세션 데이터베이스 복호화 및 주요 아티팩트 분류

2조
20232106 정유진 / 20212052 이동훈
alice2002@kookmin.ac.kr / dhsama51@kookmin.ac.kr
25.04.01

목차

Android Keystore의 개념

Android 기기에서 **암호키를 안전하게 관리하는 시스템.**

TEE (Trusted Execution Environment), StrongBox 등에 의해 키가 보안 HW 외부에 노출되지 않음

→ 기기 외부로 복사하거나 내보낼 수 없음

앱은 키 생성, 저장, 암/복호화, 서명/검증을 위임함

그러나 Nox 등 에뮬레이터는 루팅 + HW 보안 모듈이 없음

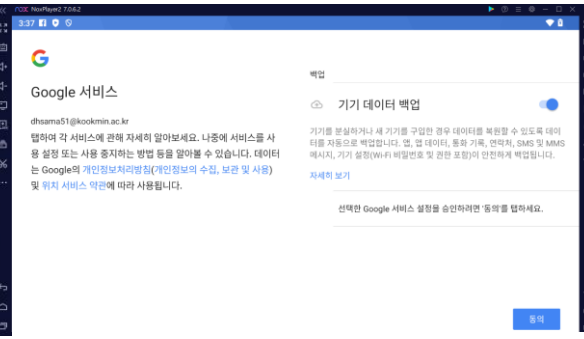
