MS Office 2010

암호화 과정 분석 결과

2013-11-30

충남대학교 정보보호 연구실
전 준 희
x15kangx @ nate.com

목차

개 요

개 요

◆ 개 요

- MS Office 2010 프로그램을 MS 社에서 공개한 내용에 기반하여 리버싱으로 실제 암호 처리 과정을 분석한 내용임.

◆ 주요내용

- 분석대상 : MS Office 2010(32bit)
- 분석인원 : 전준희 외 1명
- 분석기간 : 2013. 1. 1. ~ 3. 1.(3개월)
- 분석도구 : Ollydbg, IDA, Hxd 등.
- 분석환경 : MS Windows XP Home Edition sp3

◆ 분석결과

- 발표 내용 참조

MS Office 관련 기본 내용

MS Office 관련 기본 내용

MS 社 암호화 방식 공개 내용

[MS-OFFCRYPTO]:
Office Document Cryptography Structure

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
10/06/2008	1.01	Editorial	Revised and edited the technical content
12/12/2008	1.02	Editorial	Revised and edited the technical content
03/18/2009	1.03	Editorial	Revised and edited the technical content
07/13/2009	1.04	Major	Revised and edited the technical content

•
•
•

02/11/2013	2.8	No change	No changes to the meaning, language, or formatting of the technical content.
07/30/2013	2.8	No change	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	2.8	No change	No changes to the meaning, language, or formatting of the technical content.

