

203 - Ooops, your files have ben encrypted!

Team Information

Team Name : HSPACE

Team Member : Jinung Lee, Beomjun Park, DoHyeon Kim, Soyoung Cho

Email Address : hspacedigitalforensicslab@gmail.com

Teams must:

- Provide a detailed, step-by-step description of their problem-solving approach to ensure reproducibility by another examiner.
- List all tools used to arrive at their conclusions.

Tools used:

Name:	HashTab	Publisher:	implbits
Version:	v6.0.0		
URL:	https://hashtab.softonic.kr/		

Name:	Python	Publisher:	Python
Version:	3.10.9		
URL:	https://python.org		

Name:	zip	Publisher:	Greg Roelofs.
Version:	3.0		
URL:	https://infozip.sourceforge.net/		

Name:	FTK Imager	Publisher:	exterro
Version:	4.7.1.2		
URL:	https://www.exterro.com/digital-forensics-software/ftk-imager		

Name:	반디집	Publisher:	Bandisoft
Version:	7.36		
URL:	https://kr.bandisoft.com/bandizip/		

Step-by-step methodology:

문제를 풀기에 앞서 대회 측에서 제공한 파일의 해시값과 다운로드 받은 해시값이 일치하는지 확인했으며, 두 해시값이 동일함을 확인하였다.

Hash Value (MD5)

- PD12M_dataset.zip.enc :

a530d2abdebf60ee0d81b7dc6ed92c7a

그림 1. 대회에서 주어진 파일 정보

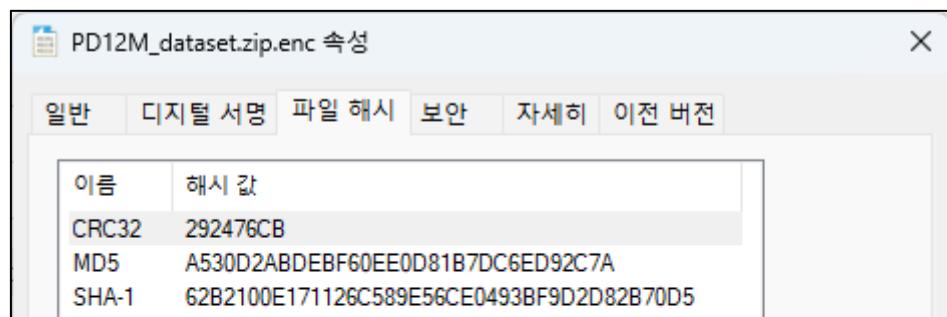


그림 2. HashTab을 이용한 해시 확인

반디집 프로그램을 통해 PD12M_dataset.zip.enc 파일을 열어보면 다음과 같은 오류를 확인 할 수 있다.

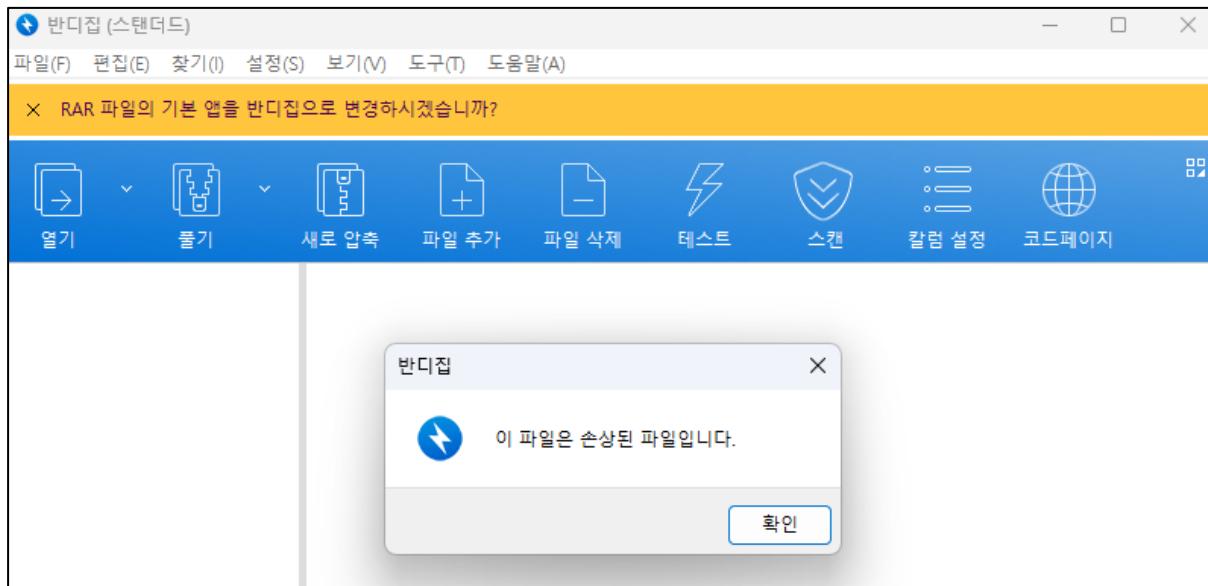


그림 3. 반디집을 통해 PD12M_dataset.zip.enc 파일 열기

오류에서 확인되는 내용은 “이 파일은 손상된 파일입니다.”이며, 확인 버튼을 클릭하면 일부 파일이 식별된다.

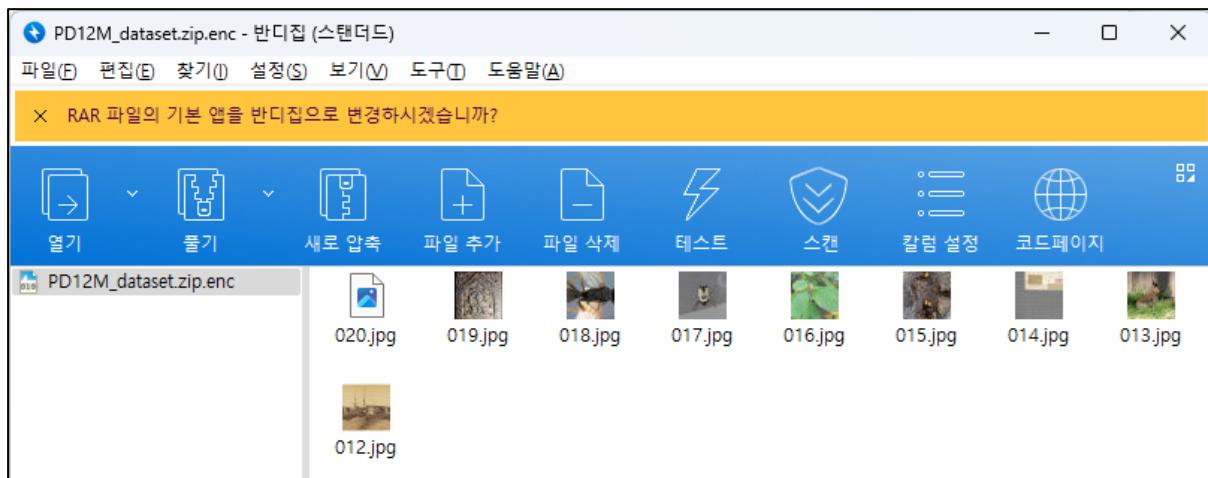


그림 4. [그림 3]에서 확인 버튼 클릭 후 보여지는 화면

추출할 수 있는 데이터는 다음[표 1]과 같다.