→ 평문 또는 약한 보호 상태로 저장되어서 추출 가능

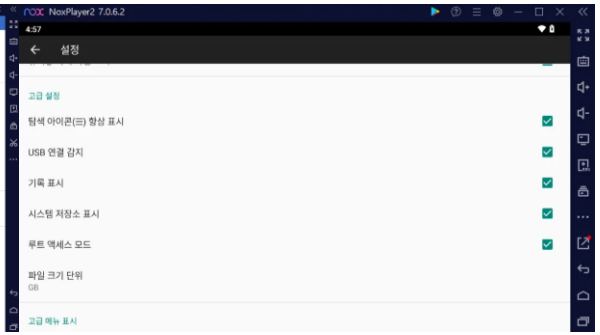
데이터베이스 복호화 – 환경 세팅

Nox, jadx, Python 사용

Nox – Root, 백업 설정

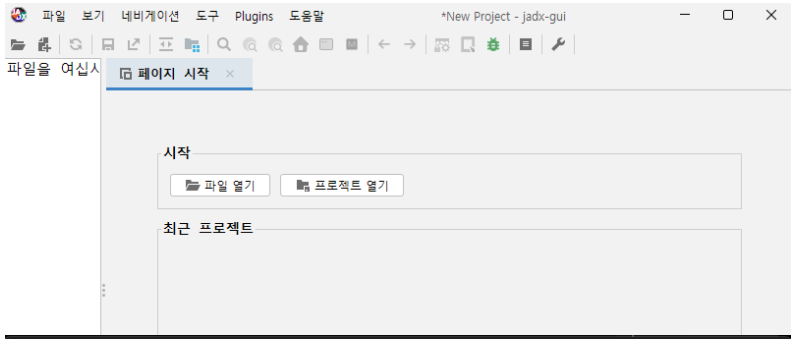
파일 탐색기 설정

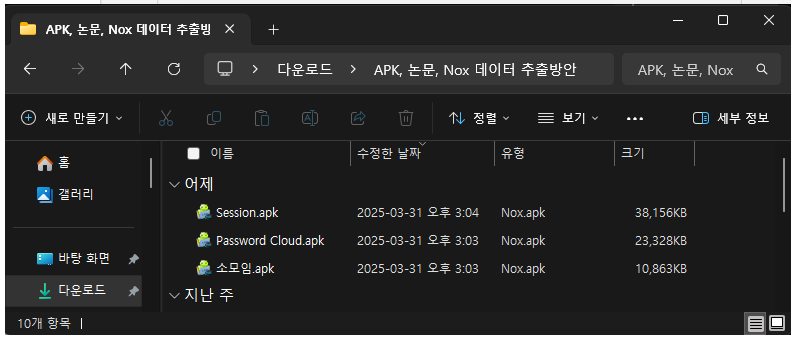


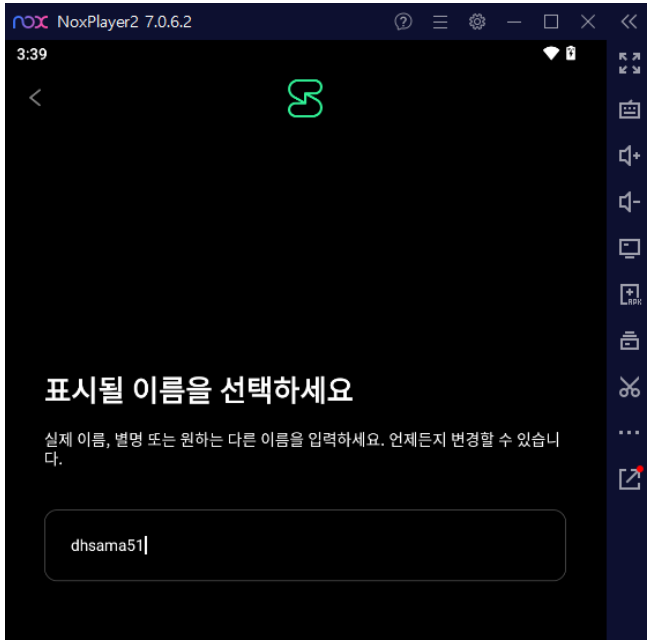


jadx – Session.apk 준비

세션 회원가입

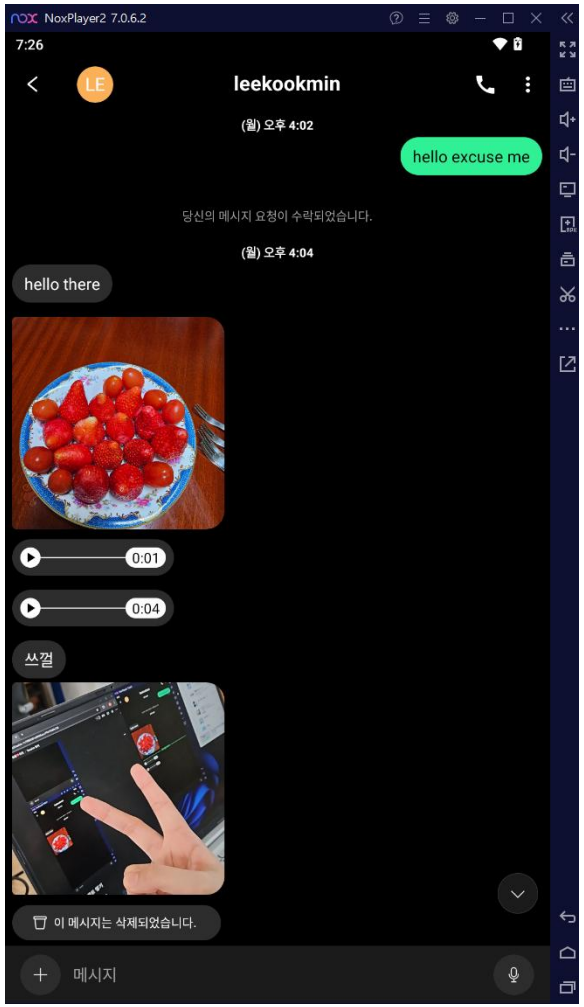


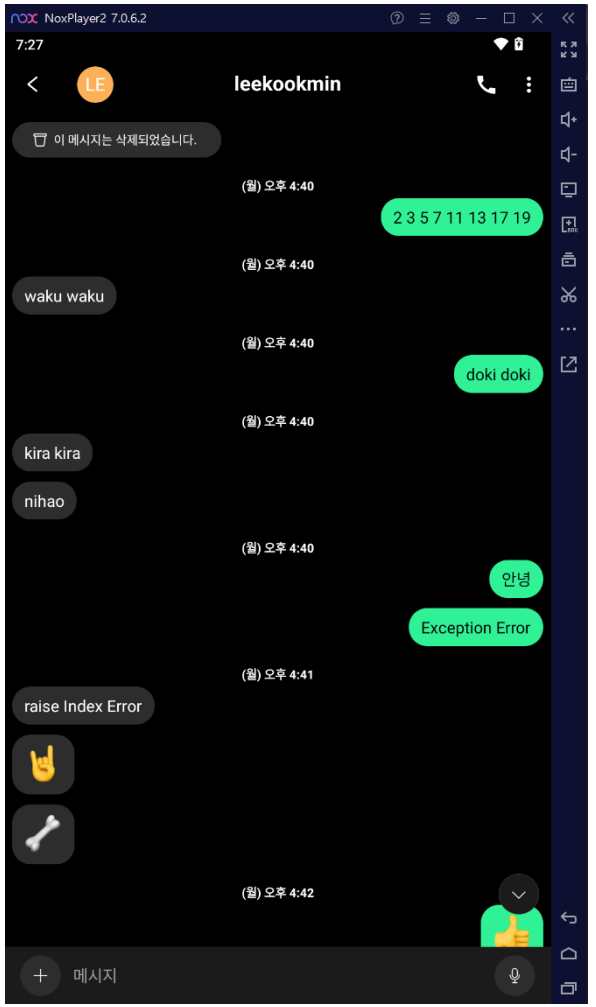


