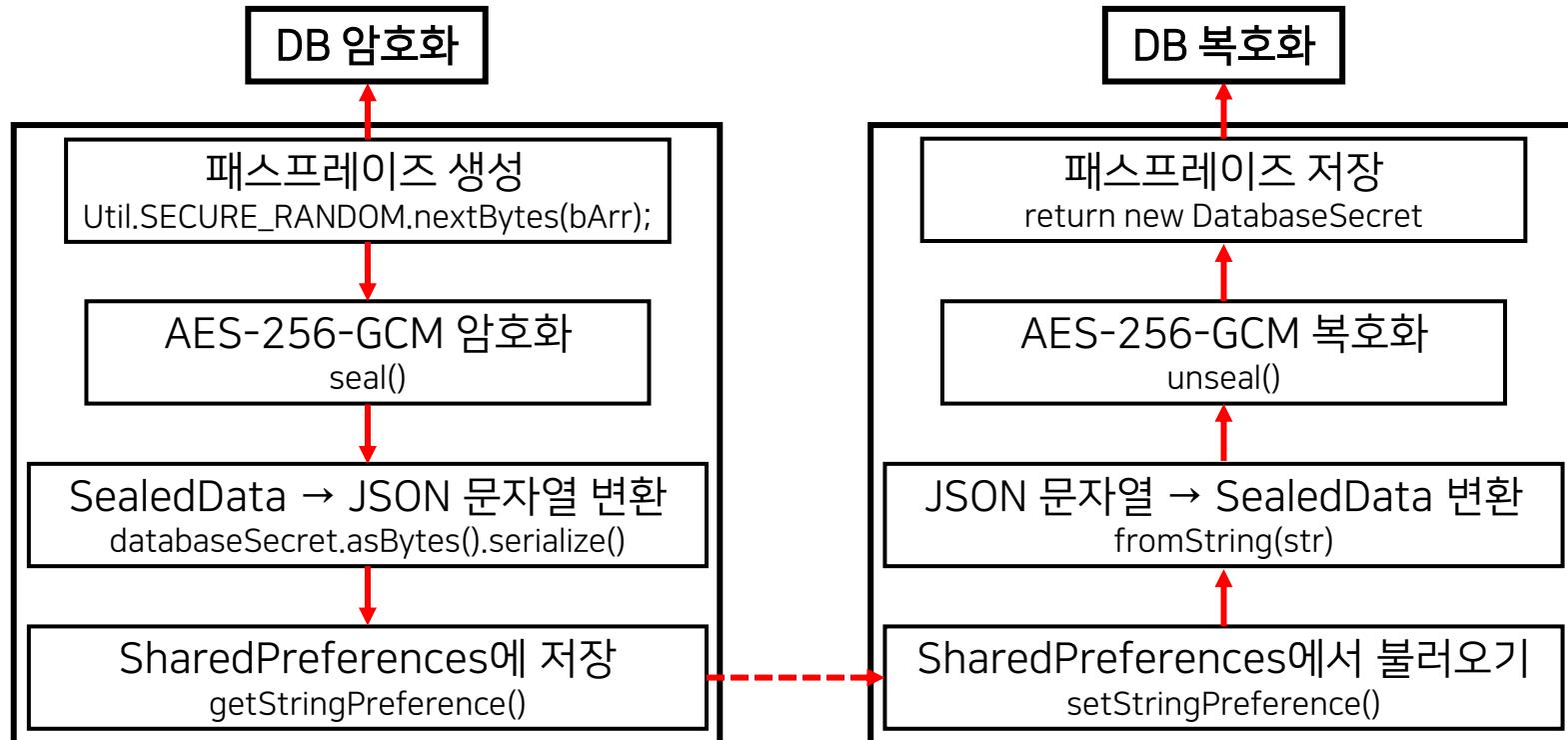
어플리케이션 분석 – DB 생성, open, close, 조작 (5)

6. signal_v4.db sms 테이블에 delete

```
public boolean deleteMessage(long j) {
    Log.i("MessageDatabase", "Deleting: " + j);
    SQLiteDatabase writableDatabase = this.databaseHelper.getWritableDatabase();
    long threadIdForMessage = getThreadIdForMessage(j);
    writableDatabase.delete("sms", "_id = ?", new String[]{j + ""});
    notifyConversationListeners(threadIdForMessage);
    return DatabaseComponent.get(this.context).threadDatabase().update(threadIdForMessage, false);
}
```

