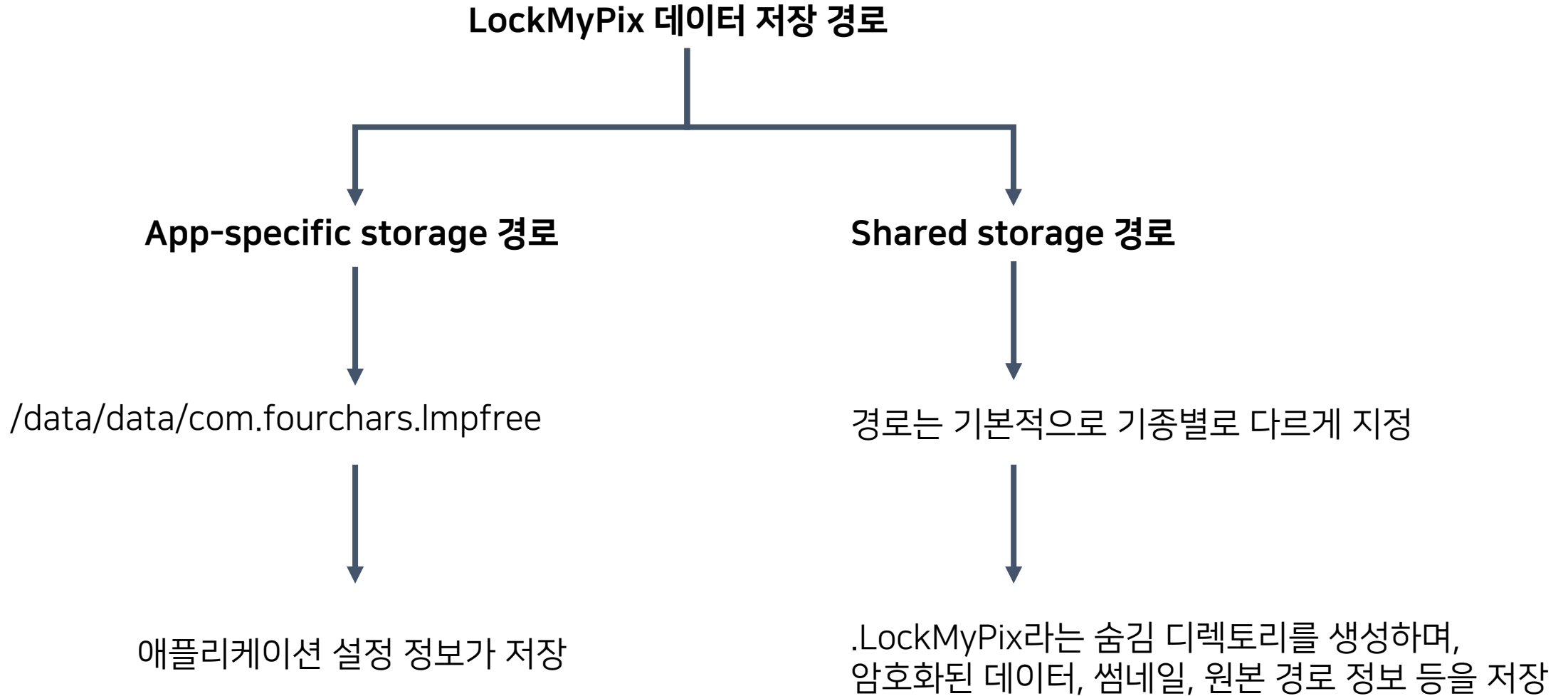

FaS

LockMyPix 실습

LockMyPix란

- LockMyPix 애플리케이션은 사진 및 동영상 파일을 암호화하여 숨김 처리된 특정 디렉토리에 복사하여 저장한다.
- 이때, 원본 데이터는 삭제되지 않으나, 사용자의 기기의 갤러리 애플리케이션에서는 보이지 않게 된다.
- 또한, 원본 데이터에서 파일을 복사한 후에 암호화하여 저장하기 때문에 원본 파일을 삭제하여도 LockMyPix 애플리케이션에서 사용자가 선택하여 설정한 패스워드를 입력하면 원본 데이터의 내용을 확인할 수 있다.
- 패스워드는 애플리케이션의 초기 설정 시 지정하며, 애플리케이션 내에서 설정 메뉴를 통해 변경할 수 있다.