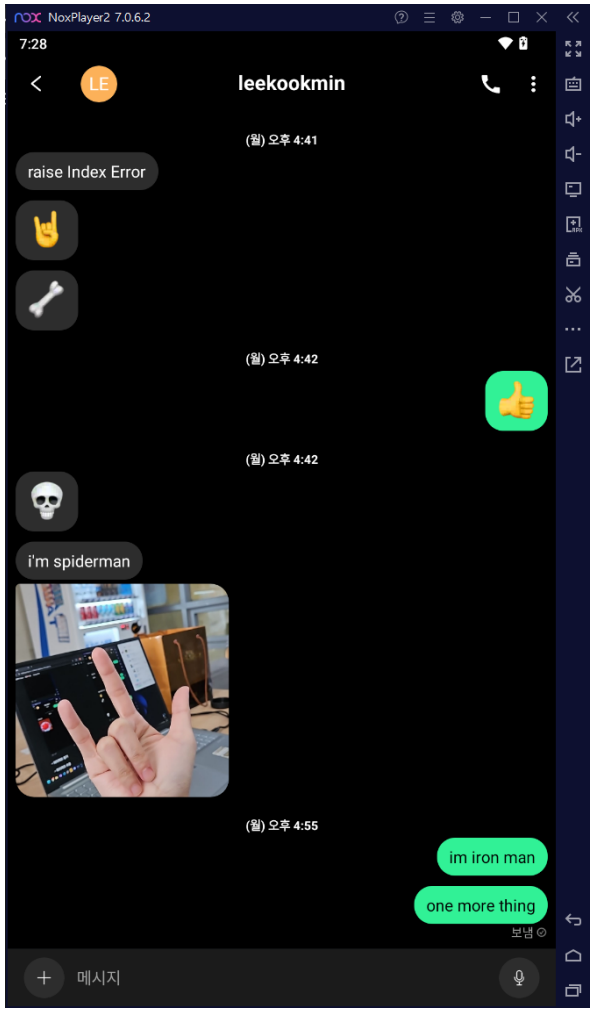


데이터베이스 복호화 – 아티팩트 쌓기

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

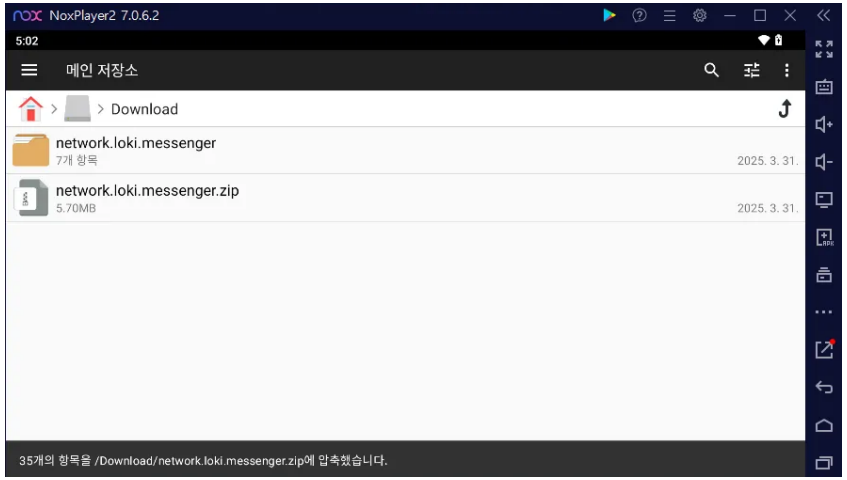


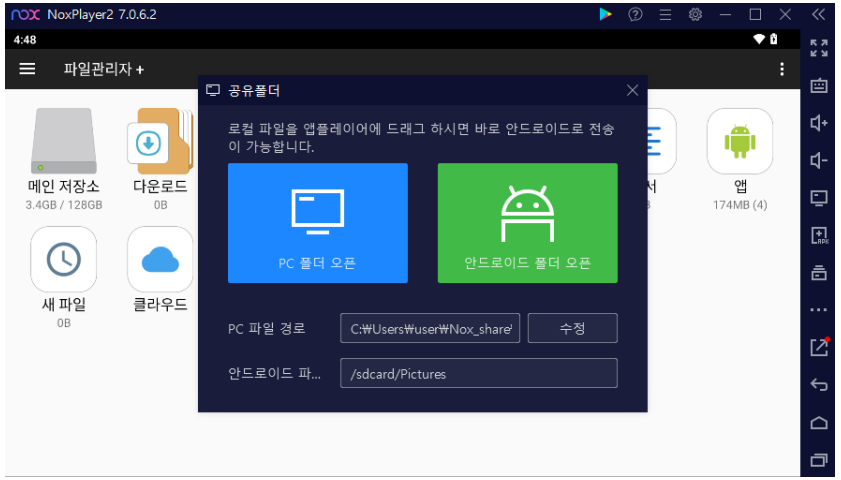


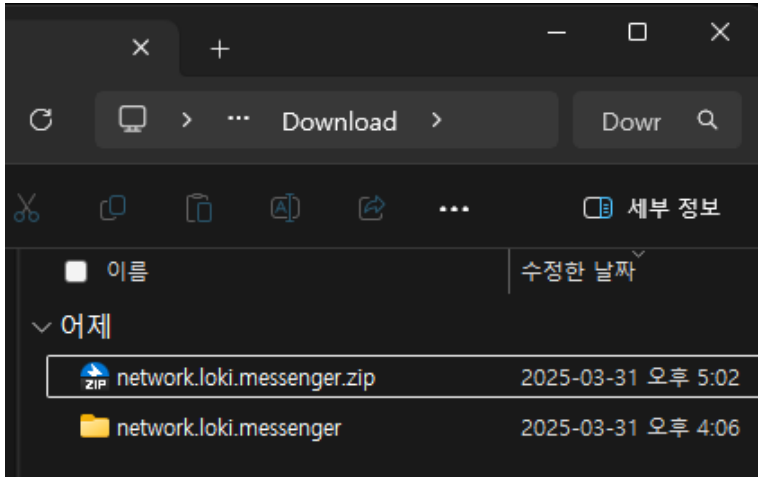


데이터베이스 복호화 – 아티팩트 추출

network.loki.messenger 폴더 Download로 복사, Windows로 이동





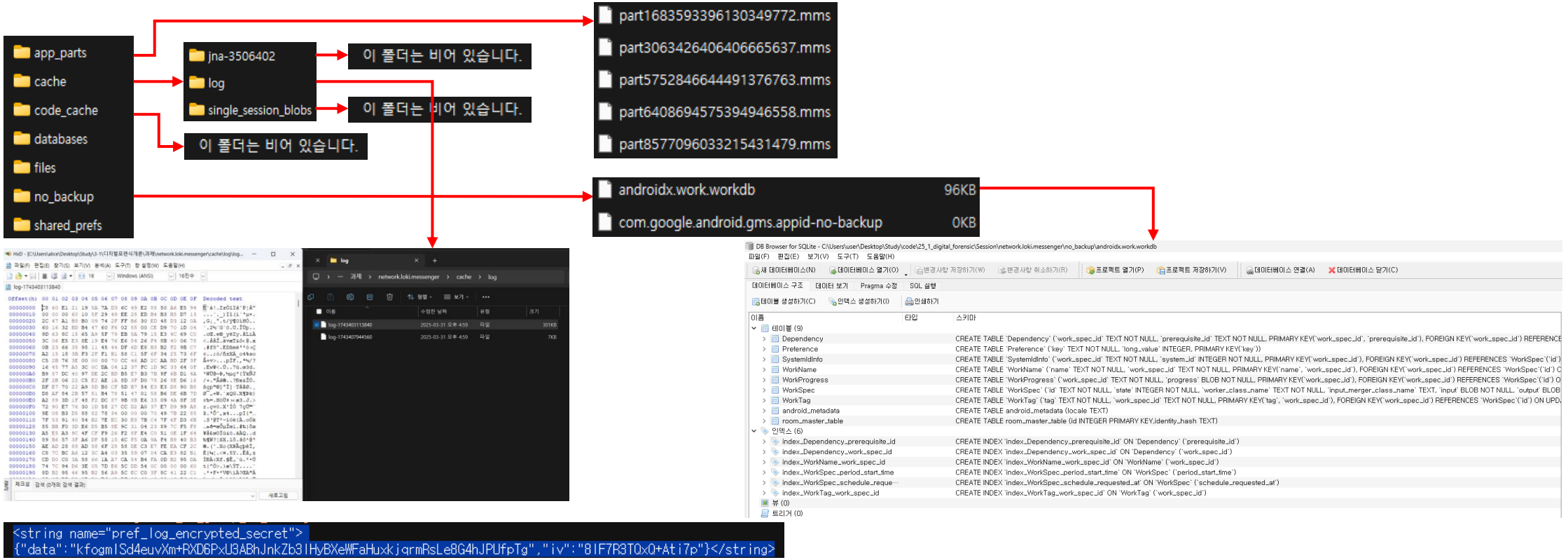


데이터베이스 복호화 – 아티팩트 분석 (1)