어플리케이션 분석 정리

Session 구조도



DB 복호화 (1)

패스프레이즈 획득

shared_prefs/network.loki.messenger_preferences.xml에서
pref_database_encrypted_secret에서 data, iv 획득

data: "ze7wmwGfvFx15KBUC/4gNQM582R7SlowkJM9h2e5d2qUDZdRwxklx8xyljyRLTH/"

암호화된 패스프레이즈 + GCM 태그 연결

iv: "D3DRepyI1KsXAuHO"

암호화에 사용된 iv가 Base64 인코딩되어 있음

```
<string name="pref_database_encrypted_secret">{"data":"ze7wmwGfvFx15KBUC/4gNQM582R7SlowkJM9h2e5d2qUDZdRwxklx8xyljyRLTH/","iv":"D3DRepyI1KsXAuHO"}</string>
```

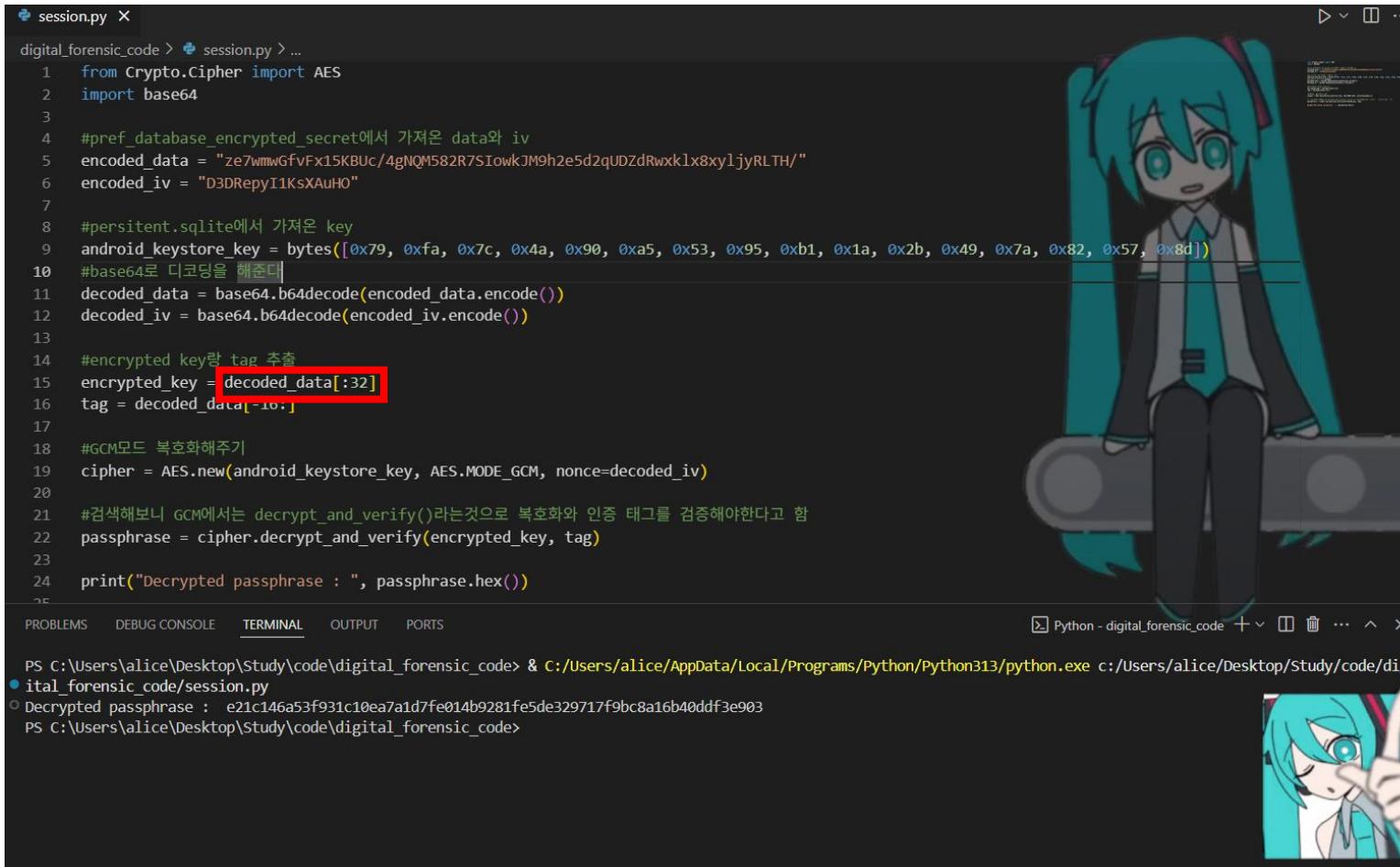
```
<string name="pref_database_encrypted_secret">
{"data":"ze7wmwGfvFx15KBUC/4gNQM582R7SlowkJM9h2e5d2qUDZdRwxklx8xyljyRLTH/","iv":"D3DRepyI1KsXAuHO"}
</string>
```