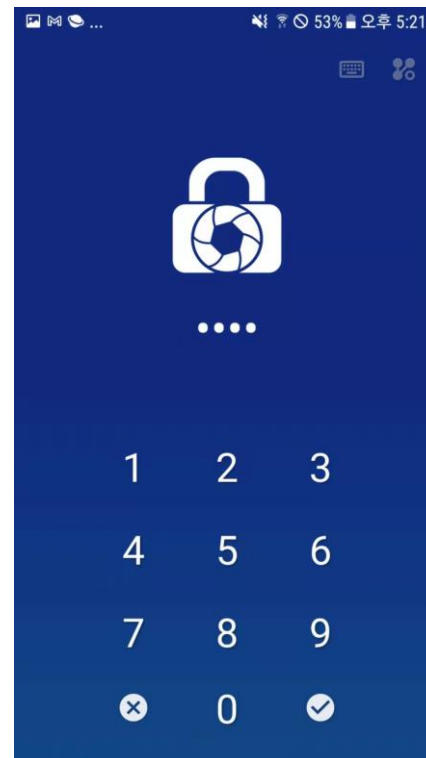
LockMyPix 구조



LockMyPix 데이터 추출 실습



갤럭시 S6 edge

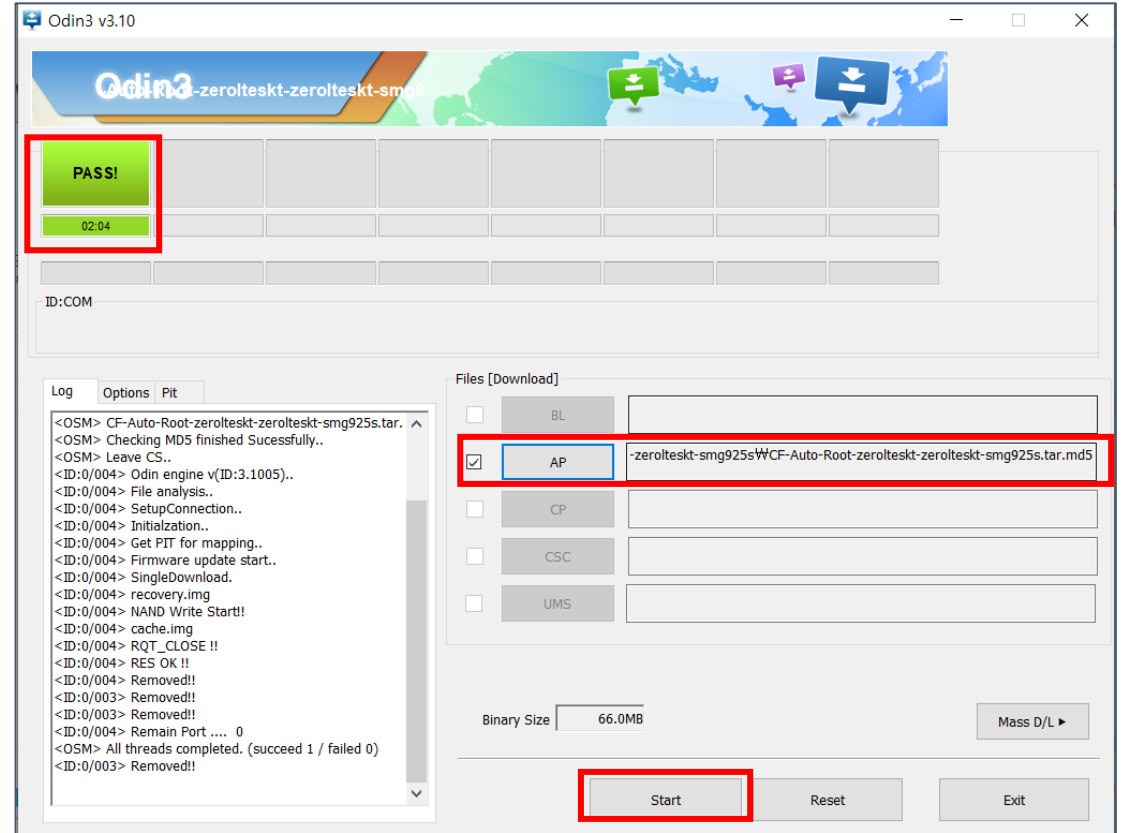


LockMyPix 4.5.4.2

루팅 후 adb pull 데이터 추출

→ Odin을 이용해 루팅하기

1. 스마트폰 리커버리 모드 진입 → 초기화 진행
2. PC에 스마트폰 모델번호에 맞는 루팅파일 설치
3. 스마트폰과 PC 연결
4. 스마트폰 다운로드 모드 진입
5. PC에서 Odin 실행 후 AP에 설치한 루팅파일 넣기
6. 다운로드 시작
7. PASS라고 뜨면 루팅 완료



LockMyPix 데이터 추출 - adb pull

1. App-specific storage 경로

```
C:\Users\kimsu\AppData\Local\Android\Sdk\platform-tools>adb shell
```

```
zerolteskt:/ $ su  
zerolteskt:/ # cd data/data  
zerolteskt:/data/data # ls
```

```
com.enhance.gameservice  
com.expway.embmsserver  
com.fourchars.Impfree  
com.gd.mobicore.pa
```

com.fourchars.Impfree 확인

```
zerolteskt:/data/data # cp -r /data/data/com.fourchars.Impfree /sdcard/appstorage
```

/data/data/com.fourchars.Impfree를 보기 쉽게 /sdcard/appstorage에 복사

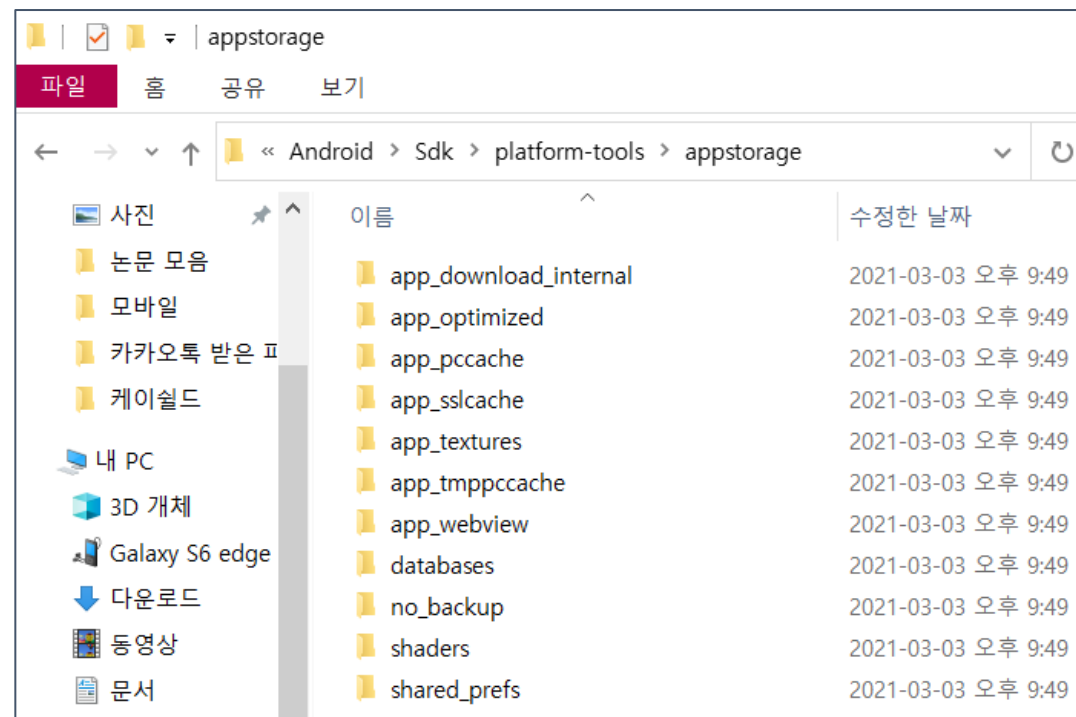
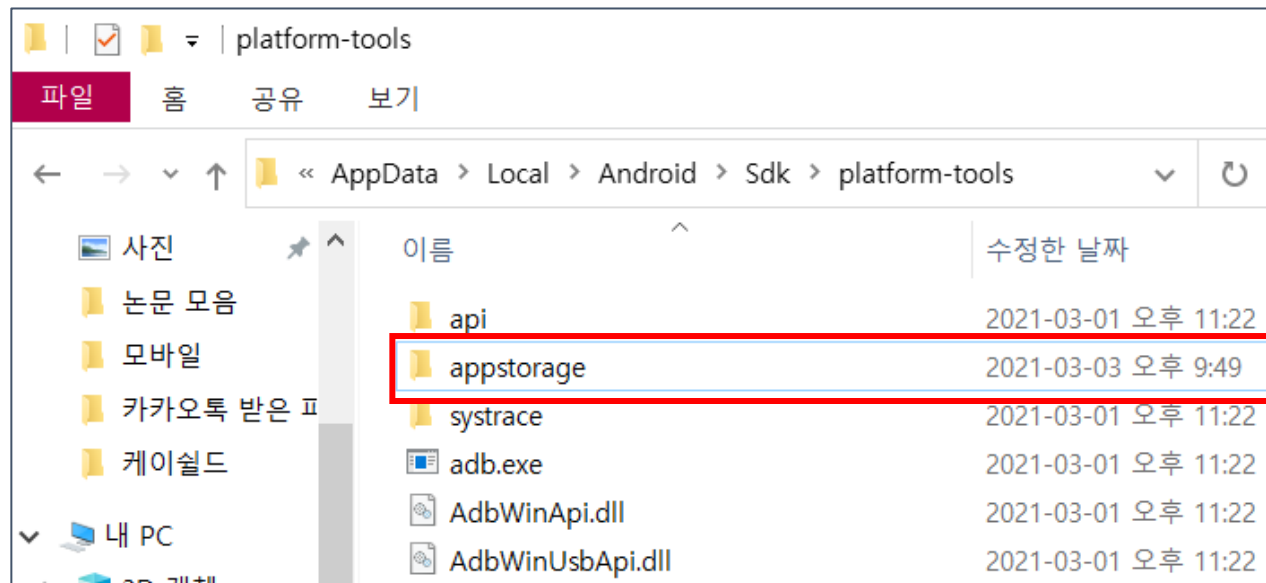
adb shell 들어가기
su로 권한 획득
data/data 위치로 이동