app_parts: mms 파일인 것으로 보아 멀티미디어 파일(사진, 음성 파일)일 가능성 있음

cache/log: 특이사항을 찾을 수 없고, log 파일 암호화 가능성 있음

no_backup/android.wor.workdb: 앱 설정과 관련된 내용으로 보임

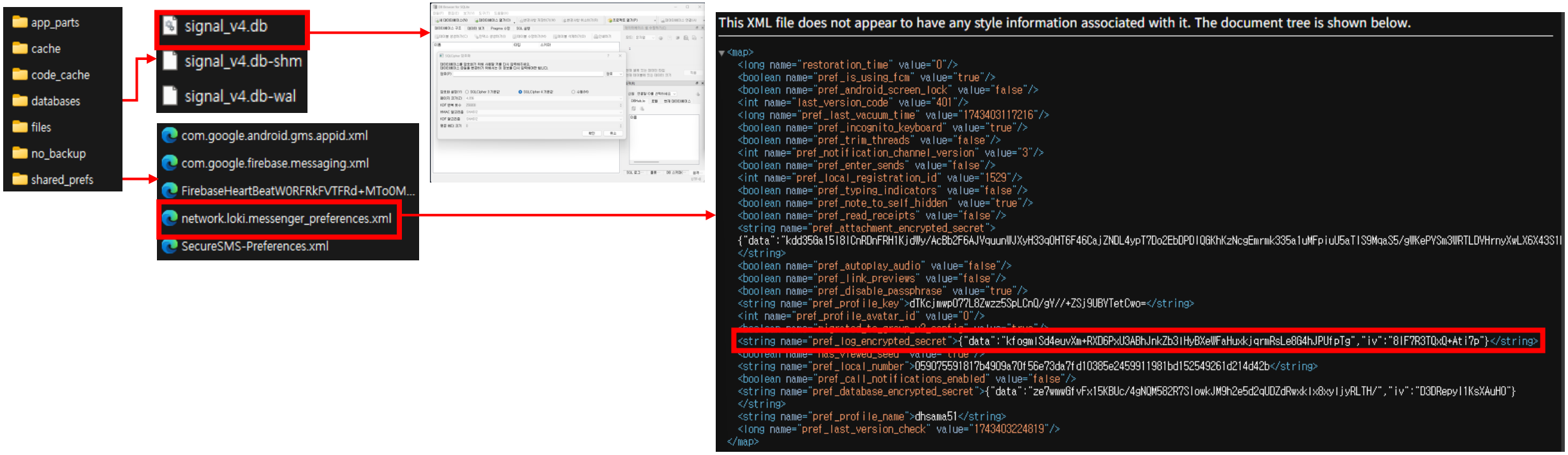


데이터베이스 복호화 – 아티팩트 분석 (2)

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

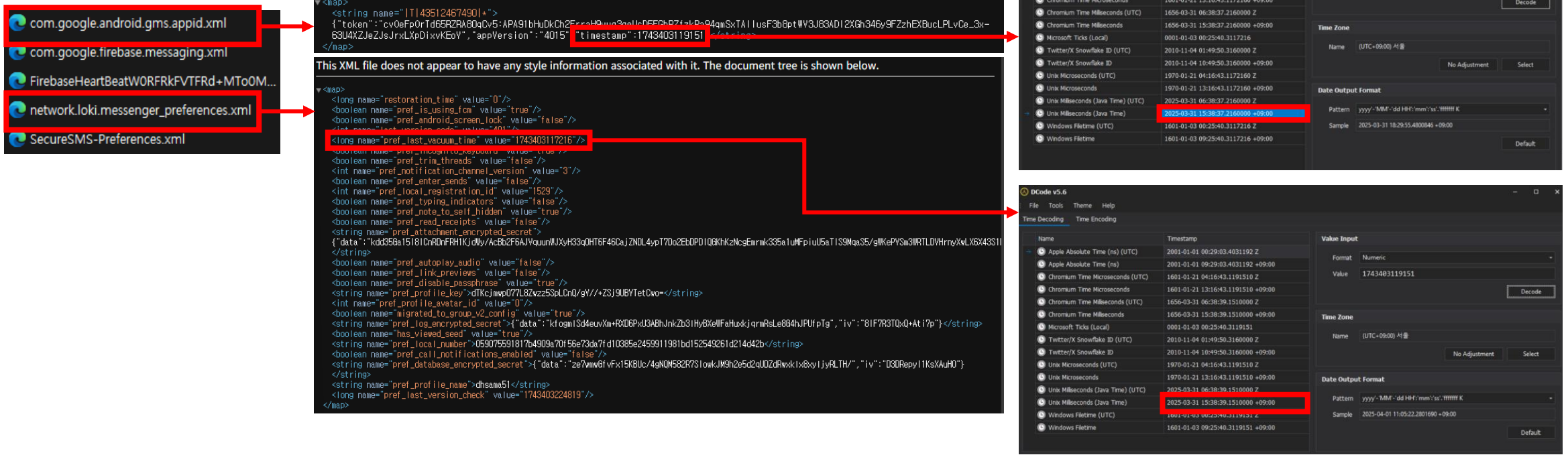
pref_last_vacuum_time, pref_attachment_encrypted_secret, pref_profile_key, pref_log_encrypted_secret, pref_database_encrypted_secret, pref_profile_name

databases/signal_v4.db: 데이터베이스가 암호화되어 있음



데이터베이스 복호화 – 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) 시간으로 추정



데이터베이스 복호화 – 어플리케이션 분석 (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)

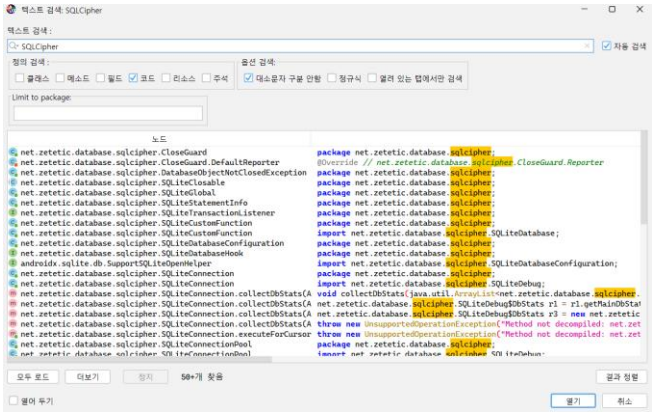
데이터베이스 복호화 – 어플리케이션 분석 (2)

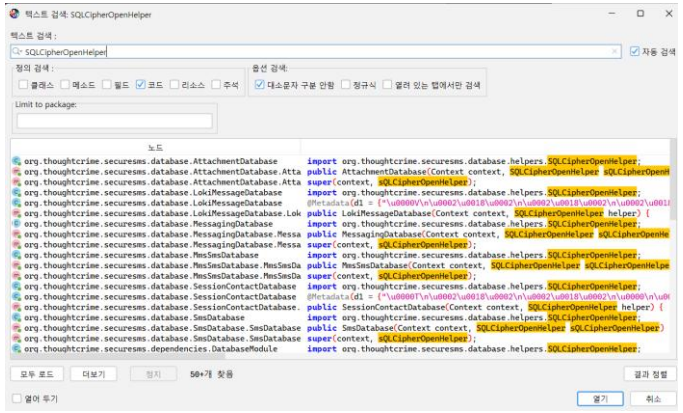
데이터베이스 암호화 정보 확인

SQLCipher로 암호화되어 있음 → 관련 코드 검색

암호화된 데이터베이스 이름인 signal_v4.db 확인

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






```
/* loaded from: classes4.dex */
107 public class SQLCipherOpenHelper extends SQLiteOpenHelper {
    private static final String CIPHER3_DATABASE_NAME = "signal.db";
    public static final String DATABASE_NAME = "signal_v4.db";
    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;
```

코드 Smali Simple Fallback Split view

SQLCipher로 암호화되었다.