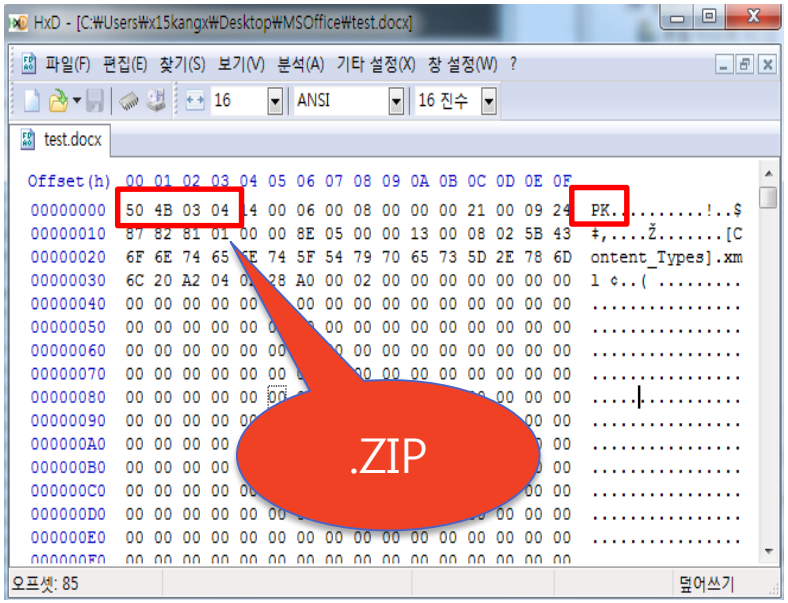
MS Office 관련 기본 내용

일반 오피스 파일과 암호화된 오피스 파일의 저장 방식 차이

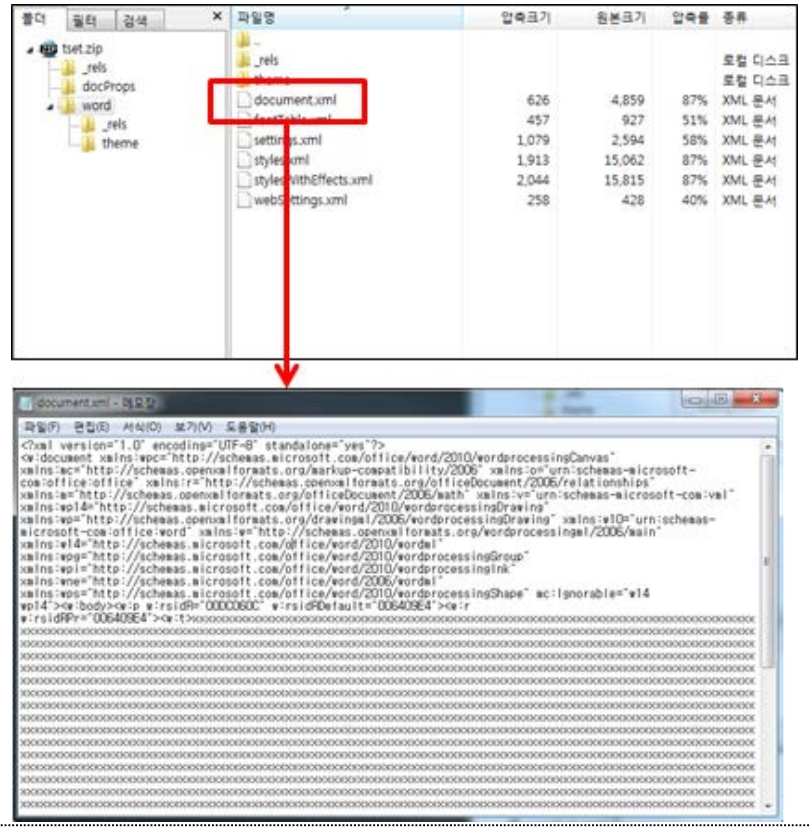
- **일반 오피스 파일 : Office Open XML 형식**
- 암호화된 오피스 파일 : Compound Document File 포멧