FileName
012.jpg
013.jpg
014.jpg
015.jpg
016.jpg
017.jpg

018.jpg
019.jpg
020.jpg

표 1. 반디집을 통해 PD12M_dataset.zip.enc에서 추출한 파일



그림 5. [그림 4]에서 확인된 파일 모습

하지만 추출된 020.jpg 파일에서 확인 할 수 있듯, 제대로 추출되지 않음을 육안으로 확인 할 수 있다. 따라서 손상된 zip 아카이브를 복구 하기 위해 zip도구를 사용한다.

zip도구의 강력모드인 -FF 인자를 통해 복구된 결과를 temp.zip으로 출력하는 명령은 다음과 같으며 [그림 6]을 통해 결과를 확인할 수 있다.

```
zip -FF PD12M_dataset.zip.enc --out temp.zip
pental@DESKTOP-O31MQLG:/mnt/c/Users/penta/Desktop/203/203_attachment$ zip -FF PD12M_dataset.zip.enc --
out temp.zip
Fix archive (-FF) - salvage what can
zip warning: Missing end (EOCDR) signature - either this archive
is not readable or the end is damaged
Is this a single-disk archive? (y/n): y
Assuming single-disk archive
Scanning for entries...
copying: 012.jpg (99883 bytes)
copying: 013.jpg (147803 bytes)
copying: 014.jpg (82886 bytes)
copying: 015.jpg (139276 bytes)
copying: 016.jpg (100985 bytes)
copying: 017.jpg (49081 bytes)
copying: 018.jpg (103083 bytes)
copying: 019.jpg (194662 bytes)
copying: 020.jpg (98978 bytes)
copying: 029.jpg (103168 bytes)
copying: 030.jpg (58410 bytes)
copying: 031.jpg (72680 bytes)
copying: 032.jpg (136124 bytes)
copying: 033.jpg (130695 bytes)
copying: 034.jpg (89660 bytes)
copying: 035.jpg (201695 bytes)
copying: 036.jpg (124239 bytes)
```

그림 6. zip 명령어를 사용한 강제 복구 과정 및 결과

[그림 6]의 결과로 temp.zip에서 추출할 수 있는 데이터는 [그림 7]과 같다.

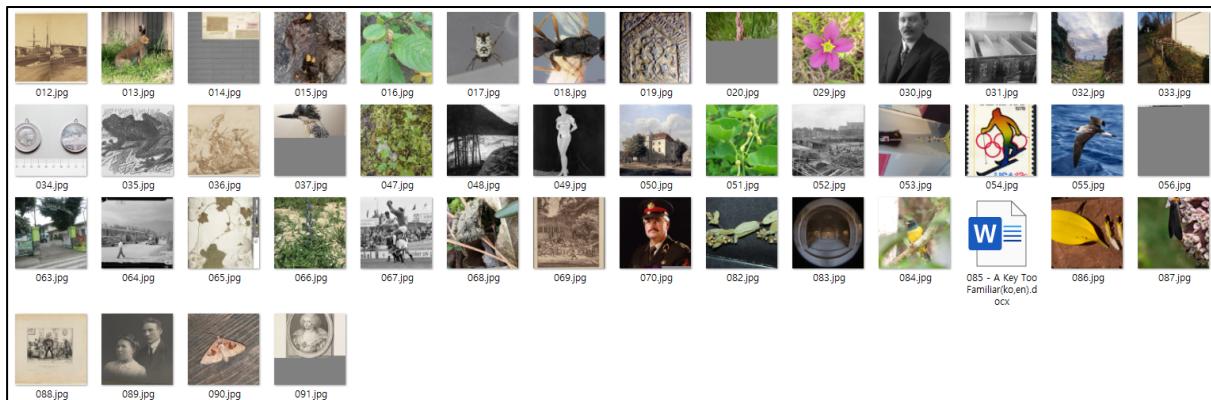


그림 7. temp.zip에서 발견된 46개의 데이터

총 46개의 데이터를 확인할 수 있으며, 그 중 085번의 파일은 docx파일로 확인되었다. 해당 docx파일에서는 “**100 – A Key Too Familiar**” 문제를 추가적으로 확인 할 수 있다.

203번의 문제를 풀기 위해서는 100번 문제를 필수적으로 풀어야 하기에, 해당 문제 풀이를 먼저 진행한다.

Instructions

Description 당신은 컴퓨터에 존재하는 랜섬웨어 프로그램을 역공학 하였습니다. 결과, 암호화에 사용된 키는 찾아낼 수 없었지만, 랜섬웨어에 취약점이 있음을 확인하였습니다. 암호화된 파일들을 분석하여 FLAG를 찾아내시기 바랍니다.

Hint.

- 랜섬웨어는 스트림 암호를 사용하였습니다.
- FLAG는 바탕화면에 존재하는 FILE_H에 저장되어 있습니다.
- FILE_A ~ FILE_H는 ASCII 문자열로만 구성되어 있습니다.
- FLAG는 FILE_A ~ FILE_H 만으로도 획득 가능합니다.

Target	Hash (MD5)
Disk image (.img)	D330AF1D7D67349334F96B902652945D Download url: https://www.dropbox.com/scl/fi/9u0jepu2z80mkvgdlg4ub/diskdump.img?rlkey=6x8mfdw557vkd29nmf3deqmpv&st=i9kf1zjy&dl=0

Credits

1) FLAG를 획득함 (100 points)

HashTab 을 이용하여 다운로드 URL을 통해 받은 diskdump.img 파일과 문제에서 주어진 MD5의 해시값이 동일함을 확인하였다.



그림 8. HashTab을 이용한 diskdump.img 파일 해시 정보

다운로드 받은 img 파일을 FTK Imager에 탑재해 분석을 진행한다.