데이터베이스 복호화 – 어플리케이션 분석 (3)

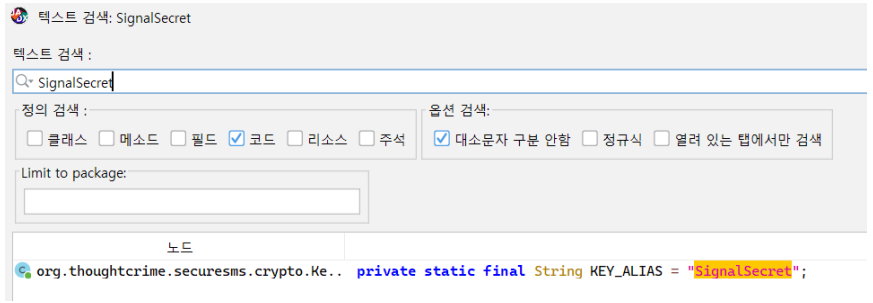
Session 암호키 정보

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).



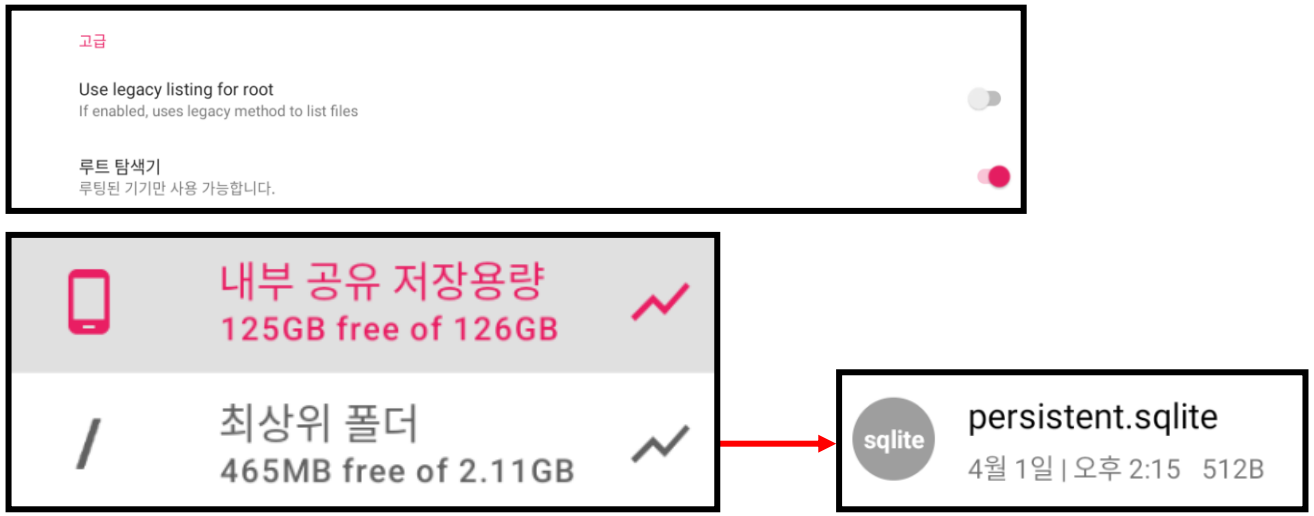

```
/* loaded from: classes4.dex */
47 public final class KeyStoreHelper {
       private static final String ANDROID_KEY_STORE = "AndroidKeyStore";
       private static final String KEY_ALIAS = "SignalSecret";

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

데이터베이스 복호화 – 어플리케이션 분석 (4)

Android Keystore에서 암호키 추출 (1)

Amaze 파일 매니저 이용해 persist.sqlite 추출

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



데이터베이스 복호화 – 어플리케이션 분석 (5)

Android Keystore에서 암호키 추출 (2)

keyentry에서 SignalSecret 발견,

blobentry에서 같은 id를 가진 것 발견

0xD~0x1C에서 암호키 추출

이름 | 타입

- 테이블 (7)
  - blobentry
  - blobmetadata
  - grant
  - keyentry
  - keymetadata
  - keyparameter
  - version
- 인덱스 (6)
  - blobentry_keyentryid_index
  - blobmetadata_blobentryid_index
  - keyentry_domain_namespace_index
  - keyentry_id_index
  - keymetadata_keyentryid_index
  - keyparameter_keyentryid_index
- 뷰 (0)
- 트리거 (0)

테이블(T): keyentry

	id	key_type	domain	namespace	ali
1	6926690968309298972	0	0	1017	boot_level_key
2	-58582767595457548 4	0	0	10046	c5635988b7157a3a
3	-1154276977243341612	0	0	10046	mobstore_encrypt
4	6044086939184787821	0	0	10046	nearby_presence_connection_secr
5	-6262722488817596635	0	0	10046	nearby_presence_connection_secr
6	-1994549063571697783	0	0	10046	nearby_presence_connection_secr
7	-3169231379557241686	0	0	10046	nearby_presence_connection_secr
8	3392414123250361534	0	0	10046	nearby_presence_connection_secr
9	-7823036944918506034	0	0	10046	nearby_presence_connection_secr
10	-5423159458800941720	0	0	10046	nearby_presence_connection_secr
11	6940908571265112464	0	0	10046	nearby_presence_connection_secr
12	-9167606571973839702	0	0	10046	nearby_presence_connection_secr
13	-4036893587965478311	0	0	10062	SignalSecret

테이블(T): blobentry

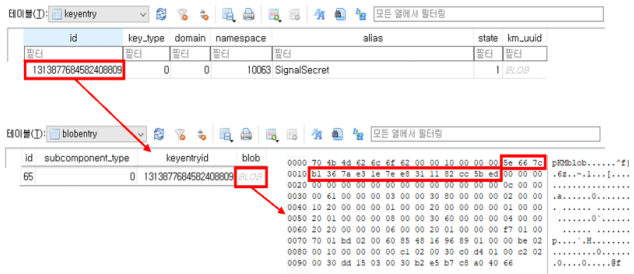
	id	subcomponent_type	keyentryid	blob
4	4	0	-1154276977243341612	BLOB
5	7	0	6044086939184787821	BLOB
6	10	0	-6262722488817596635	BLOB
7	13	0	-1994549063571697783	BLOB
8	16	0	-3169231379557241686	BLOB
9	19	0	3392414123250361534	BLOB
10	22	0	-7823036944918506034	BLOB
11	25	0	-5423159458800941720	BLOB
12	28	0	6940908571265112464	BLOB
13	31	0	-9167606571973839702	BLOB
14	32	0	-4036893587965478311	BLOB

데이터베이스 셀 수정하기(C)

모드: 바이너리

```
0000 70 4b 4d 62 6c 6f 62 00 00 10 00 00 00 79 fa 7c  pKMblob.....y.|
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 00 03 00 00 30 80 00 00 00 02 00 00  .a......0.......
0040 10 20 00 00 00 01 00 00 20 00 00 00 00 01 00 00  . ...... .......
0050 20 01 00 00 00 08 00 00 30 60 00 00 00 04 00 00   .......0`......
0060 20 20 00 00 00 06 00 00 20 01 00 00 00 f7 01 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 02  .........0......
0090 00 30 dd 15 03 00 90 9b 4c f8 38 9a 10 68        .0......L.8..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 저장되어있다(그림 5).