암호화된 패스프레이즈는 network.loki.messenger_preferences 내 pref_database_encrypted_secret 요소에 JSON 형태로 저장되어 있다(그림 6).

DB 복호화 (2)

패스프레이즈 복호화

Python 이용 - 결과: 'e21c146a53f931c10ea7a1d7fe014b9281fe5de329717f9bc8a16b40ddf3e903'



```
session.py X
digital_forensic_code > session.py > ...
1  from Crypto.Cipher import AES
2  import base64
3
4  #pref_database_encrypted_secret에서 가져온 data와 iv
5  encoded_data = "ze7wmwGfvFx15KBUC/4gNQM582R7SIowkJM9h2e5d2qUDZdRwxk1x8xyljyRLTH/"
6  encoded_iv = "D3DRepyI1KsXAUHO"
7
8  #persistent.sqlite에서 가져온 key
9  android_keystore_key = bytes([0x79, 0xfa, 0x7c, 0x4a, 0x90, 0xa5, 0x53, 0x95, 0xb1, 0x1a, 0x2b, 0x49, 0x7a, 0x82, 0x57, 0x8d])
10 #base64로 디코딩해준다
11 decoded_data = base64.b64decode(encoded_data.encode())
12 decoded_iv = base64.b64decode(encoded_iv.encode())
13
14 #encrypted key랑 tag 추출
15 encrypted_key = decoded_data[32:]
16 tag = decoded_data[-16:]
17
18 #GCM모드 복호화해주기
19 cipher = AES.new(android_keystore_key, AES.MODE_GCM, nonce=decoded_iv)
20
21 #검색해보니 GCM에서는 decrypt_and_verify()라는것으로 복호화와 인증 태그를 검증해야한다고 함
22 passphrase = cipher.decrypt_and_verify(encrypted_key, tag)
23
24 print("Decrypted passphrase : ", passphrase.hex())
25
```

PROBLEMS DEBUG CONSOLE TERMINAL OUTPUT PORTS

PS C:\Users\alice\Desktop\Study\code\digital_forensic_code> & C:/Users/alice/AppData/Local/Programs/Python/Python313/python.exe c:/Users/alice/Desktop/Study/code/digital_forensic_code/session.py

● Decrypted passphrase : e21c146a53f931c10ea7a1d7fe014b9281fe5de329717f9bc8a16b40ddf3e903

PS C:\Users\alice\Desktop\Study\code\digital_forensic_code>

Input : pref_database_encrypted_secret, Android_KeyStore_Key
OutPut : Passphrase
1 : IV ← Base64_Decode(pref_database_encrypted_secret.iv)
2 : Data ← Base64_Decode(pref_database_encrypted_secret.data)
3 : Tag ← Data[-16:]
4 : Encrypted_key ← Data[0: 48] Data[0: 48]
5 : Passphrase ← AES256/GCM/NoPadding_Decrypt
(Android_Keystore_Key, IV, Encrypted_key, Tag)
6 : return Passphrase