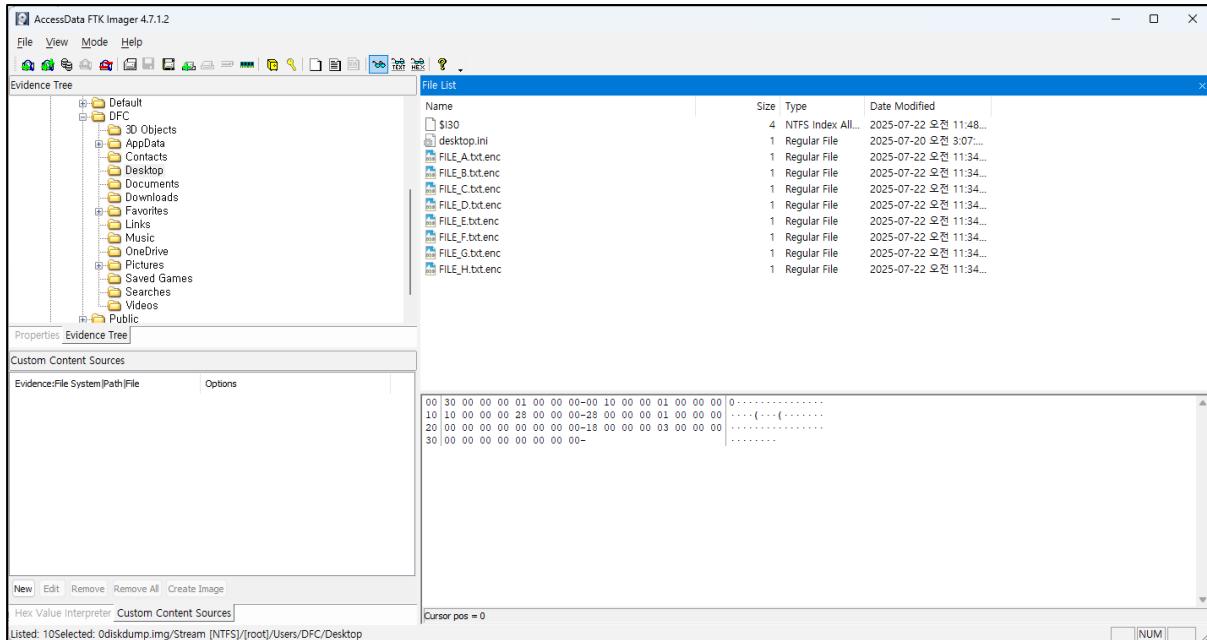


그림 9. FTK Imager을 사용하여 diskdump.img 인식 모습

FTK Imager을 통해서 [NTFS]/[root]/Users/DFC/Desktop 폴더에서 FILE_A ~ H.txt 파일 총 8개를 확인 할 수 있다. 또한 [NTFS]/[root]/Users/DFC/Downloads 폴더에서 각 프로그램 설치파일 또한 enc 확장자를 가진 채 암호화된 모습을 확인 할 수 있다.

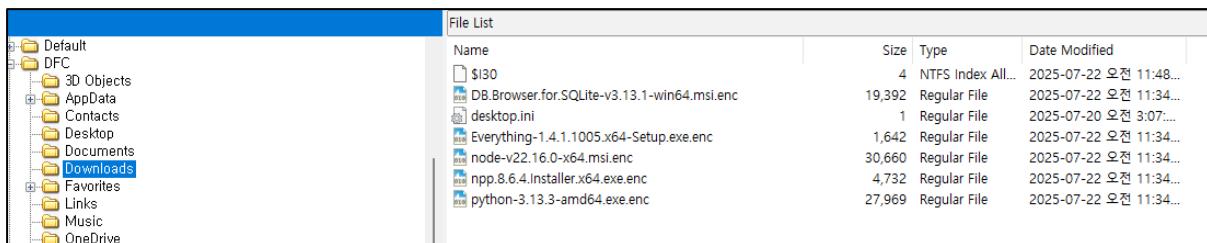


그림 10. Downloads 폴더에서 확인된 암호화된 파일들

분석을 위해, Everything-1.4.1.1005.x64-Setup.exe, python-3.13.3-amd64.exe, node-v22.16.0-x64.msi 파일을 공식 홈페이지를 통해서 다운로드 했다. 아래는 각 다운로드 버전 및 링크이다.

FileName	Version	URL
Everything	1.4.1.1005.x64	https://www voidtools com/ko-kr/support/everything/previous_versions/
python	3.13.3-amd64	https://www.python.org/downloads/release/python-3133/
node	v22.16.0	https://nodejs.org/en/blog/release/v22.16.0/

표 2. 분석을 위해 사용된 설치 파일 정보

이 케이스에서는 랜섬웨어가 스트림 암호를 사용했다는 것이 주어졌고, 동일한 키스트림을 여러 파일에 재사용했다고 전제하고 시작한다. 따라서 각 암호화된 파일과 원본에 대해서 XOR 연산 결과를 비교하여 키스트림 추출을 진행한다.

```
encrypted_file = "python-3.13.3-amd64.exe.enc"
original_file = "python-3.13.3-amd64.exe"
keystream_output = "python-keystream"

with open(encrypted_file, "rb") as enc_fp, open(original_file, "rb") as orig_fp:
    enc_data = enc_fp.read()
    orig_data = orig_fp.read()

if len(enc_data) != len(orig_data):
    print(f"[!] Warning: File sizes are off ({len(enc_data)} vs {len(orig_data)}), will chop to the shorter one.")

length = min(len(enc_data), len(orig_data))
keystream = []
for i in range(length):
    keystream.append(enc_data[i] ^ orig_data[i])

xor_result = bytes(keystream)

with open(keystream_output, "wb") as out_fp:
    out_fp.write(xor_result)

print(f"[+] XOR complete. Output written to '{keystream_output}' ({len(xor_result)} bytes.)")
```

표 3. XOR 연산을 통한 키스트림 추출 코드

위 코드를 수행한 결과 python-keystream 파일이 생성되었으며 아래는 해당 파일의 해시 값 및 정보이다.