LockMyPix 데이터 추출 - adb pull

1. App-specific storage 경로

adb pull 명령어로 /sdcard/appstorage 데이터 획득

```
C:\Users\kimsu\AppData\Local\Android\Sdk\platform-tools>adb pull /sdcard/appstorage
adb: error: cannot create '.\appstorage\shared_prefs\fr_1:896395563931:android:1105b97041cfa7c7_firebase_settings.xml': No such file or directory
```



LockMyPix 데이터 추출 - adb pull

1. App-specific storage 경로

```
C:\Users\kimsu\AppData\Local\Android\Sdk\platform-tools>adb shell pm list packages -f | find "fourchars"  
package:/data/app/com.fourchars.lmpfree-1/base.apk=com.fourchars.lmpfree  
  
C:\Users\kimsu\AppData\Local\Android\Sdk\platform-tools>adb pull /data/app/com.fourchars.lmpfree-1/base.apk lockmypix.apk  
/data/app/com.fourchars.lmpfree-1/base.apk: 1 file pulled, 0 skipped. 19.4 MB/s (11598560 bytes in 0.570s)
```

1. com.fourchars.lmpfree의 apk 경로 찾기
2. adb pull 명령어로 lockmypix.apk 추출

LockMyPix 데이터 추출 - adb pull

2. Shared storage 경로

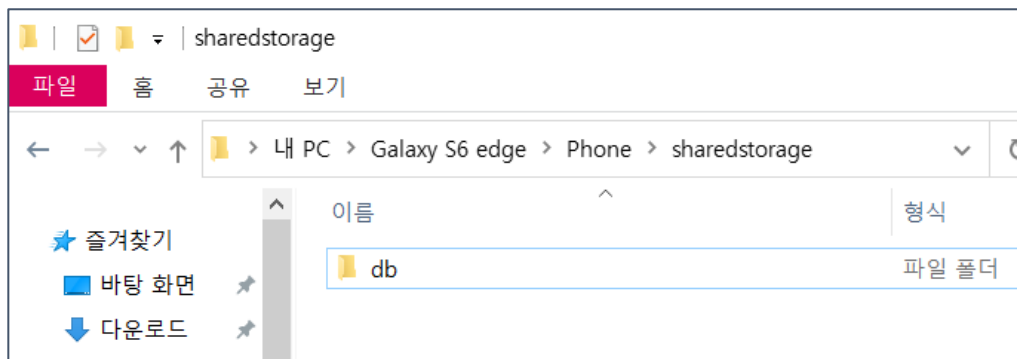
경로는 기종별로 다르게 지정, .lockmypix 폴더가 숨김으로 저장되어 있음.

```
C:\Users\kimsu\AppData\Local\Android\Sdk\platform-tools>adb shell
zerolteskt:/ $ su
zerolteskt:/ # find . -name ".ini.keyfile.ctr"
./storage/emulated/0/.LockMyPix/.ini.keyfile.ctr
```

- find 명령어를 이용해 .ini.keyfile.ctr 파일이 존재하는 경로 찾기
- shared storage 경로는 /storage/emulated/0/.LockMyPix 인 것을 알 수 있다.

```
2|zerolteskt:/storage/emulated/0/.LockMyPix # cp -r /storage/emulated/0/.LockMyPix /sdcard/sharedstorage
```