(그림 3) Android Keystore Key

데이터베이스 복호화 – 어플리케이션 분석 (6)

패스프레이즈 획득

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

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

data: "ze7wmwGfvFx15KBUc/4gNQM582R7SIowkJM9h2e5d2qUDZdRwxklx8xyljyRLTH/"

암호화된 패스프레이즈 + GCM 태그 연접

iv: "D3DRepyI1KsXAuHO"

암호화에 사용된 iv가 Base64 인코딩되어 있음

<string name="pref_database_encrypted_secret">
{"data":"ze7wmwGfvFx15KBUc/4gNQM582R7SIowkJM9h2e5d2qUDZdRwxklx8xyljyRLTH/","iv":"D3DRepyI1KsXAuHO"}
</string>

데이터베이스 복호화 – 어플리케이션 분석 (7)

패스프레이즈 복호화

Python 이용 – 결과: 'e21c146a53f931c10ea7a1d7fe014b9281fe5de329717f9bc8a16b40ddf3e903'

session.py

digital_forensic_code > session.py > ...

```python
from Crypto.Cipher import AES
import base64

#pref_database_encrypted_secret에서 가져온 data와 iv
encoded_data = "ze7wmwGfvFx15KBUc/4gNQM582R7SIowkJM9h2e5d2qUDZdRwxklx8xyljyRLTH/"
encoded_iv = "D3DRepyI1KsXAuHO"

#persitent.sqlite에서 가져온 key
android_keystore_key = bytes([0x79, 0xfa, 0x7c, 0x4a, 0x90, 0xa5, 0x53, 0x95, 0xb1, 0x1a, 0x2b, 0x49, 0x7a, 0x82, 0x57, 0x8d])
#base64로 디코딩을 해준다
decoded_data = base64.b64decode(encoded_data.encode())
decoded_iv = base64.b64decode(encoded_iv.encode())

#encrypted key랑 tag 추출
encrypted_key = decoded_data[:32]
tag = decoded_data[-16:]

#GCM모드 복호화해주기
cipher = AES.new(android_keystore_key, AES.MODE_GCM, nonce=decoded_iv)

#검색해보니 GCM에서는 decrypt_and_verify()라는것으로 복호화와 인증 태그를 검증해야한다고 함
passphrase = cipher.decrypt_and_verify(encrypted_key, tag)

print("Decrypted passphrase : ", passphrase.hex())
```