  ※ also called Microsoft OLE2, Structured Storage, Compound File Binary Format

Office Open XML 파일 헤더



Office Open XML 파일의 구성



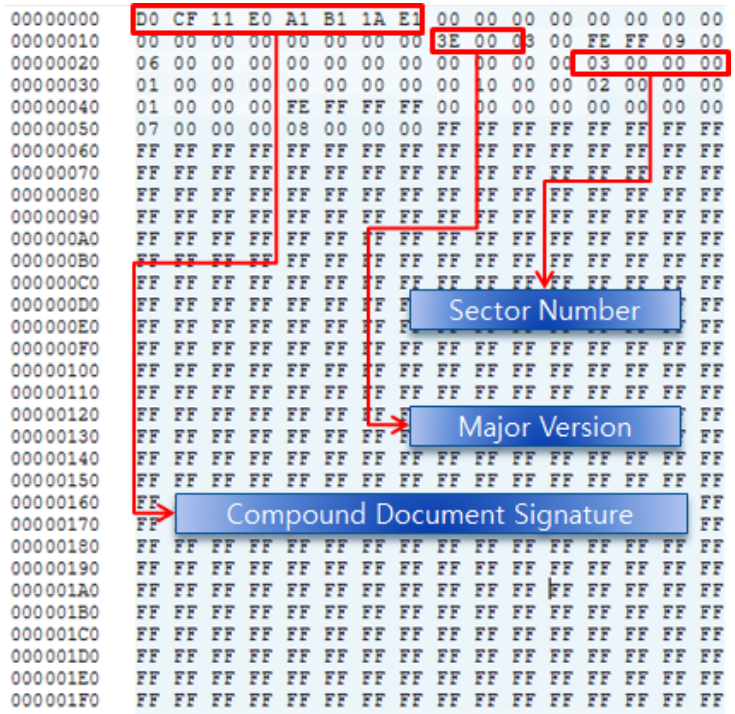
MS Office 관련 기본 내용

일반 오피스 파일과 암호화된 오피스 파일의 저장 방식 차이

- 일반 오피스 파일 : Office Open XML 형식
- **암호화된 오피스 파일 : Compound Document File 포멧**

  ※ also called Microsoft OLE2, Structured Storage, Compound File Binary Format

CDF 파일 헤더

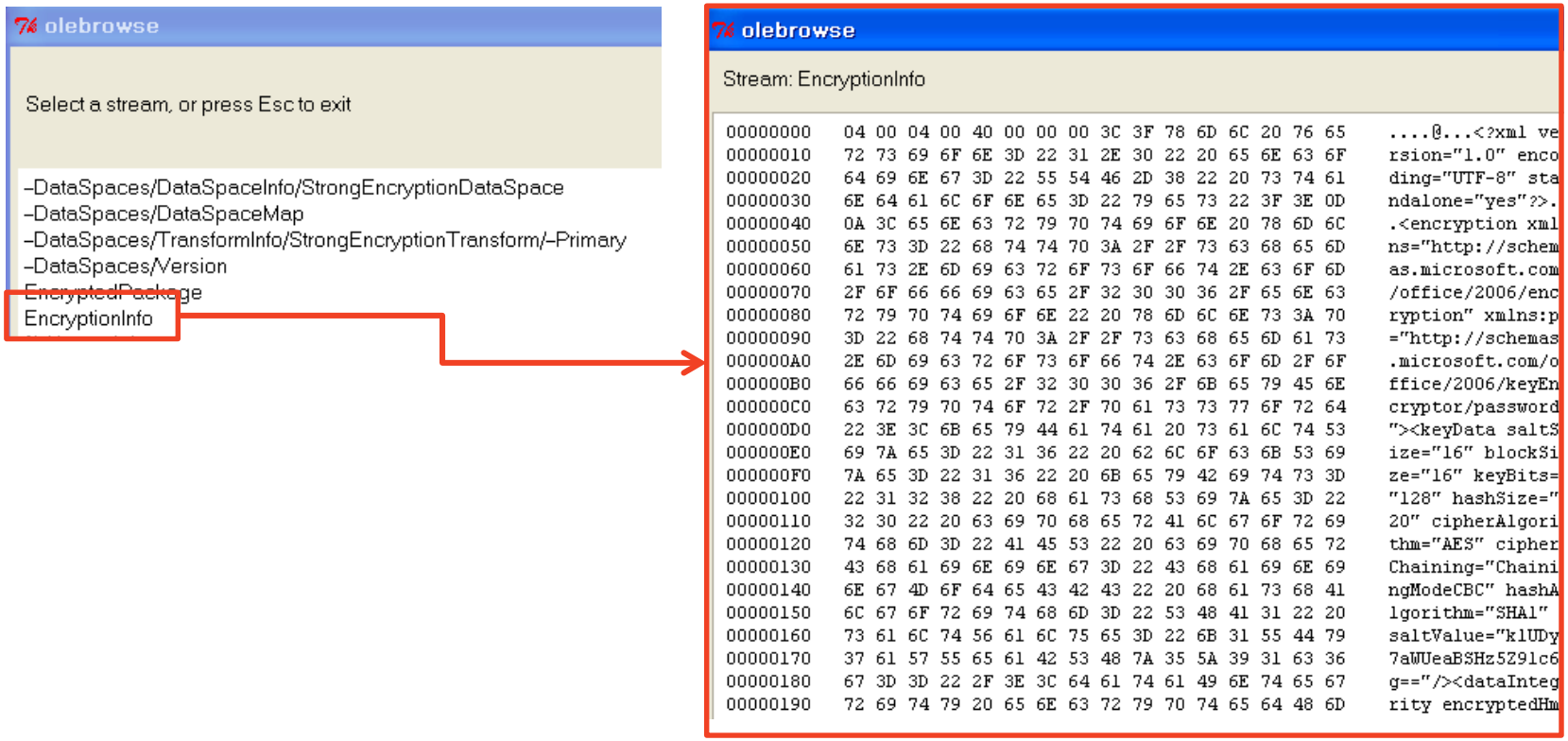


CDF 파일 포멧 內 XML 블럭

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000F40   22 31 32 38 22 20 68 61 73 68 53 69 7A 65 3D 22  "128" hashSize="
00000F50   32 30 22 20 63 69 70 68 65 72 41 6C 67 6F 72 69  20" cipherAlgori
00000F60   74 68 6D 3D 22 41 45 53 22 20 63 69 70 68 65 72  thm="AES" cipher
00000F70   43 68 61 69 6E 69 6E 67 3D 22 43 68 61 69 6E 69  Chaining="Chaini
00000F80   6E 67 4D 6F 64 65 43 42 43 22 20 68 61 73 68 41  ngModeCBC" hashA
00000F90   6C 67 6F 72 69 74 68 6D 3D 22 53 48 41 31 22 20  lgorithm="SHA1"
00000FA0   73 61 6C 74 56 61 6C 75 65 3D 22 43 2F 46 54 72  saltValue="C/FTr
00000FB0   59 34 43 57 70 39 6D 61 50 75 76 65 52 2F 66 70  Y4CWp9maPuveR/fp
00000FC0   67 3D 3D 22 2F 3E 3C 64 61 74 61 49 6E 74 65 67  g=="/><dataInteg
00000FD0   72 69 74 79 20 65 6E 63 72 79 70 74 65 64 48 6D  rity encryptedHm