- /storage/emulated/0/.LockMyPix를 /sdcard/sharedstorage에 복사



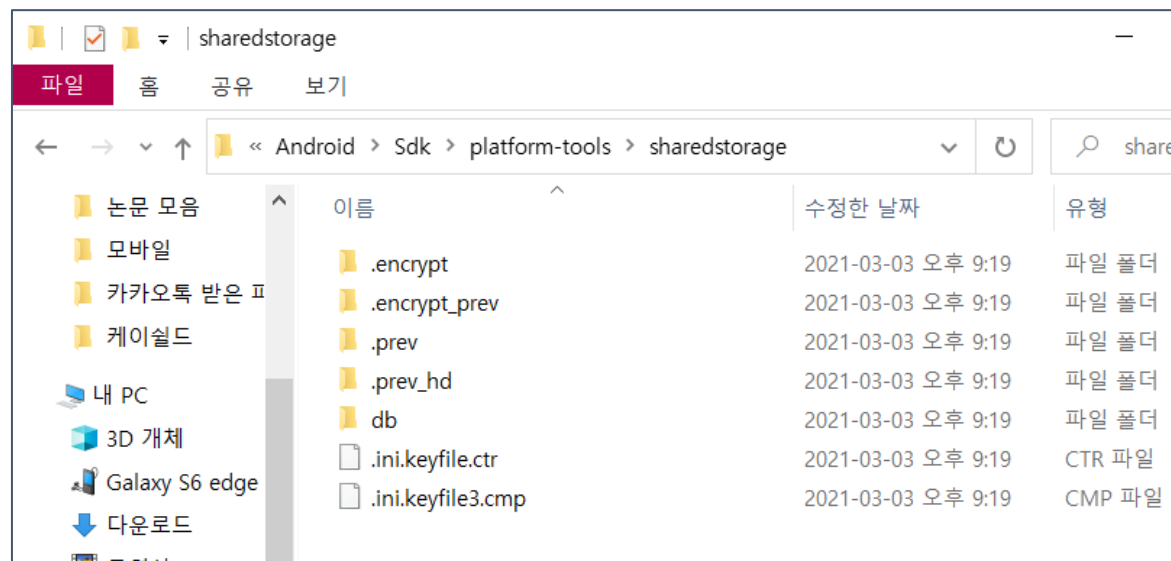
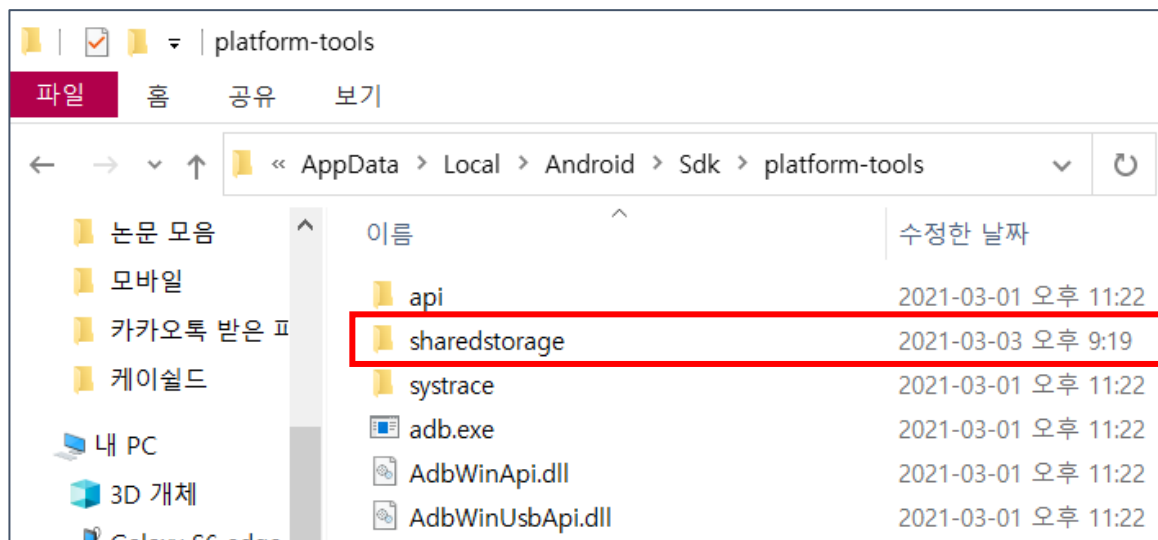
- 숨김파일로 되어있어서 sdcard에서는 파일이 보이지 않는다.

LockMyPix 데이터 추출 - adb pull

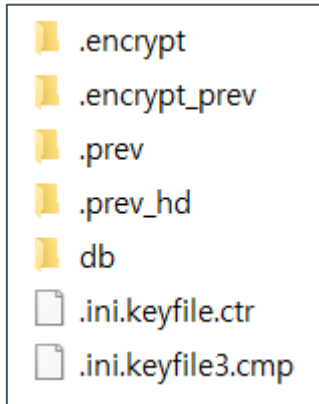
2. Shared storage 경로

```
C:\Users\kimsu\AppData\Local\Android\Sdk\platform-tools>adb pull /sdcard/sharedstorage  
/sdcard/sharedstorage/: 10 files pulled, 0 skipped. 12.0 MB/s (4155176 bytes in 0.332s)
```