PROBLEMS DEBUG CONSOLE TERMINAL OUTPUT PORTS

Python - digital_forensic_code

```
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]
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

≠

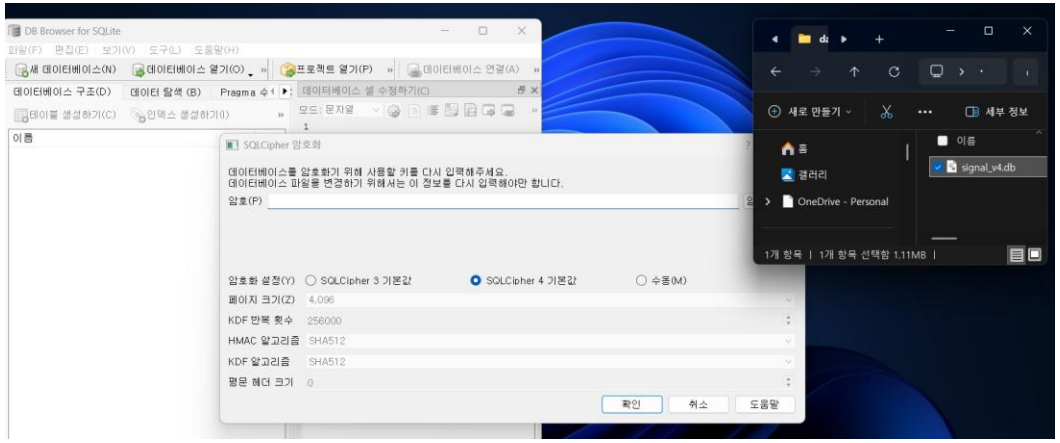
AES256/GCM/NoPadding 키 길이 =32

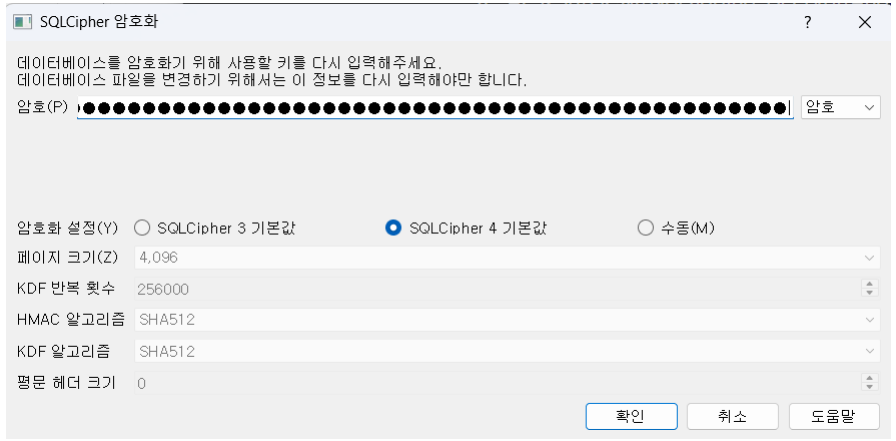
→ Data[0:32]로 진행

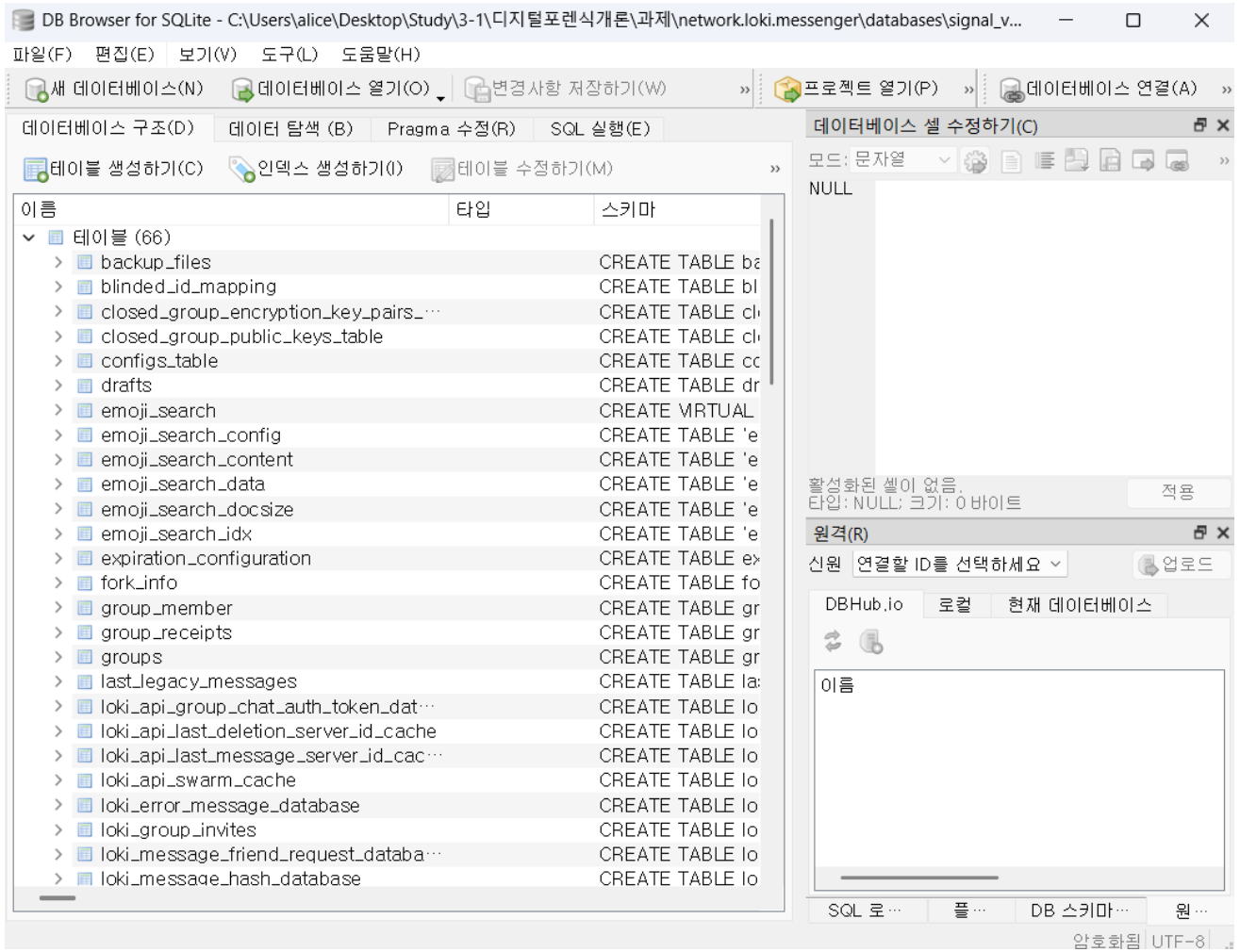
데이터베이스 복호화 – 어플리케이션 분석 (8)

데이터베이스 복호화

패스프레이즈 입력을 통해 DB 복호화 확인



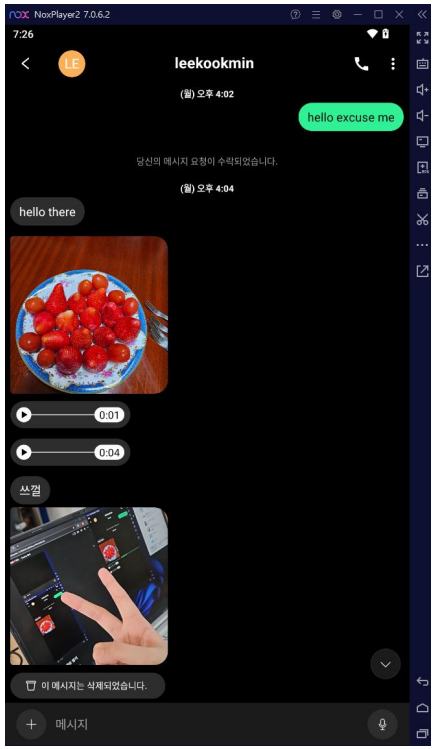


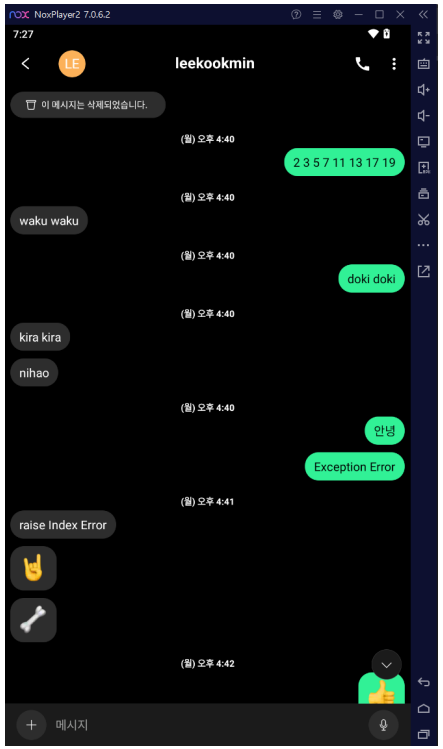


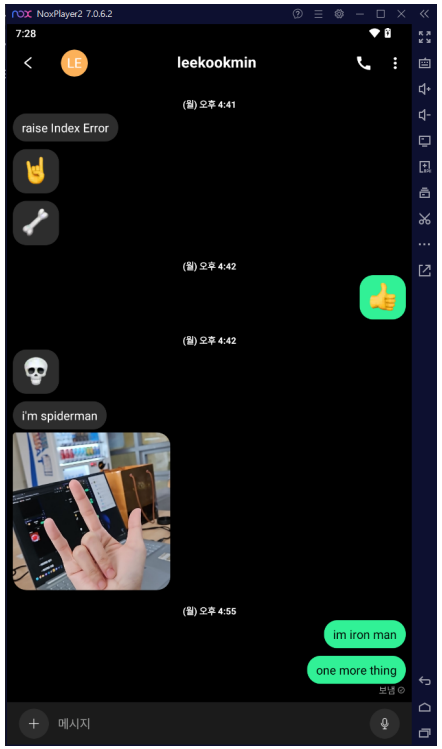
데이터베이스 복호화 – 어플리케이션 분석 (9)

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	subscript
	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터	필터
1	3714414	NULL	1	-1	10485783	NULL	0	NULL	hello_world	NULL	NULL	
2	4522201	NULL	1	-1	10485783	NULL	0	NULL	hello excuse me	NULL	NULL	
3	4697984	31337	1	-1	10485780	1	0	NULL	hello there	NULL	GCM	
4	5810215	NULL	1	-1	10485783	NULL	0	NULL	2 3 5 7 11 13 17 19	NULL	NULL	
5	5818811	31337	1	-1	10485780	1	0	NULL	waku waku	NULL	GCM	
6	5826267	NULL	1	-1	10485783	NULL	0	NULL	doki doki	NULL	NULL	
7	5827020	31337	1	-1	10485780	1	0	NULL	kira kira	NULL	GCM	
8	5832320	31337	1	-1	10485780	1	0	NULL	nihao	NULL	GCM	
9	5852568	NULL	1	-1	10485783	NULL	0	NULL	안녕	NULL	NULL	