00000FE0   61 63 4B 65 79 3D 22 65 4B 5A 4F 6D 64 32 36 63  acKey="eKZOmd26c
00000FF0   49 61 65 6C 65 63 57 30 63 63 6E 2F 6C 37 54 34  IaelecW0ccn/l7T4
00001000   81 00 00 00 82 00 00 00 83 00 00 00 84 00 00 00  ....‚...ƒ...„...
00001010   85 00 00 00 86 00 00 00 87 00 00 00 88 00 00 00  ….....†...‡...ˆ...
00001020   89 00 00 00 8A 00 00 00 8B 00 00 00 8C 00 00 00  ‰...Š...‹...Œ...

                              .
                              .
                              .

00243FD0   3F 82 41 6D 17 01 56 85 94 E5 42 1B 5C AF D3 32  ?‚Am..V…”åB.\¯Ó2
00243FE0   D7 41 B1 B7 E5 6C 12 D1 60 73 C4 FE 3E 3F A5 4C  ×A±·ål.Ñ`sÄþ>?¥L
00243FF0   5D 4D D0 61 4F A1 9C AD E5 10 C6 3B 2B 7E 3C FD  ]MÐaO¡œ.å.Æ;+~<ý
00244000   39 6C 37 2F 37 57 4C 79 71 42 6B 53 65 70 46 67  9l7/7WLyqBkSepFg
00244010   65 38 3D 22 20 65 6E 63 72 79 70 74 65 64 48 6D  e8=" encryptedHm
00244020   61 63 56 61 6C 75 65 3D 22 6F 46 59 79 4D 66 44  acValue="oFYyMfD
00244030   6A 4A 33 59 35 53 5A 57 34 46 43 2B 34 6C 58 6E  jJ3Y5SZW4FC+4lXn
00244040   49 5A 4F 50 57 57 51 52 35 64 76 30 62 34 31 70  IZOPWWQR5dv0b41p
00244050   77 72 4F 51 3D 22 2F 3E 3C 6B 65 79 45 6E 63 72  wrOQ="/><keyEncr
```

MS Office 관련 기본 내용

- Python 기반의 OleFileIO_PL 및 oletools 라이브러리를 활용한 CDF 포멧 확인



MS Office 관련 기본 내용

XML 블럭 내용