→ adb pull 이용하여 /sdcard/sharedstorage 데이터 추출



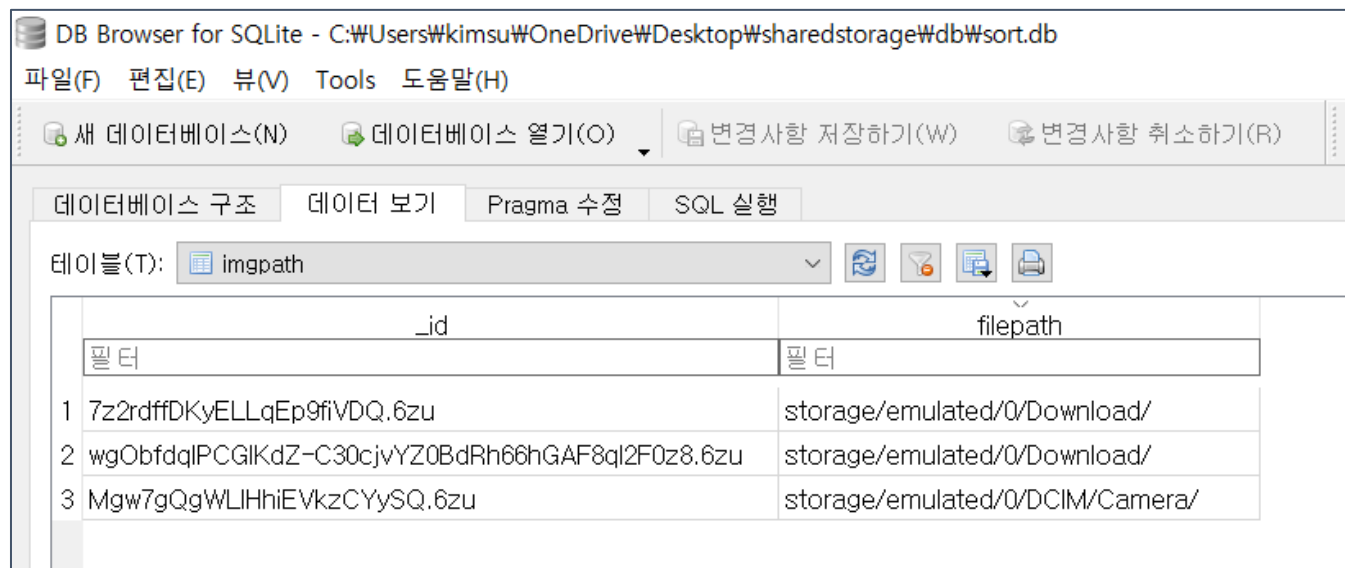
LockMyPix Shared Storage 데이터 분석



File/Directory	Description
.encrypt	원본 파일을 암호화한 데이터를 저장
.encrypt_prev	암호화한 데이터 파일의 원본이 아니라 썸네일 파일을 이용하여 애플리케이션에서 사용자가 설정한 데이터 구조를 나타냄
.prev	애플리케이션에서 이용하기 위한 썸네일 사진을 저장
.prev_hd	애플리케이션에서 썸네일을 클릭했을 때 보여주는 큰 사진을 저장
db	암호화된 파일의 원본 경로를 저장하고 있는 sort.db 파일이 저장
.ini.keyfile.ctr	사용자가 설정한 PIN 번호 또는 패스워드가 암호화되어 저장
.ini.keyfile3.cmp	사용자가 설정한 복구 이메일을 암호화하여 저장

LockMyPix Shared Storage 데이터 분석

sharedstorage/db/sort.db



The screenshot shows the 'DB Browser for SQLite' application. The title bar indicates the file path: 'C:\Users\Wkimsu\OneDrive\Desktop\sharedstorage\ddb\sort.db'. The menu bar includes '파일(F)', '편집(E)', '뷰(V)', 'Tools', and '도움말(H)'. The toolbar contains buttons for '새 데이터베이스(N)', '데이터베이스 열기(O)', '변경사항 저장하기(W)', and '변경사항 취소하기(R)'. The '데이터베이스 구조' tab is active, showing the 'imgpath' table. The table has two columns: '_id' and 'filepath'. The data is as follows:

	_id	filepath
필터	필터	
1	7z2rdffDKyELLqEp9fiVDQ.6zu	storage/emulated/0/Download/
2	wgObfdqIPCGIKdZ-C30cjvYZ0BdRh66hGAF8qI2F0z8.6zu	storage/emulated/0/Download/
3	Mgw7gQgWLIHhiEVkzCYySQ.6zu	storage/emulated/0/DCIM/Camera/

'sort.db' 파일의 imgpath 테이블에는 변환된 파일명과 원본 파일의 경로가 저장됨
→ 파일의 이름도 암호화 프로세스를 거쳐 저장되었음을 알 수 있다.

LockMyPix Shared Storage 데이터 분석

```
public static Cipher a(int arg4) {  
    byte[] v0 = {50, 0x30, 50, 99, 98, 57, 54, 50, 97, 99, 53, 57, 49, 50, 51, 0};  
    Cipher v1 = Cipher.getInstance("AES/CBC/PKCS7Padding");  
    v1.init(arg4, g.a(v0), new IvParameterSpec(v0));  
    return v1;  
}
```

그림 2. 파일명 암호화 코드
Figure 2. File name encryption code

```
private static SecretKeySpec a(byte[] arg2) {  
    return new SecretKeySpec(arg2, "AES");  
}
```

그림 3. 파일명 암호화 키 지정 코드
Figure 3. File name encryption key designation code

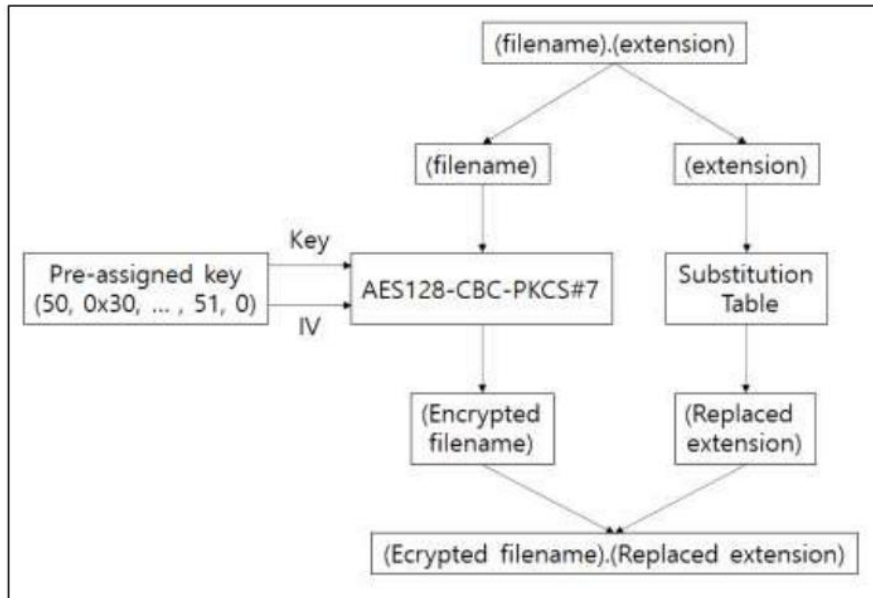


그림 4. 파일명 암호화 과정
Figure 4. File name encryption process

파일명 암호화

파일명 암호화는 파일명과 확장자를 각각 대상으로 한다.