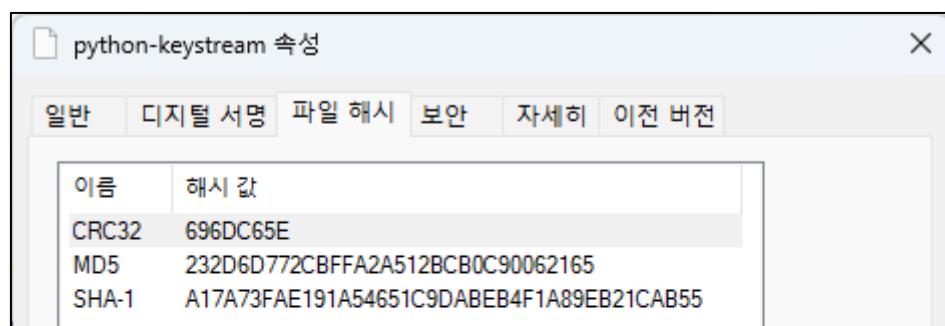


그림 11. 추출된 keystream의 해시값 정보

FileName	python-keystream
FileSize	28,640,016 바이트
MD5	232D6D772CBFFA2A512BCB0C90062165

SHA-1	A17A73FAE191A54651C9DABEB4F1A89EB21CAB55
-------	--

표 4. keystream의 정보

추출된 키스트림을 바탕으로 바탕화면에 존재하는 FILE_A ~ H.txt 파일 복호화를 진행한다. 다음은 복호화에 사용한 코드이다.

```

import pathlib
keystream_path = "python-keystream"
encrypted_files = [
    "FILE_A.txt.enc",
    "FILE_B.txt.enc",
    "FILE_C.txt.enc",
    "FILE_D.txt.enc",
    "FILE_E.txt.enc",
    "FILE_F.txt.enc",
    "FILE_G.txt.enc",
    "FILE_H.txt.enc"
]
output_path = "TEMP.txt"
try:
    ks_bytes = pathlib.Path(keystream_path).read_bytes()
except Exception as e:
    print(f"Error: Couldn't read keystream file - {e}")
    exit(1)
with open(output_path, "wb") as out_file:
    for enc_filename in encrypted_files:
        try:
            encrypted_data = pathlib.Path(enc_filename).read_bytes()
        except FileNotFoundError:
            print(f"(!) Skipping missing file: {enc_filename}")
            continue
        except Exception as e:
            print(f"(!) Error reading {enc_filename}: {e}")
            continue
        usable_len = min(len(encrypted_data), len(ks_bytes))
        decrypted = []
        for i in range(usable_len):
            decrypted.append(encrypted_data[i] ^ ks_bytes[i])
        out_file.write(bytes(decrypted))

```

```
print(f"Decryption done. Combined output written to '{output_path}'")
```

표 5. FILE_A ~ H.txt 파일 복호화를 위해 사용된 코드

해당 코드를 실행한 결과 temp.txt 파일이 생성되었으며 아래는 텍스트 파일에서 발견된 내용이다.

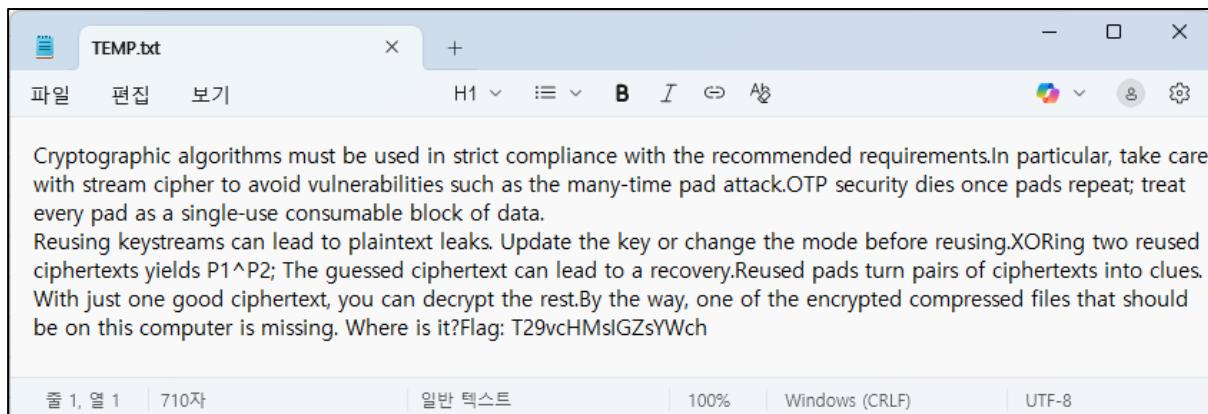


그림 12. temp.txt에서 확인 된 평문 메시지

Cryptographic algorithms must be used in strict compliance with the recommended requirements. In particular, take care with stream cipher to avoid vulnerabilities such as the many-time pad attack. OTP security dies once pads repeat; treat every pad as a single-use consumable block of data.

Reusing keystreams can lead to plaintext leaks. Update the key or change the mode before reusing. XORing two reused ciphertexts yields P1^P2; The guessed ciphertext can lead to a recovery. Reused pads turn pairs of ciphertexts into clues. With just one good ciphertext, you can decrypt the rest. By the way, one of the encrypted compressed files that should be on this computer is missing. Where is it? Flag: T29vcHMsIGZsYWch

표 6. temp.txt 파일의 평문

Flag는 T29vcHMsIGZsYWch 으로 적혀있으며 base64를 통해 인코딩 된 것을 확인하였다. base64decode.org 를 통해 온라인 base64 디코딩을 진행한다.

Decode from Base64 format

Simply enter your data then push the decode button.

T29vcHMsIGZsYWch

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 ▾ Source character set.

Decode each line separately (useful for when you have multiple entries).

 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

Ooops, flag!

그림 13. base64decode.org를 이용한 복호화

답 : Ooops, flag!