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<encryption xmlns="http://schemas.microsoft.com/office/2006/encryption" xmlns:p="http://schemas.microsoft.com/office/2006/keyEncryptor/password">
    <keyData
        saltSize="16"
        blockSize="16"
        keyBits="128"
        hashSize="20"
        cipherAlgorithm="AES"
        cipherChaining="ChainingModeCBC"
        hashAlgorithm="SHA1"
        saltValue="C/FTrY4CWp9maPuveR/fpg=="
    />
    <dataIntegrity
        encryptedHmacKey="eKZOmd26cIaelecW0ccn/I7T49I7/7WLyqBkSepFge8="
        encryptedHmacValue="oFYyMfDjJ3Y5SZW4FC+4IXnIZOPWWQR5dv0b41pwrOQ="
    />
    <keyEncryptors>
        <keyEncryptor uri="http://schemas.microsoft.com/office/2006/keyEncryptor/password">
        <p:encryptedKey
            spinCount="100000"
            saltSize="16"
            blockSize="16"
            keyBits="128"
            hashSize="20"
            cipherAlgorithm="AES"
            cipherChaining="ChainingModeCBC"
            hashAlgorithm="SHA1"
            saltValue="T5JIFF6I8mn19u4QE7VnLw=="
            encryptedVerifierHashInput="zYaDErh6vXBApsxKCI2M6Q=="
            encryptedVerifierHashValue="6U9p0H4TJR1bRi1p9H7hRYYvRnnrgC4OY7X1Scxzezw="
            encryptedKeyValue="n4CjMfCHt+y7zkbZz9qd2A=="
        />
        </keyEncryptor>
    </keyEncryptors>
</encryption>
```