파일명 암호화 : AES128-CBC-PKCS#7

암호화 키, IV : [50, 48, 50, 99, 98, 57, 54, 50, 97, 99, 53, 57, 49, 50, 51, 0]

1. AES-CBC-PKCS7 암호화
2. 암호화된 파일명을 Base64로 인코딩
3. Base64의 패딩 문자열을 제거

LockMyPix Shared Storage 데이터 분석

복호화 실습

아까 구해냈던 lockmypix.apk를 jeb로 열기

```
public class d0 {  
    public static String a(String arg0) {  
        return "202cb962ac59";  
    }  
  
    public static byte[] b(String arg0, int arg1) {  
        return Arrays.copyOf(arg0.getBytes(), arg1 / 8);  
    }  
  
    public static Cipher c(int arg4) {  
        StringBuilder v0 = new StringBuilder();  
        v0.append(d0.a("123"));  
        v0.append("123");  
        byte[] v0_1 = d0.b(v0.toString(), 128);  
        Cipher v1 = Cipher.getInstance("AES/CBC/PKCS7Padding");  
        v1.init(arg4, d0.d(v0_1), new IvParameterSpec(v0_1));  
        return v1;  
    }  
  
    public static SecretKeySpec d(byte[] arg2) {  
        return new SecretKeySpec(arg2, "AES");  
    }  
}
```

파일명 암호화 : AES128-CBC-PKCS7padding
key = '202cb962ac59' + '123' + '₩x00'
iv = key

→ bytes([50, 48, 50, 99, 98, 57, 54, 50, 97, 99, 53, 57, 49, 50, 51, 0])
= '202cb962ac59123₩x00'

LockMyPix Shared Storage 데이터 분석

```
from Crypto.Cipher import AES
import base64

def unpad(ct): return ct[:-ct[-1]]

enc_name = '7z2rdffDKyELLqEp9fiVDQ'
enc = base64.b64decode(enc_name + '==')

key = b'202cb962ac59123\x00'
iv = key
cipher = AES.new(key, AES.MODE_CBC, iv)
dec = cipher.decrypt(enc)
dec_name = unpad(dec).decode()
print(dec_name)
```

downloadfile

파일명 복호화하기

암호화된 파일명 : 7z2rdffDKyELLqEp9fiVDQ

1. Base64 패딩 문자열 붙이기
2. Base64로 디코딩
3. AES-CBC로 복호화
4. PKCS7 패딩 없애기

복호화된 파일명 : downloadfile

→ 파일명 복호화 성공

LockMyPix Shared Storage 데이터 분석

표 3. 확장자 치환 표
Table 3. Extension substitution table

Before substitution	After substitution	Before substitution	After substitution
mp4	vp3	ts	vo4
webm	vo1	mkv	v99
mpg	v27	mpeg	vr2
avi	vb9	dpg	vv3
mov	v77	flv	v91
wmv	v78	rmvb	v81
dv	v82	vob	vz8
divx	vz9	asf	vi2
ogv	vi3	h263	vi4
h261	v1u	h264	v6m
f4v	v2u	jpeg / jpg	6zu
m4v	v76	gif	tr7
ram	v75	png	p5o
rm	v74	bmp	8ur
mts	v3u	tif / tiff	33t
webp	20i	dng	v92
heic	v93	ps	r89
eps	v91	3gp	v79
3gpp	v80		

확장자 복호화하기

확장자의 암호화는 기존 확장자에 일대일대응 하는 확장자로 치환하여 암호화

암호화된 파일 확장자 : 6zu

복호화된 파일 확장자 : jpeg

파일 : 7z2rdffDKyELLqEp9fiVDQ.6zu
→ downloadfile.jpeg

LockMyPix Shared Storage 데이터 분석

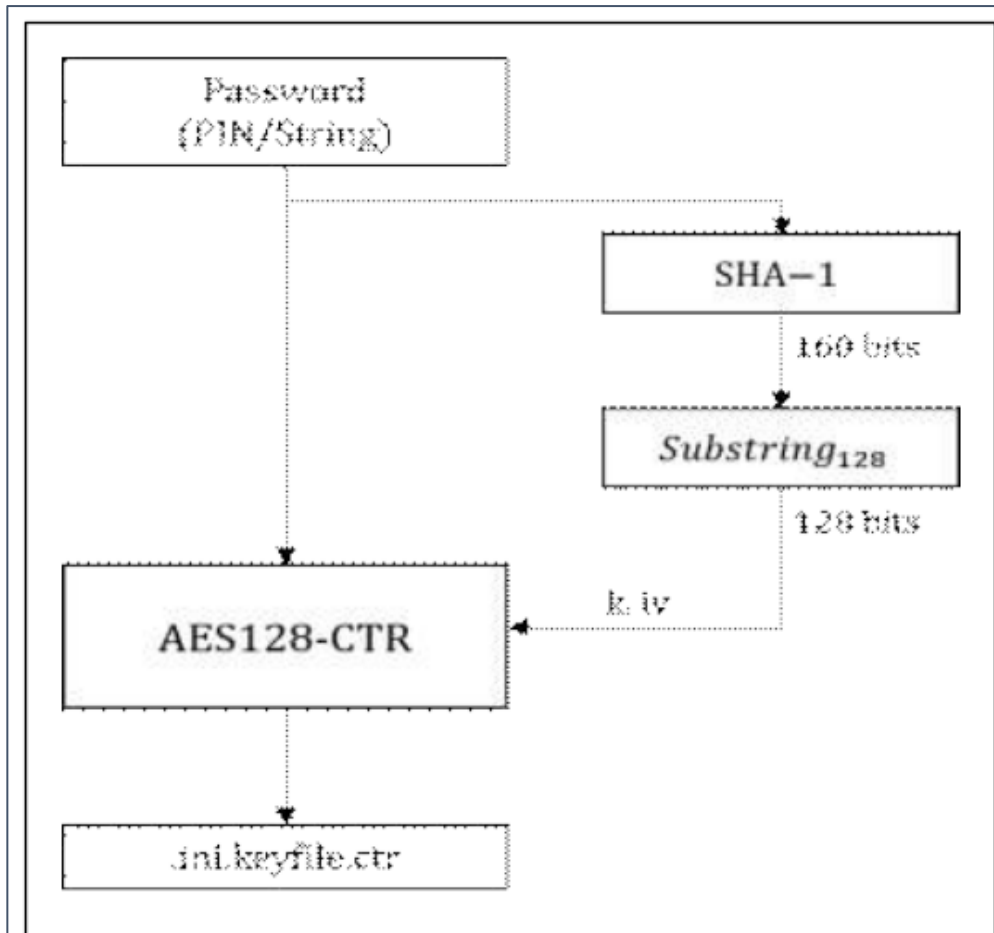


그림 5. 사용자 비밀번호 암호화 과정

Figure 5. User password encryption process

패스워드 암호화

1. 패스워드를 SHA-1 해시함수로 해시
2. 해시 값 160bit에서 상위 128bit 추출
→ key, iv로 사용
3. 패스워드를 AES128-CTR로 암호화

→ 키를 구할 때 패스워드가 사용되므로,
.ini.keyfile.ctr을 복호화하여 패스워드를 찾지 못한다.