다시 203번 문제로 돌아와 zip파일 암호화에 사용되었을 것으로 추정되는 스트림분석을 진행하였다.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	04	20	20	32	87	51	4F	EB	D0	4B	97	7C	D2	, .y2_wZ0eBKé- ö			
00000010	6D	90	27	A1	F1	FF	49	35	95	59	E9	21	68	,.Ay15*{.1;h			
00000020	97	E0	30	96	73	BC	F8	01	72	1B	30	1C	09	-mä0\$zWq4.gr..ö.s			
00000030	7D	2F	94	9B	27	9C	71	B4	1D	06	D2	7C	F4	4B)T*{.w@t.i..ö16K		
00000040	CA	F4	6D	6A	99	EF	51	56	FD	86	59	65	64	Eöm{jg!IA_6Y1Yle4			
00000050	A9	AC	S4	18	0D	BE	72	20	79	S4	CE	1F	DC	0\.,.Bn@yJU,			
00000060	ED	AB	DC	31	62	99	65	1C	43	79	BF	ED	EE	1F	.A,0\$1bPm..Ct11ij		
00000070	AF	29	26	DD	1A	02	98	89	C3	65	CD	91	0C	65Y1..,s'kæd@ö.			
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000090	11	A1	BD	70	67	50	34	57	43	28	23	CF	7D	,.n@p@G-0W.Hf17			
000000A0	DT	7B	DB	8A	9B	C1	D9	59	2B	2A	70	54	08	16	C4	(.U\,,.AU\-.pt..h	
000000B0	DE	37	14	51	BC	3B	7D	7B	AC	09	08	28	D2	02..C4..k*-...,*&e			
000000C0	CF	24	C3	C9	19	1A	BB	00	07	28	31	F4	49	1E	I5@l..,i..gl1@I		
000000D0	88	37	99	91	37	39	45	B1	77	21	9C	52	42	C7	,.7*^9H..L.. o@		
000000E0	92	51	A3	95	95	D7	E7	AB	D7	50	41	6C	7C	EA	25	3A	Q6..*o@p@FAl1@:
000000F0	4E	30	31	10	82	12	D8	95	6A	B6	A7	C3	61	3D	DB	No1,..,0..jg\$@a@	
00000100	71	34	1B	35	E4	68	34	13	96	44	8C	B2	3F	C4	55	q4K5bAh..-D@E7@U	
00000110	42	49	97	BB	DE	EE	03	98	4B	3E	8A	05	FE	BK=U1U..,K@-S@			
00000120	OE	CA	85	37	92	FB	11	BE	42	87	8D	08	22	.E..78..0..W@y..,/			
00000130	ZB	EO	87	97	DC	00	35	CE	24	BD	9E	9A	DO	00	.UL51\$ö..,ö.B		
00000140	F4	10	70	02	0C	1E	F4	2B	SD	9C	14	D4	BF	12	CB	ö..p...od+1@.ö.E	
00000150	A3	99	4D	20	62	8C	95	78	68	CA	4B	27	D2	02	00	Ca..i@z..(HeR1K@	
00000160	82	A9	49	0D	22	82	C1	E7	7D	BA	44	9D	95	90	1E	,.m..,ç@p..D..v..	
00000170	04	B5	30	RE	56	10	27	BD	CA	D3	AD	31	01..,ö@P..,ö.R..i				
00000180	6E	34	S3	A6	OD	93	CA	F3	49	AT	01	03	01..,S@ö..,ö.S@ö..,ö.I				

그림 14. keystream 분석 과정 좌 node, 우 python

그림 15. 27BA74 이후 두 키스트림의 차이

오프셋 27BA74부분까지 node 키스트림과 python 키스트림이 동일한 것을 확인하였다. 하지만 python에서는 27BA74 이후 부분이 00으로 되어 있고, node 부분에는 데이터가 있는 것을 확인하였다. 이 분석을 통해서 키스트림과 널바이트의 상관 관계 분석을 진행하였으며, 그 결과 python-keystream에서는 0x27BA74부분의 2배인 0x4F74E8 부분에 동일하게 키스트림이 있는 것을 확인하였다.

그림 16. 오프셋 27BA74에서 확인된 동일한 키스트림

위에서 확인했던 46개의 데이터에서 숫자 연속성이 깨지는 부분을 확인하였으며, 001.jpg ~ 011.jpg, 021.jpg ~ 028.jpg, 038.jpg ~ 046.jpg, 057.jpg ~ 062.jpg, 071.jpg ~ 081.jpg, 092.jpg ~ 100.jpg 가 누락된 것을 확인 할 수 있다. 따라서 주어진 enc 파일에 020.jpg 뒷 부분을 확인한다.

001DF3C0	4B 24 7F 26 F0 A8 41 17 97 90 01 54 E3 64 65 03 K\$.&8~A.-..Täde.
001DF3D0	0C B0 2C 49 15 F5 F1 82 8E 1E 9C 5E E9 6B F7 83 .°, I.őñ, Ž.œ^ék÷f
001DF3E0	8B 4A 2A 6A D6 5A 9F FF D9 50 4B 03 04 14 00 02 <J*jÖZÝyÜPK.....
001DF3F0	00 08 00 15 AC F3 5A 5D A8 60 AB A2 82 01 00 3B¬óZ] ``«¢,...;
001DF400	83 01 00 07 00 00 00 30 32 30 2E 6A 70 67 9C B7 f.....020.jpgœ·
001DF410	65 50 5C 5D F8 ED D9 04 0B 1E DC 21 C1 43 23 41 eP\]øiÜ...Ü!ÁC#A
001DF420	83 93 84 00 21 04 08 EE 10 20 C1 1A D7 06 1A 09 f"!...i. Á.×...
001DF430	EE 2E C1 93 E0 DA 41 9A C6 DD 35 B8 5B E3 DE 8D i.Á"àÚAsÆY5, [äP.
001DF440	36 3E 79 FF B7 66 E6 CB 7C 98 7B D7 A9 BD EB 9C 6>yÝ·fæÈ ~{×@¾œœ
001DF450	DA 7B 9F 5A BF F3 54 9D 5A CF E3 DC E3 1A E0 D9 Ú(ÝZçÓT.ziäÜä.àÙ
001DF460	47 05 25 05 00 06 06 00 80 F1 EF 02 3C 2E 02 E4 G.%....€ñi.<.ä

그림 17. 020.jpg 를 검색한 모습

020.jpg 다음으로 발견되는 jpg 문자열은 029.jpg이다. 따라서 020.jpg 내용부터 029.jpg가 발견된 부분을 따로 저장해 분석을 진행했다. (0x1DF40F ~ 0x2DE60D)

복호화에 사용한 키스트림의 길이와 시작 위치를 찾기 위해 키스트림 파일(key)을 이용한 XOR 브루트포스 복호화 실험을 수행하였다. 우선 key 파일의 앞부분을 이용해 암호화된 ZIP 파일 전체를 XOR으로 복호화해 보았고, 복호화가 정상적으로 진행되는 시작 지점이 0x1DF40F에서 1079 바이트 떨어진 위치임을 확인하였다. 이를 통해 총 키 길이가 $0x1DF40F + 1079 = 1,964,102$ 바이트로 추정되었다.