Data key

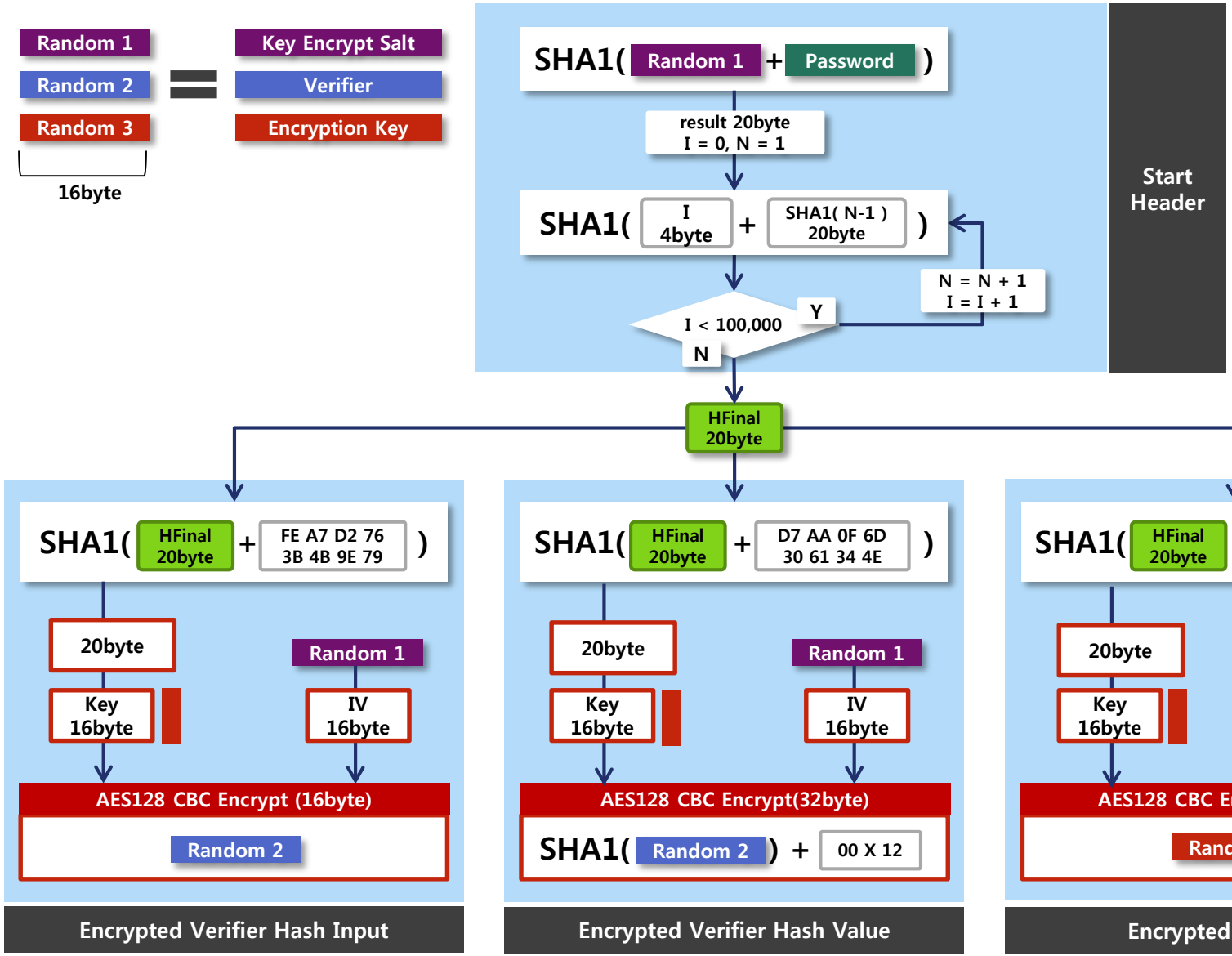
Data Integrity

Key Encryptors

결론 2-1

- 암호키 암호화
- 데이터 암호화
- 무결성 검사값 암호화

암호키 암호화

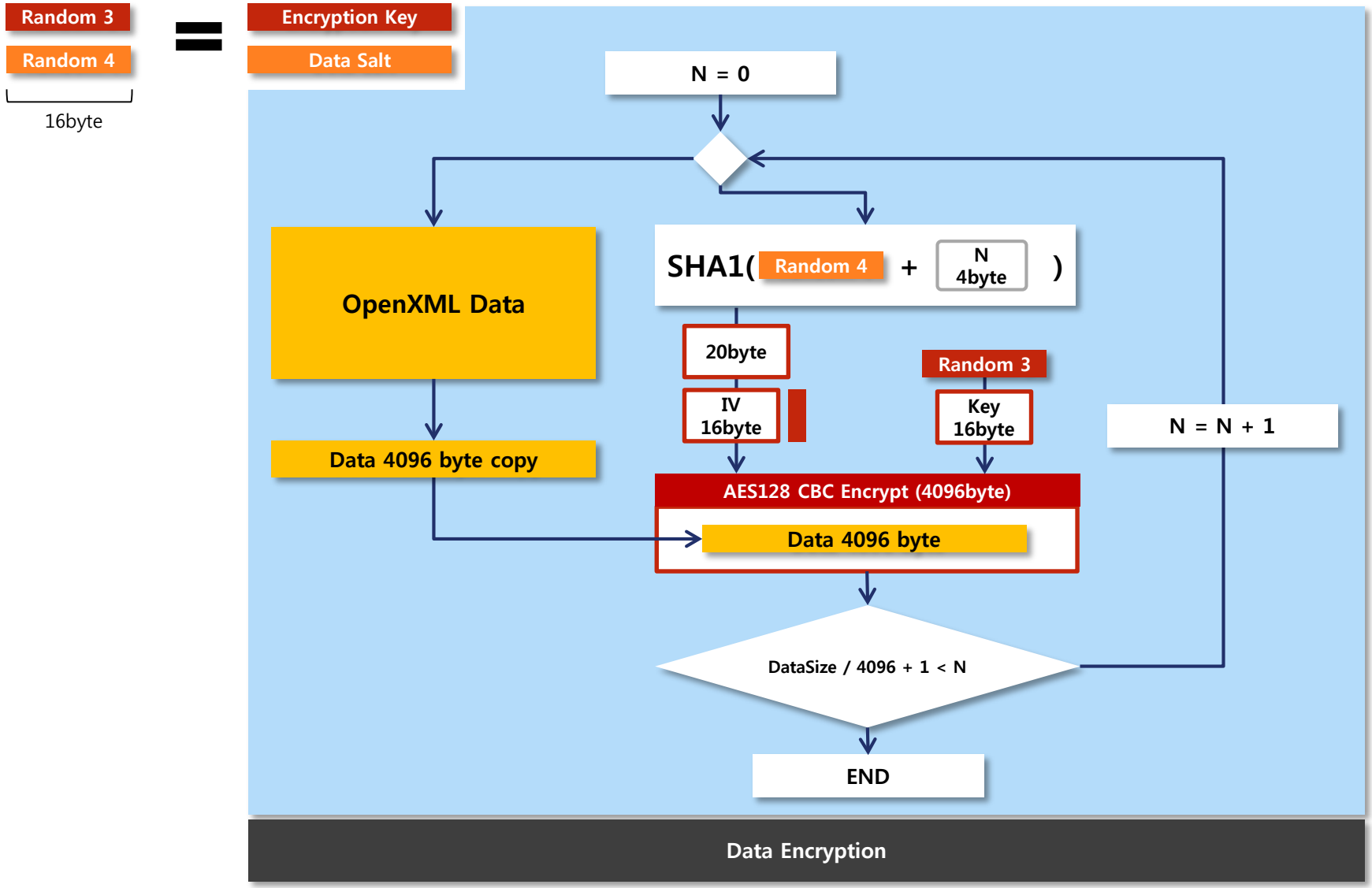

Random 1
Random 2
Random 3
16byte
Key Encrypt Salt
Verifier
Encryption Key
SHA1(Random 1 + Password)
result 20byte
I = 0, N = 1
SHA1(I 4byte + SHA1(N-1) 20byte)
N = N + 1
I = I + 1
I < 100,000
Y
N
Start Header
HFinal 20byte
SHA1(HFinal 20byte + FE A7 D2 76 3B 4B 9E 79)
20byte
Key 16byte
Random 1
IV 16byte
AES128 CBC Encrypt (16byte)
Random 2
Encrypted Verifier Hash Input
SHA1(HFinal 20byte + D7 AA 0F 6D 30 61 34 4E)
20byte
Key 16byte
Random 1
IV 16byte
AES128 CBC Encrypt(32byte)
SHA1(Random 2) + 00 X 12
Encrypted Verifier Hash Value
SHA1(HFinal 20byte + 14 6E 0B E7 AB AC D0 D6)
20byte
Key 16byte
Random 1
IV 16byte
AES128 CBC Encrypt (16byte)
Random 3
Encrypted Key Value

암호키 암호화

- 암호키 저장 위치 재확인

```
<keyEncryptors>
        <keyEncryptor uri="http://schemas.microsoft.com/office/2006/keyEncryptor/password">