LockMyPix Shared Storage 데이터 분석

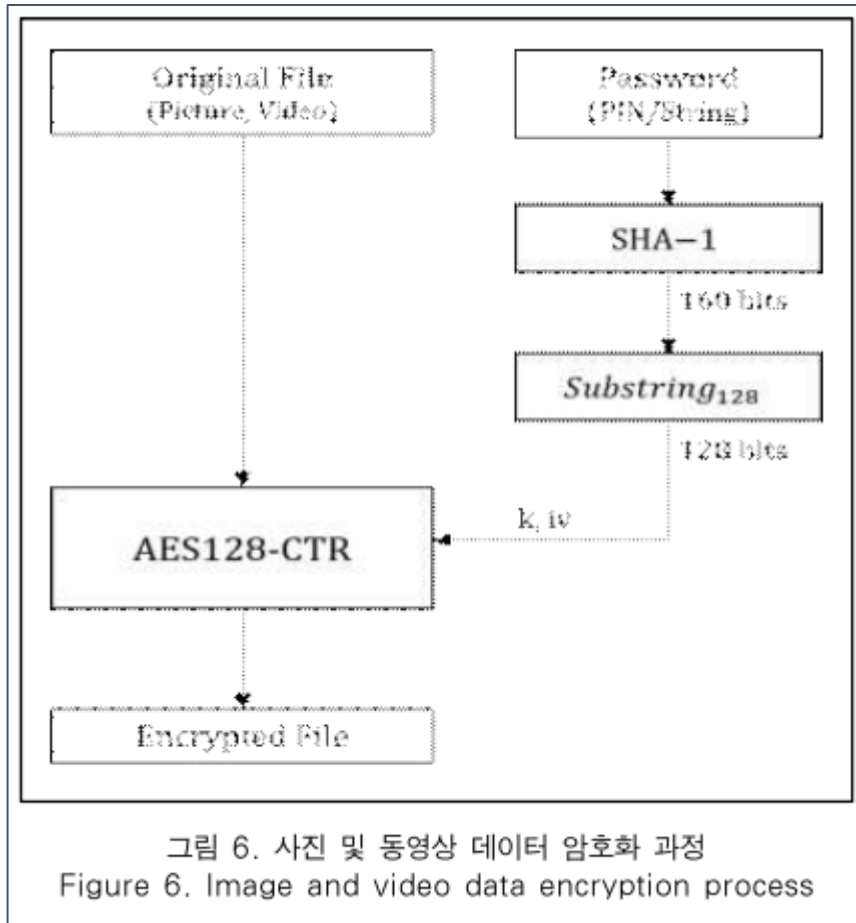


사진 및 동영상 암호화

1. 패스워드를 SHA-1 해시함수로 해시
2. 해시 값 160bit에서 상위 128bit 추출
→ key, iv로 사용
3. 원본 파일을 AES128-CTR로 암호화

→ 패스워드 암호화와 키 구하는 과정이 같다.

LockMyPix Shared Storage 데이터 분석

패스워드 추출 방법

AES-CTR의 특성을 이용해
nonce와 key가 동일한 평문, 암호문 쌍이 둘 이상 있을 경우,
평문끼리 xor 연산한 값이 암호문끼리 xor 연산한 값과 같다.

$$C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2$$

P1에는 패스워드, C1에는 패스워드를 암호화한 '.ini.keyfile.ctr'을 대입

$$(PIN \text{ number or Password}) = (.ini.keyfile.ctr) \oplus C_2 \oplus P_2$$

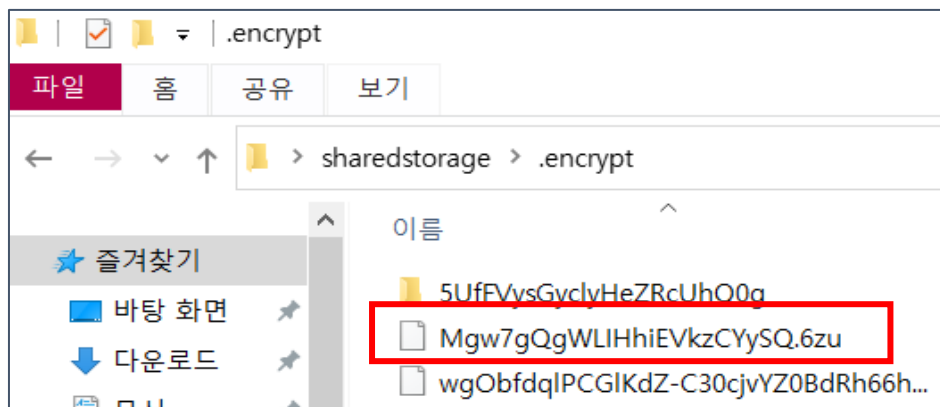
사진 암호화에서 nonce와 key를 동일한 값을 쓰므로
P2에는 원본 파일, C2에는 암호화 파일을 대입

→ 원본파일의 경로를 알기 때문에, 삭제되지 않은 원본파일이 하나라도 존재한다면
평문과 암호문 한 쌍을 구할 수 있고, 패스워드를 추출할 수 있다.