(그림 5) Getting PassPhrase of signal_v4.db

Data[0:48] → 복호화된 패스프레이즈 길이 = 48

≠

AES256/GCM/NoPadding 키 길이 = 32

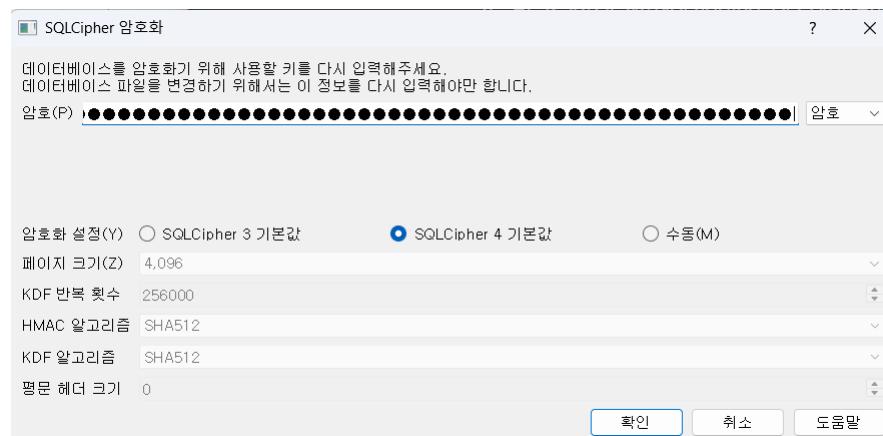
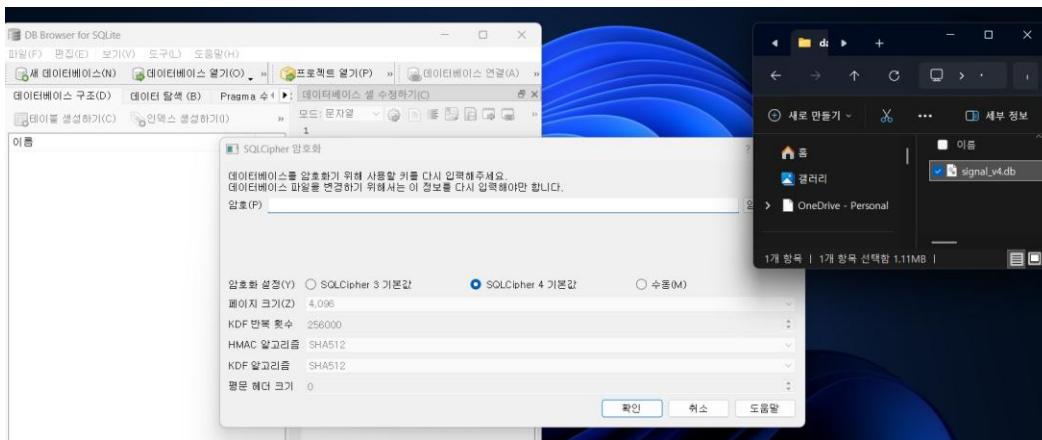
→ Data[0:32]로 진행

```
byte[] bArr = new byte[32];
Util.SECURE_RANDOM.nextBytes(bArr);
DatabaseSecret databaseSecret = new DatabaseSecret(bArr);
```

DB 복호화 (3)