        <p:encryptedKey
                spinCount="100000"
                saltSize="16"
                blockSize="16"
                keyBits="128"
                hashSize="20"
                cipherAlgorithm="AES"
                cipherChaining="ChainingModeCBC"
                hashAlgorithm="SHA1"
                saltValue="T5JIFF6I8mn19u4QE7VnLw=="
                encryptedVerifierHashInput="zYaDErh6vXBApsxKCI2M6Q=="
                encryptedVerifierHashValue="6U9p0H4TJR1bRi1p9H7hRYYvRnnrgC4OY7X1Scxzezw="
                encryptedKeyValue="n4CjMfCHt+y7zkbZz9qd2A=="
        />
        </keyEncryptor>
</keyEncryptors>
```

데이터 암호화

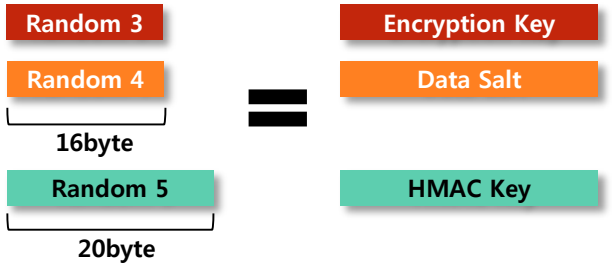

Random 3
Random 4
16byte
=
Encryption Key
Data Salt
N = 0
OpenXML Data
SHA1(Random 4 + N 4byte)
20byte
IV 16byte
Random 3
Key 16byte
N = N + 1
Data 4096 byte copy
AES128 CBC Encrypt (4096byte)
Data 4096 byte
DataSize / 4096 + 1 < N
END
Data Encryption