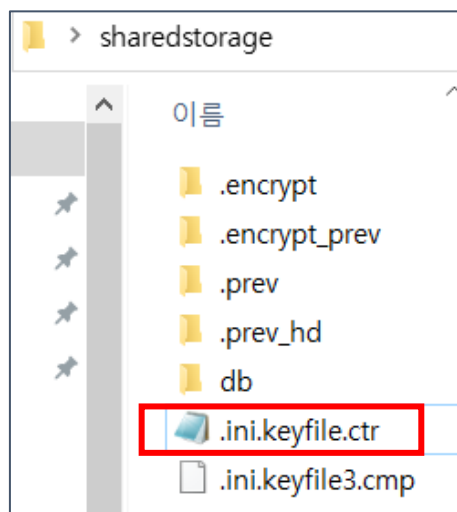
LockMyPix Shared Storage 데이터 분석

패스워드 추출 실습

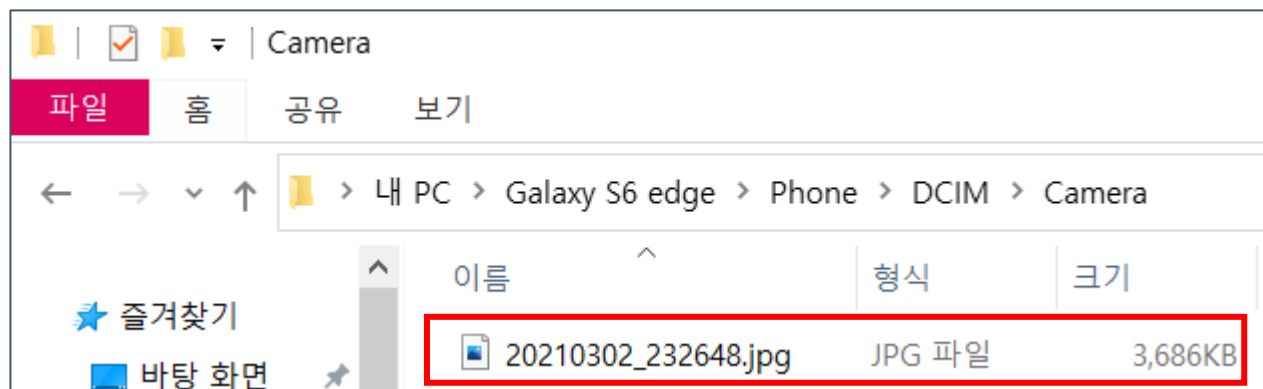
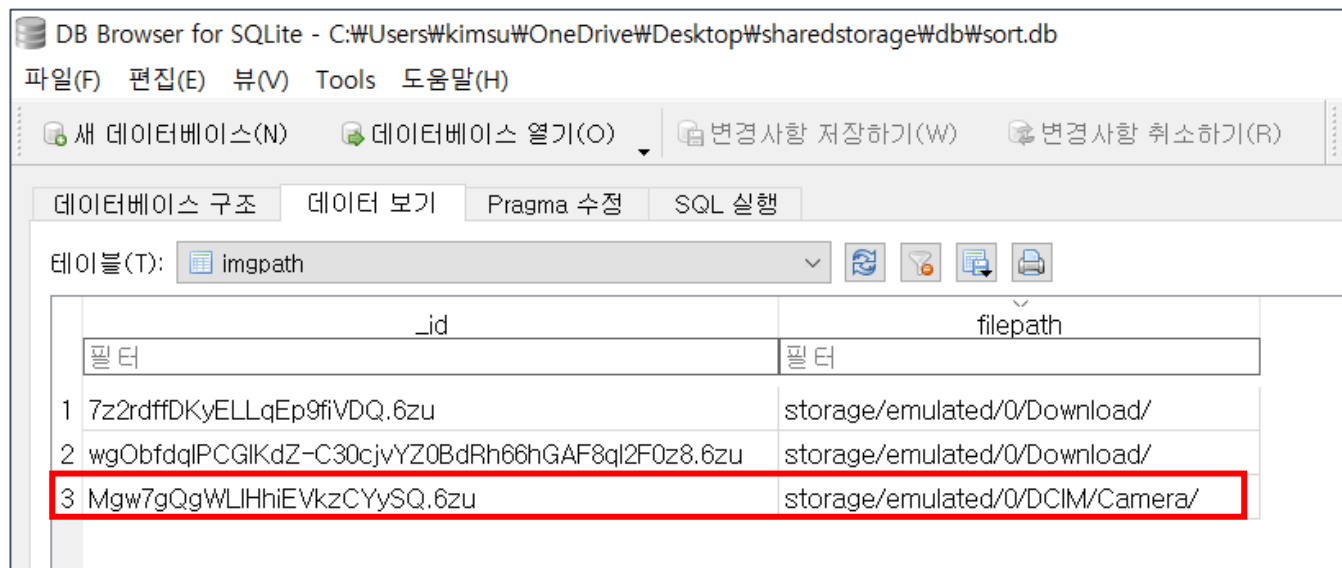
암호화된 사진 파일 얻기



암호화된 패스워드
파일 얻기



원본 사진 파일 얻기



LockMyPix Shared Storage 데이터 분석

```
with open('Mgw7gQgWLIHhiEVkzCYySQ.6zu', 'rb+') as a:
    with open('20210302_232648.jpg', 'rb+') as b:
        with open('.ini.keyfile.ctr', 'rb+') as c:
            enc_img_byte = a.read(1)
            dec_img_byte = b.read(1)
            enc_pw_byte = c.read(1)

            password = ''
            while enc_pw_byte:
                decode = ord(enc_img_byte) ^ ord(dec_img_byte) ^ ord(enc_pw_byte)
                password += chr(decode)

                enc_img_byte = a.read(1)
                dec_img_byte = b.read(1)
                enc_pw_byte = c.read(1)

            print(password)
```

패스워드 추출

각 byte를 ord로 변환 후 xor 해주기

LockzöPixContent

원래 패스워드 값인 '1234'가 나오지 않고
'LockzcPixContent' 문자가 나왔다.

→ 패스워드 추출 실패

LockMyPix Shared Storage 데이터 분석

```
import hashlib
from Crypto.Cipher import AES
from Crypto.Util import Counter
import Crypto
import base64

f = open('7z2rdffDKyELLqEp9fiVDQ.6zu', 'rb+')
enc_img = f.read()
f.close()

key = hashlib.sha1('1234'.encode()).digest()[:16]
iv = key
print(int.from_bytes(iv, 'big'))

counter = Counter.new(128, initial_value=int.from_bytes(iv, 'big'))
cipher = AES.new(key, AES.MODE_CTR, counter=counter)
dec_img = cipher.decrypt(enc_img)

f = open('downloadfile.jpeg', 'wb+')
f.write(dec_img)
f.close()
```

암호화된 사진 파일 복호화하기

패스워드 '1234' 알고있다고 가정

암호화된 사진 파일 : 7z2rdffDKyELLqEp9fiVDQ.6zu

1. 패스워드를 sha1으로 해싱 후 상위 16byte 추출
→ iv, key로 사용
2. 암호화된 사진 파일 AES-CTR로 복호화
3. downloadfile.jpeg에 저장



→ 복호화 된 사진 파일

→ 사진 복호화 성공