10	5873138	NULL	1	-1	10485783	NULL	0	NULL	Exception Error	NULL	NULL	
11	5878669	31337	1	-1	10485780	1	0	NULL	raise Index Error	NULL	GCM	
12	5906442	31337	1	-1	10485780	1	0	NULL	🤘	NULL	GCM	
13	5921921	31337	1	-1	10485780	1	0	NULL	🦴	NULL	GCM	
14	5930209	NULL	1	-1	10485783	NULL	0	NULL	👍	NULL	NULL	
15	5930513	31337	1	-1	10485780	1	0	NULL	💀	NULL	GCM	
16	7716257	NULL	1	-1	10485783	NULL	0	NULL	im iron man	NULL	NULL	
17	7951479	NULL	1	-1	10485783	NULL	0	NULL	one more thing	NULL	NULL	

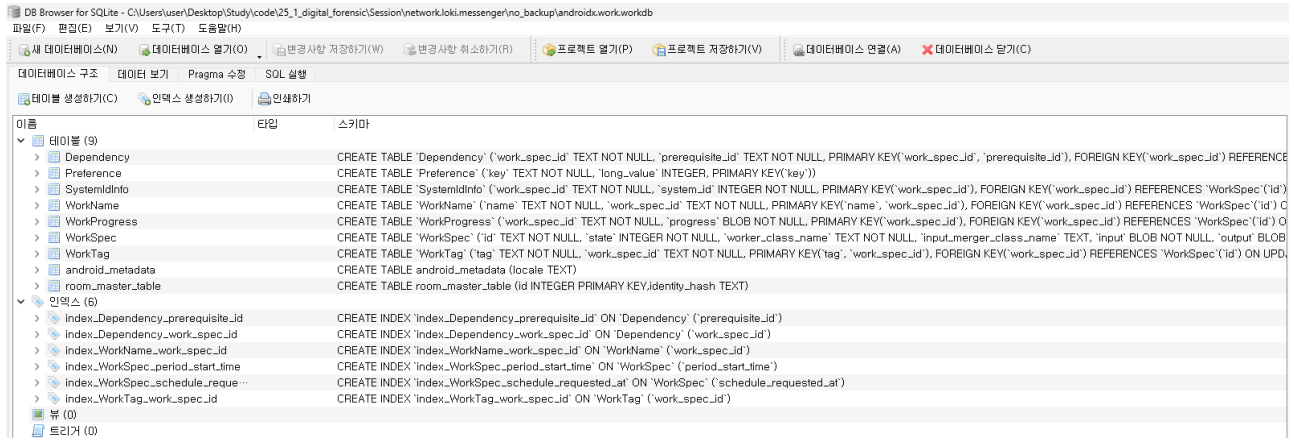
테이블(T): mms

	date	date_received	msg_box	read	m_id	sub	sub_cs	body
	필터	필터	필터	필터	필터	필터	필터	필터
1	1743404684515	1743404684515	10551316	1	NULL	NULL	NULL	NULL
2	1743404728201	1743404742385	10485780	1	NULL	NULL	NULL	
3	1743406194801	1743406200522	10485780	1	NULL	NULL	NULL	
4	1743406202226	1743406211511	10485780	1	NULL	NULL	NULL	
5	1743406345332	1743406355472	10485780	1	NULL	NULL	NULL	쓰껄
6	1743406699338	1743406709698	10485779	1	NULL	NULL	NULL	이 메시지는 삭제되었습니다.
7	1743407023074	1743407036407	10485780	1	NULL	NULL	NULL	i'm spiderman

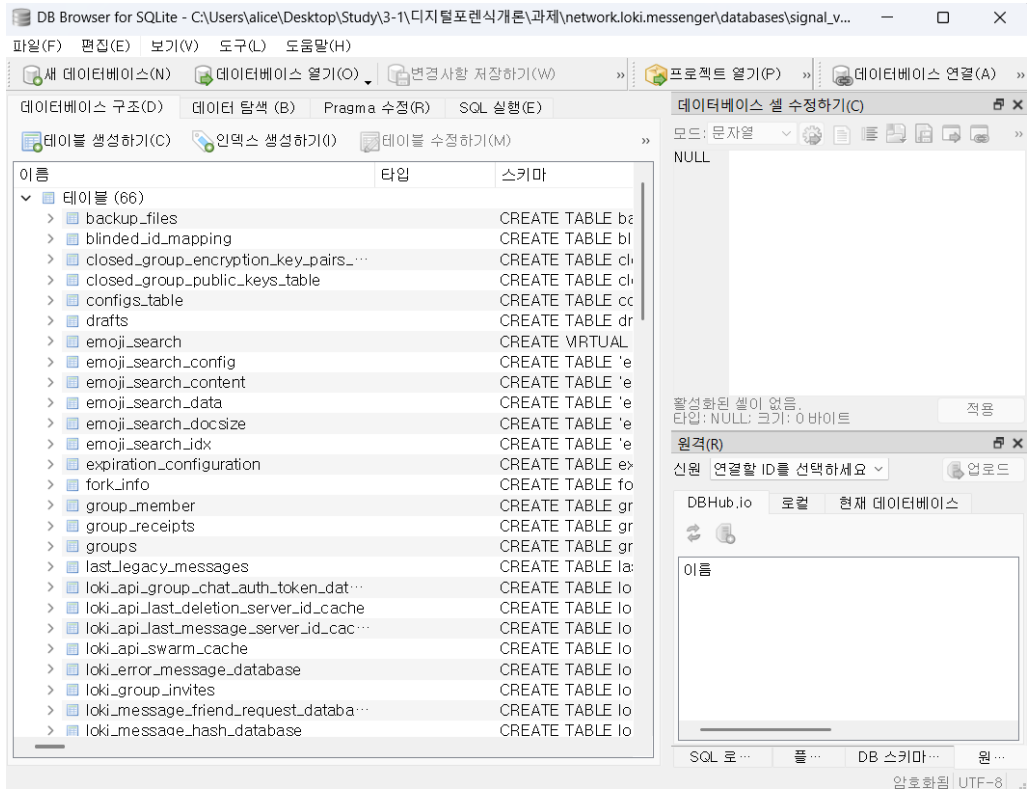
향후 연구 계획

보다 자세한 아티팩트 확인 → 표 형식으로 정리

androidx.work.workdb



signal_v4.db



Q&A

감사합니다

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