키스트림이 순환 특성을 가지므로 실제 암호화의 시작점이 위에서 구한 길이보다 얼마만큼 더 떨어져 있을 수 있음을 고려해야 했다. 특히 시작 위치가 4096바이트 단위로 어긋날 가능성이 있어, 스크립트를 일반화하고 브루트포스 방식으로 여러 후보 길이를 시도하도록 확장하였다. 키 길이를 늘려가며 각 경우에 대해 복호화 파일을 생성하였다. 그 결과 생성된 ZIP들 가운데 1개 파일이 정상적으로 압축해제 되었고 100장의 이미지 파일을 복구할 수 있었다. 사용한 코드는 다음과 같다.

```
def _xor_bytes(data, key):
    if not key:
        raise ValueError("키가 비어있습니다: 'python-keystream' 파일을 확인하세요")
    out = bytearray(len(data))
    klen = len(key)
    for idx in range(len(data)):
        out[idx] = data[idx] ^ key[idx % klen]
    return bytes(out)

with open('python-keystream', 'rb') as kf:
    keystream = kf.read()
with open('PD12M_dataset.zip.enc', 'rb') as ef:
    encrypted = ef.read()
```

```

for i in range(0, 100) :
    part_len = 982051 + 2048 * i
    prefix = keystream[:part_len]
    pad_unit = b'\x00'
    zeros = pad_unit * part_len
    predict_key = prefix + zeros

    decrypted = _xor_bytes(encrypted, predict_key)
    out_name = f'decrypted_{i}.zip'
    with open(out_name, 'wb') as out_f:
        out_f.write(decrypted)

```

표 7. 복구에 사용된 코드

decrypted_9.zip - 복디렉터리 (스캔 대상)					
	파일	문서	보기	설정	도구
<input checked="" type="checkbox"/> PARDirector 기본 암호로 파일을 복구하시겠습니까?					
	열기	폴더	새로운 폴더	파일 추가	파일 삭제
	테스트	스캔	설정	코드레이저	
decrypted_9.zip	이름	작성자	파일 크기	파일 종류	수정한 날짜
009.jpg	159,889	160,059	JPG 파일	2025-07-19 오후 9:34:27	A409C7CC
008.jpg	169,192	169,485	JPG 파일	2025-07-19 오후 9:34:24	C690C775
007.jpg	80,586	80,728	JPG 파일	2025-07-19 오후 9:34:23	AE188F99
006.jpg	115,732	115,910	JPG 파일	2025-07-19 오후 9:34:21	CSA9244A
005.jpg	60,755	60,912	JPG 파일	2025-07-19 오후 9:34:21	6D34675
004.jpg	133,033	133,190	JPG 파일	2025-07-19 오후 9:34:20	0B984002
003.jpg	150,085	151,109	JPG 파일	2025-07-19 오후 9:34:19	27382224
002.jpg	78,473	78,632	JPG 파일	2025-07-19 오후 9:34:16	146F5964
001.jpg	133,254	133,375	JPG 파일	2025-07-19 오후 9:34:15	F6C44766
090.jpg	144,780	144,924	JPG 파일	2025-07-19 오후 9:34:14	E5987CA
089.jpg	53,162	53,334	JPG 파일	2025-07-19 오후 9:34:13	E41770F6
088.jpg	93,698	93,859	JPG 파일	2025-07-19 오후 9:34:12	9592A501
72.jpg	72,000	72,000	JPG 파일	2025-07-19 오후 9:34:10	0807ED09
085.jpg	133,012	134,044	JPG 파일	2025-07-19 오후 9:34:09	C2009002
085_A Key Too Familiar(ko,en).docx	22,905	26,076	Microsoft Word 문서	2025-07-31 오후 8:19:13	E5320EE1
084.jpg	66,579	66,903	JPG 파일	2025-07-19 오후 9:34:06	2A71BC5E
083.jpg	68,339	68,476	JPG 파일	2025-07-19 오후 9:34:05	009872A2
082.jpg	133,380	133,523	JPG 파일	2025-07-19 오후 9:34:03	EFE0D0BC
081.jpg	158,097	158,225	JPG 파일	2025-07-19 오후 9:34:00	0E88947E
080.jpg	79,145	79,280	JPG 파일	2025-07-19 오후 9:34:01	D4803538
079.jpg	108,331	108,398	JPG 파일	2025-07-19 오후 9:34:00	698A374A
078.jpg	106,012	106,155	JPG 파일	2025-07-19 오후 9:33:59	1FB6948A
077.jpg	84,375	84,507	JPG 파일	2025-07-19 오후 9:33:57	2EDAB61A
076.jpg	112,981	113,121	JPG 파일	2025-07-19 오후 9:33:55	6ED9C57A
075.jpg	82,994	83,145	JPG 파일	2025-07-19 오후 9:33:54	61E4118B
074.jpg	130,048	130,189	JPG 파일	2025-07-19 오후 9:33:53	2A0C00D9
073.jpg	144,087	145,264	JPG 파일	2025-07-19 오후 9:33:52	55107039
072.jpg	54,698	55,046	JPG 파일	2025-07-19 오후 9:33:51	58834E55
071.jpg	69,026	69,201	JPG 파일	2025-07-19 오후 9:33:49	F58A5007
070.jpg	63,892	64,051	JPG 파일	2025-07-19 오후 9:33:47	075A20E7
069.jpg	155,479	155,626	JPG 파일	2025-07-19 오후 9:33:46	00873421
068.jpg	144,427	144,589	JPG 파일	2025-07-19 오후 9:33:45	A261B98E
067.jpg	100,558	101,080	JPG 파일	2025-07-19 오후 9:33:44	1D47D01C
066.jpg	201,527	201,735	JPG 파일	2025-07-19 오후 9:33:43	B979C7C6