DB 복호화

패스프레이즈 입력을 통해 DB 복호화 확인

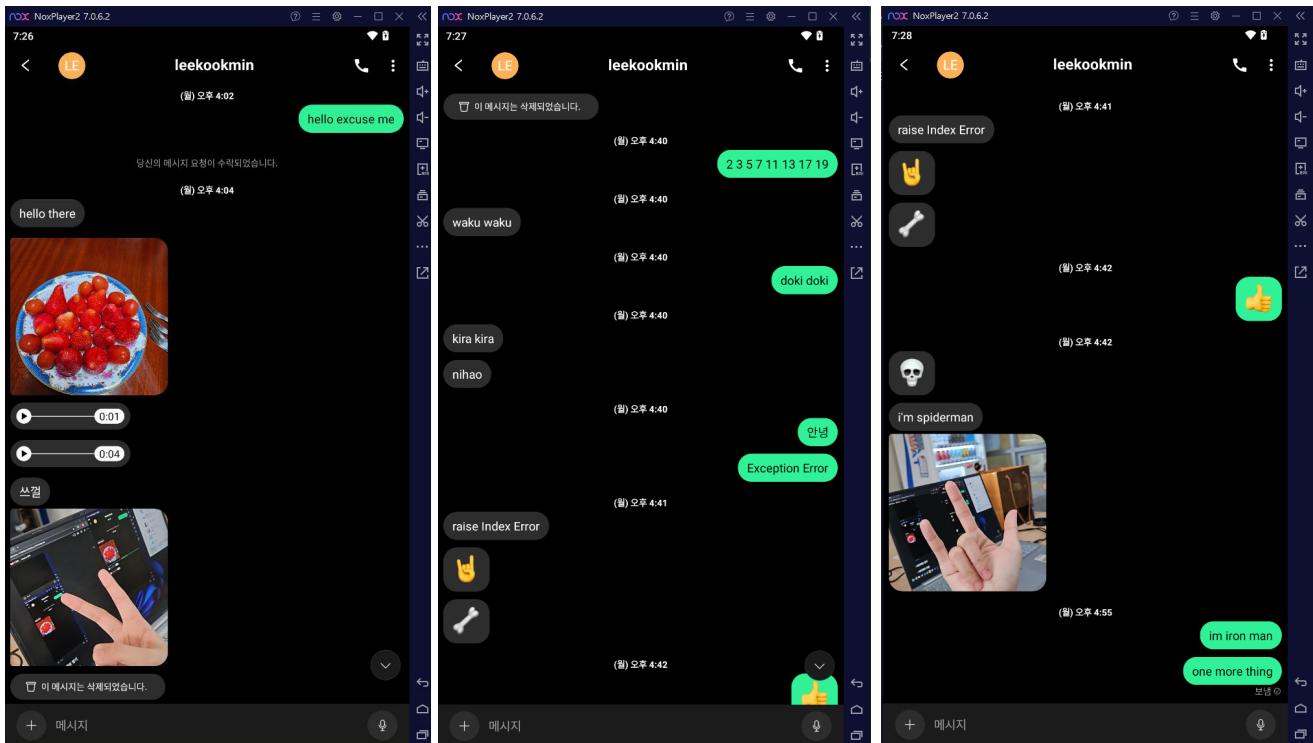


The screenshot shows the main interface of DB Browser for SQLite for an SQLite database named 'signal_v4.db'. The table list pane shows 66 tables, each with its corresponding CREATE TABLE statement. The schema pane on the right shows the current schema of the database. A message at the bottom left indicates that no schema changes have been made since the last update.

이름	타입	스키마
backup_files	CREATE TABLE	ba
blinded_id_mapping	CREATE TABLE	bl
closed_group_encryption_key_pairs...	CREATE TABLE	cl
closed_group_public_keys_table	CREATE TABLE	cl
configs_table	CREATE TABLE	cc
drafts	CREATE TABLE	dr
emoji_search	CREATE VIRTUAL	'e
emoji_search_config	CREATE TABLE	'e
emoji_search_content	CREATE TABLE	'e
emoji_search_data	CREATE TABLE	'e
emoji_search_docsizes	CREATE TABLE	'e
emoji_search_idx	CREATE TABLE	'e
expiration_configuration	CREATE TABLE	ex
fork_info	CREATE TABLE	fo
group_member	CREATE TABLE	gr
group_receipts	CREATE TABLE	gr
groups	CREATE TABLE	gr
last_legacy_messages	CREATE TABLE	la
loki_api_group_chat_auth_token_dat...	CREATE TABLE	lo
loki_api_last_deletion_server_id_cache	CREATE TABLE	lo
loki_api_last_message_server_id_cac...	CREATE TABLE	lo
loki_api_swarm_cache	CREATE TABLE	lo
loki_error_message_database	CREATE TABLE	lo
loki_group_invites	CREATE TABLE	lo
loki_message_friend_request_database...	CREATE TABLE	lo
loki_message_hash_database	CREATE TABLE	lo

DB 복호화 (4)