그림 18. 코드 실행 후 100개의 파일이 모두 복구된 모습

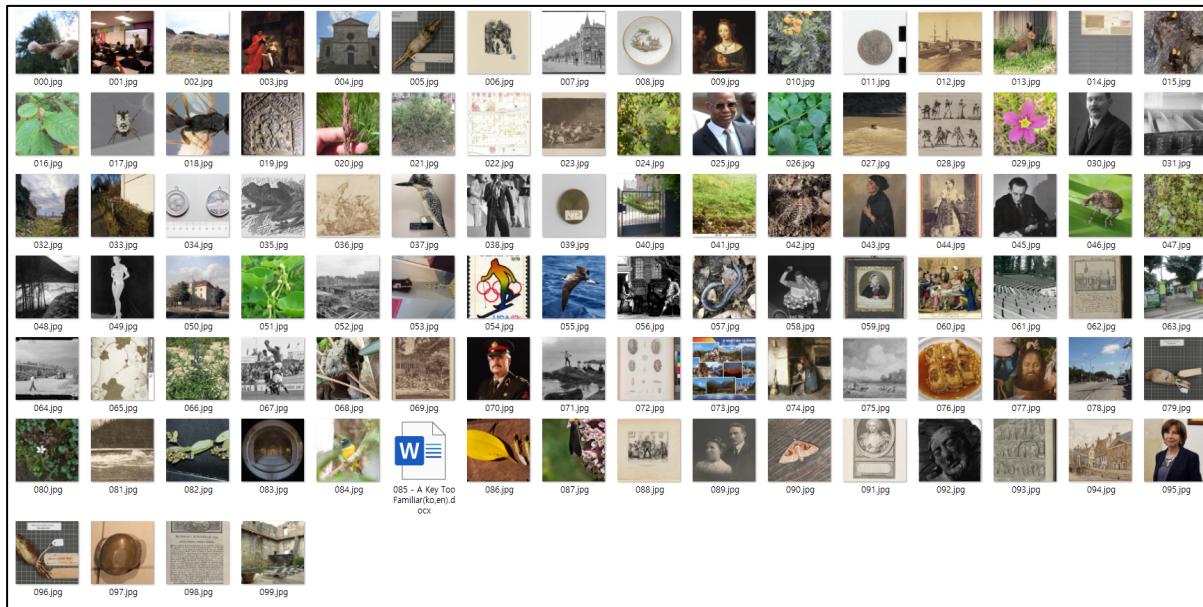


그림 19. 추출된 100개의 파일

복구 된 파일은 총 100개이며 각 파일 이름 및 해시값은 표 8과 같다.

FileName	SHA256
000.jpg	e55d9e636ce8197ecad3c52b1073de098e28cd9a02c035a1f5ada5c186c3e086
001.jpg	54272c394e867833f17099d1fd2462c9562ff33397eda9795a9a228080b2915
002.jpg	432e775467541125a8923bad3646a51688ccc7af08562ff5670fab1490ab71f
003.jpg	bad50ba579e971f73942f90a9c014bc52fef02c56155fdc600c6f262a7cd75c4
004.jpg	7602d9134fb58ac8433081e3161de088ba54da857056c31c04b20a9a924629bf
005.jpg	f4136cef4c29775390075e25eece8fd2a62dccd466969d9f0dcc02d5e11e8df
006.jpg	8488fcfe4dc24d5ffe0d1dc42dc8662b04978cf55c540c98991177f164fcfd64
007.jpg	420da57b90142851499968f85dc99616b103febbb60ace31cab72b64d9076700
008.jpg	e2a648f9729b029f3f53329409aca962d98f7763d8110f93d8ffdd882a402233
009.jpg	41891d418c99586c508cf3947426761933f200e37c68ef4776816b8fe0c6e2b4
010.jpg	73b060e408f0bf8c2133d6b5bb4fefb191ccfc108f28e90b98331797c743016
011.jpg	93e76032b7a0f0db464b5e004859f09576485e5370fdab102b7305d63638354a
012.jpg	65c682f04647569debcbc5f321d43a18c10617ea180e7db6cf45e23c0f1090e7a
013.jpg	75116d1266299dd256eb9e00dffae90054c725c9b7b4718d9903f9537675711
014.jpg	a1e51ea589d0d69757065365b845fe550a1848c962dccd4f8b81d1f0ddef1155
015.jpg	25121a3273ee88f0df1337d35aad7f3b7f10b84edfa1814b298eed43f88bd11a
016.jpg	12ed595337cef2251e9fc6ac12d592264847ffbc9e07e788988b8bf54ee5db38
017.jpg	9517cfe1e757e16eff5e3259ef8ee83fd98a40af9b954e807894382bac3fc9c0
018.jpg	d378aa5496e30b4f2f498966e5f7c608c3b86bd1c0f1d748ea1c487db8dac335
019.jpg	677ec871700acd0e0ce6b0e1f3068109438f88ce3d0cd39b895ae53c000c9042
020.jpg	283f9dba470570467c661b18dbba9cf1007d7342334e8187437ff9ea7fe2429a
021.jpg	54b63d61782c384e4bed74b7b0e984f4d7ac6e954f649f8542450cd074060366
022.jpg	32f8e76a8973c98b1400634c84917c5d6ef1468dfbbd44f271823e3cca708f42

023.jpg	bc2396495058e8b02717a58be4fad9d1c14ddcf4319e611093d30b5736851083
024.jpg	a123d84525160846f58bb2551e6a01091c871f10e835f9bb65299734bbcabec
025.jpg	70a0678489b8dc88b63803110ec8d1b8b34888607da479b5d7b102bb6725a759
026.jpg	5ef67a9a3b0f7c1f404f82f17365fe1991f44a94d85f84eeee651c252bae4575
027.jpg	e553fb2d611ec2fc617c5b5e4f4e08e5b8094b55ac594191b6e1e4c4eea4d13
028.jpg	7cd5ca48fc223a182bc10e0730247e85e4694e21f19444382c23f6df90912c3
029.jpg	119bf2d66f4226b8309dfc1e3763de5061bea8bec3379f76a17a12380db0fc7e
030.jpg	3e2863569378a3c1ad574352e5c040ace6146956f1fd70f7b03c52f292bce673
031.jpg	8db8a962144e815483844703ac04e5668b994289a631f4a47ac436b9d2bc5a20
032.jpg	1a29041185e7ceaa3684f91edcfead59a17b9b392e0aa21c97a340e4e01abb8e
033.jpg	54402ef3e746e4fafef11b8e62851f37901e3186277f75227d346d8007f2f9fe6
034.jpg	952a3b521a166c9a0298ecd6b4b6c984e64dd230cc2780d3b7eb0e74207e9956
035.jpg	e4fa96b54edd334969c1befabc50642caf4ea317f34b4c6137353b58a20c3a7f
036.jpg	95a1dc9c7b613f7692d12cb9af04b9148fdafef122cea4b4bce72cf582c96e43c
037.jpg	8a121ea8fe8a812f70e25d39eac401e4e76d6dd7097a3be33b23b8d1de47008e
038.jpg	51e688aa7eba6be0fec1986f23b3a621ac8186c2d26bbe9955e58063713d5599
039.jpg	f63c4e097b374e37256e5343a5d6c7258fd0403ce3e7e635baa6f949c4be37cc
040.jpg	f8fe607f1299b595fc6284fa9f806b1214ff4fab6eae45f07a162d884dbdf378
041.jpg	ad29d72b517e64d96553ade02aa3d9a315f539c84bed72c904a70398754a2f67
042.jpg	1941f33daa2ed8ed046d449bf1e30bfa7b0b89bc55f4561d667c2147997df53d
043.jpg	4f2e8ccafa0b37ee96729e4510f32d9f945dc5fdf46059f90d96cdb233883fe5
044.jpg	cfaade6a2cf26ea6b398605510d1365d17968439d70f4eede484e4a74a0dfd083
045.jpg	743e154ee31a780647770869fe1e81985430a403bf3d876932a34d309524970d
046.jpg	5d2d7a738f6f7f2434e412ae13ff698e7e630f128db597ec7af219616d3a41ff
047.jpg	4ae608c262f4a3690f76b6a4c43ce670b2a156be00dce90d1430fbadc21b1d68
048.jpg	575ef9765c2545e85391e608fdf9cd0f514f92bad34e6f8eb52ad4a6e3877750
049.jpg	f5bc9f8cc2ed62e53dc29c3cd8085c4015d2fdb59551dafd6ff0ef35b9efe9b
050.jpg	f8a57d0643609ff2b7dd29a1a6b1a04a866540422d5d2113c717968147da157e
051.jpg	9c47048b9f4cb927639a50b5ccce067f77b44eef7097cf827d6dd537066a4bd7
052.jpg	e38209e2ad3df0b5c36b8cbf408d7063cca58c8e256686452535829d38aedf28
053.jpg	ddea6d062b8d2b5cf2c5a8971d6a5e90670c7b3181478b348f17feab8c0ac50
054.jpg	b2a813f44fda174f4dc553b9dea1f0349f526e3cde54ce0b09f7c30ad61349ef
055.jpg	43a721782cf0aa8153dd1208d96df75913b7cc1c4fa36e7c26160d7b1a3892bb
056.jpg	4096de8557452cd4340a39f490d83d73db063379a1faa0c3aca56c676c3c358e
057.jpg	6b2173ac8b74273df65bc6198ed78fe0ffa974a512ffe2a2ca219ccedc0fccd1
058.jpg	c0ec02162ead5297f463abcdc21d10f391f3e3f53f0520253407cd6f37fb0008
059.jpg	f4881a1ddd87ae81366f9b67548fc62f9f27617d2c1b7599c7ec2cf164cf8976
060.jpg	69a1f40718d35c45cd3394b6063e2c9f0160d3610210d4ee9d222f3fa12f393
061.jpg	63919b22fdff3796c193bf7c22e8ef5f230088a62b3053fee5b865ebe0331a6d
062.jpg	6f408488ab51d95d4b88ee3c3f950e4d30e096b7a8ee30b04e46e5719eafbaee
063.jpg	f223e40609779931c233d3e520281083427d8dd21c767a837f283f9b354d9b2b
064.jpg	34c60e55adc557b77bda9a6bad312de6aa6d0cb5d9fd0ddecf83be2eaa82edf8

065.jpg	6f70487ffb30962d737d3580ffcded0e980c93af9e015fa79d7f799f593967fb
066.jpg	5e3fdc8a1c6348d990bcc164d0513d504d20d29374e3f2cff28b68eb18b02c57
067.jpg	e16e05553c6f69260b463a16077a6601071bbde3ec9bdceae72f1066d25c871f
068.jpg	9dc9d7d8476883476e517cfbc9a3b56fc793d464403cb222f72696ab4034791
069.jpg	dd5cae541faaaa5e51e649699838278e2374f4b496169756236a3c67b3b2d1e2
070.jpg	ad7ffb7bae55a0772b4fdf74f960c687170a9c381c3781d7a332ab72340a873
071.jpg	b879856dc38bf5a0de6955ffc872811254f81783f47bf78449f771c598ce4066
072.jpg	17ba314bf7b7d5537edd42f1c5d93c2a25f3ba7af65487aa1bfb9ca1b030ca0
073.jpg	f0f3e4f2f2bc57a49d4d761ac603d4a079fac7702396dcb0ee3cb1c7fd1d893d
074.jpg	ec915beb9e12f32d467298388ba678d933ba539cf92e104aaa2b281a5b3fe083
075.jpg	b1069fee3cdc995f2f93447501af29cdc8bbb8c8ee690f66757f6ab3415c89d6
076.jpg	6e3baf882e5d310cdb59d95de5fd71baf6345b68c99d4eb2a205fea5f4759407
077.jpg	f5eecbd13cd597ee214ab788ccf9574f6d4ecb7d0da845678b0cf4605f8abebe
078.jpg	2341b398277d2211453e6a8a4df61a6480a1643c5bee639bee3dbcedfca754cf
079.jpg	7188c9059aabedcbd01770ba34184fd7ed3d7f02c5d6cadb29fd441b7c932602
080.jpg	49ad230c6c11ac9124768566f6b3cb57350a2648609251ddaebe5a5793eee2af
081.jpg	9fb35e59bed3ada2069b707047a30c77d2fc4b5c5ccc0fc6aa15846564dcc8fa
082.jpg	2383505b28b2b5e01d2457d646e50aa06843880607c54c957c2e5fe6d28b158f
083.jpg	e0c55fbca33757cdde58c28a073e2ec6d8d18fb89f522c9a0e344b3c1939149d
084.jpg	7c705703060c68b8af61dadcadaf497ef90541681711af66d2e41c6976516f8c
085 - A Key Too Familiar(ko,en).docx	70bf4cf76692accca06779b384d2281b3d96a9386ac593d75b3d7da55fac9a22
086.jpg	15d43227afc5cdbba30e36cd68c521d8132859d09a1e4c6963e61fe4f07a7f8b
087.jpg	26c866c5e9bdef211d52dbc62113bd167f5ef6f3226e0604639e8a35d1e6c3
088.jpg	1a3ffe670cbe0603b53e2be18349578807495b0c4e92abea2f7c0f06e8df5b4c
089.jpg	76caa577d7f8a72bc16121f94ae9d8639353e354f52d5ade6de5114117cb1f38
090.jpg	c9dc8d000bab94e0e4694d1fae8d52c09e91a55778dd6f28ae3f0b04ebf3c44
091.jpg	ad8e0b8ee16970db43d683d82f5463eab8ae3dad260f8f51a924c71291821ac8
092.jpg	cf2e5b962894e317a139d8fcc66fcbbcceec0c8e8d62c9627cb402f28e86f35c6
093.jpg	66e787c1d66cab51602ed86b3efe9b445105b1f624627f5203a46c8aff9b12b8
094.jpg	04b4b5b135e1da4848697bb7baef7cd8332a3b3d3e0ed716fa24ac7721a23046
095.jpg	e66a9a404bb8980e4bd498b8533bb6cdce4bf9a260e9692f07e5aad4414843ea
096.jpg	2d228defbfafa7ba082be19dd25a51196d5136acde510fcc997b4523e96a6afe
097.jpg	0d1a2f1354b0a13ef2f6c66b2b7465f749d18aa24b64d8870788e60a078b30ab
098.jpg	848e1b11a6030a4d61672b9ad3fbba1309abd978e42f66f17016274e3c173f76
099.jpg	9bdc53c96af007c5887c33dbf8dfa5aeee4e81a928ae3a9e8b4514cc02169ab0b

표 8. 복구에 성공한 파일 정보