복호화된 데이터베이스에서 대화 확인
주고받은 텍스트를 확인할 수 있었음



테이블(T): sms												
sent	protocol	read	status	type	reply_path_present	delivery_receipt_count	subject	body	mismatched_identities	service_center	subscription	filter
1 3714414	NULL	1	-1	10485783	NULL	0	NULL	hello_world	NULL	NULL	NULL	필터
2 1522201	NULL	1	-1	10485783	NULL	0	NULL	hello excuse me	NULL	NULL	NULL	필터
3 1697984	31337	1	-1	10485780	1	0	NULL	hello there	NULL	GCM	필터	
4 3810215	NULL	1	-1	10485783	NULL	0	NULL	2 3 5 7 11 13 17 19	NULL	NULL	NULL	필터
5 3818811	31337	1	-1	10485780	1	0	NULL	waku waku	NULL	GCM	필터	
6 3826267	NULL	1	-1	10485783	NULL	0	NULL	doki doki	NULL	NULL	NULL	필터
7 3827020	31337	1	-1	10485780	1	0	NULL	kira kira	NULL	GCM	필터	
8 3832320	31337	1	-1	10485780	1	0	NULL	nihao	NULL	GCM	필터	
9 3852568	NULL	1	-1	10485783	NULL	0	NULL	안녕	NULL	NULL	NULL	필터
10 3873138	NULL	1	-1	10485783	NULL	0	NULL	Exception Error	NULL	NULL	NULL	필터
11 3878669	31337	1	-1	10485780	1	0	NULL	raise Index Error	NULL	GCM	필터	
12 3906442	31337	1	-1	10485780	1	0	NULL	🤘	NULL	GCM	필터	
13 3921921	31337	1	-1	10485780	1	0	NULL	👉	NULL	GCM	필터	
14 3930209	NULL	1	-1	10485783	NULL	0	NULL	🔥	NULL	NULL	NULL	필터
15 3930513	31337	1	-1	10485780	1	0	NULL	💀	NULL	GCM	필터	
16 3716257	NULL	1	-1	10485783	NULL	0	NULL	i'm iron man	NULL	NULL	NULL	필터
17 3951479	NULL	1	-1	10485783	NULL	0	NULL	one more thing	NULL	NULL	NULL	필터

테이블(T): mms							
date	date_received	msg_box	read	m_id	sub	sub_cs	body
1 1743404684515	1743404684515	10551316	1	NULL	NULL	NULL	NULL
2 1743404728201	1743404742385	10485780	1	NULL	NULL	NULL	NULL
3 1743406194801	1743406200522	10485780	1	NULL	NULL	NULL	NULL
4 1743406202226	1743406211511	10485780	1	NULL	NULL	NULL	NULL
5 1743406345332	1743406355472	10485780	1	NULL	NULL	NULL	쓰篾
6 1743406699338	1743406709698	10485779	1	NULL	NULL	NULL	이 메시지는 삭제되었습니다.
7 1743407023074	1743407036407	10485780	1	NULL	NULL	NULL	i'm spiderman

DB 내 sms 테이블 아티팩트 분석

Columns	Contents	body	message
_id	id	mismatched_identities	NULL / 1(본인 키 불일치)
thread_id	1 st person / 2 nd person ...	service_center	NULL(sent) / GCM(google cloud messaging)
address	person's address	subscription_id	-1 (default)
address_device_id	1 (default, used in multi device)	expires_in	0 / expiring time
person	0 (default, used in multi chat)	expired_started	0 / expired time
date	message save time	notified	0 (default)
date_sent	message sent time	Read_receipt_count	'Read Receipt' count
protocol	NULL(sent) / 31337(received)	unidentified	0 / 1(상대 공개키 불일치)
read	0 (unread) / 1 (read)	is_deleted	0 (default) / 1 (deleted)
status	-1 (default)	reactions_unread	0 (unread) / 1 (read)
type	Received / Sent / Push / Encrypted	has_mention	0(default) / 1 (mentioned)
reply_path_present	NULL(sent) / 1(received)	is_group_update	0 (default) / 1(updated)
delivery_receipt_count	'Delivery Receipt' count		
subject	NULL (used in MMS as title)		

Q&A

감사합니다

출처: 보안 인스턴트 메신저 Session에 대한 데이터베이스 복호화 방안 연구
<https://developer.android.com/privacy-and-security/keystore?hl=ko>