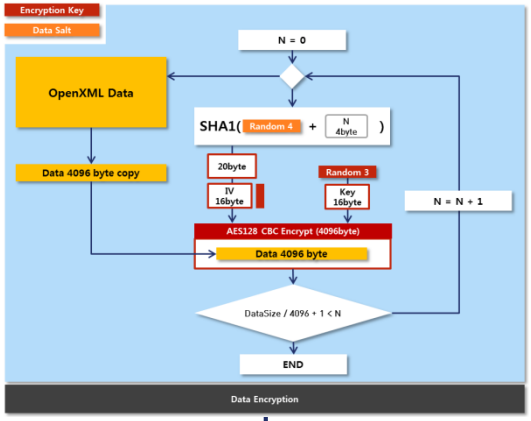
데이터 암호화

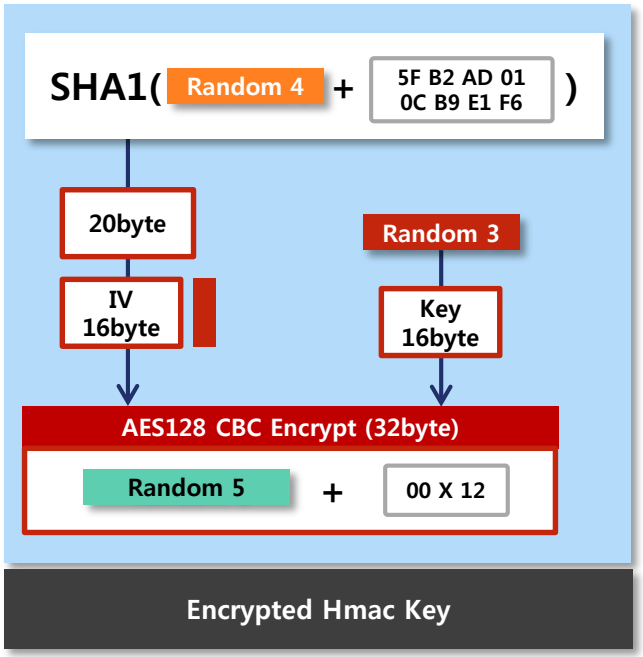
- Data Salt 저장 위치

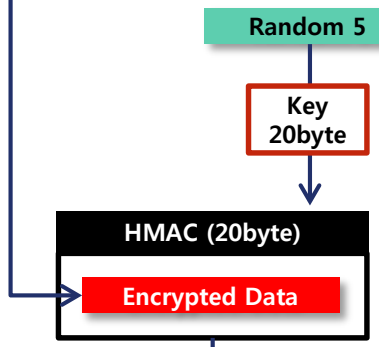
```
<keyData
        saltSize="16"
        blockSize="16"
        keyBits="128"
        hashSize="20"
        cipherAlgorithm="AES"
        cipherChaining="ChainingModeCBC"
        hashAlgorithm="SHA1"
        saltValue="C/FTrY4CWp9maPuveR/fpg=="
/>
```

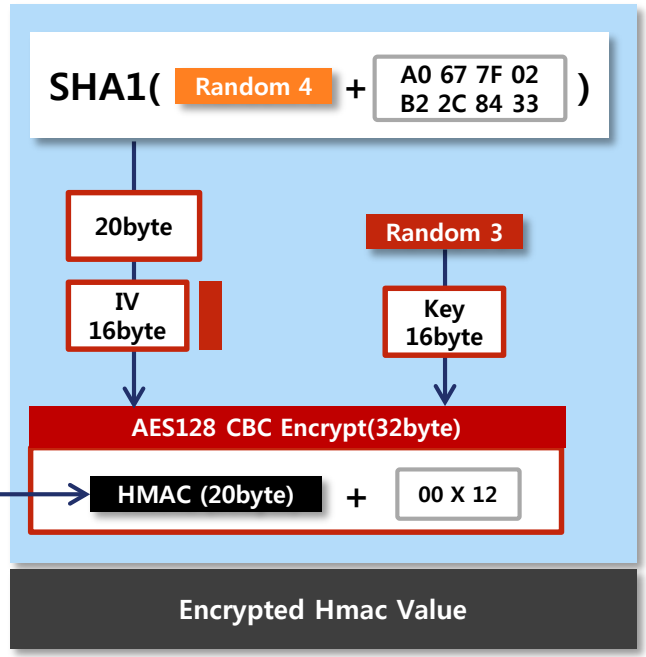
무결성 정보 암호화


Random 3
Random 4
16byte
Random 5
20byte
Encryption Key
Data Salt
HMAC Key


Encryption Key
Data Salt
N = 0
OpenXML Data
SHA1(Random 4 + N 4byte)
Data 4096 byte copy
20byte
IV 16byte
Random 3
Key 16byte
N = N + 1
AES128 CBC Encrypt (4096byte)
Data 4096 byte
DataSize / 4096 + 1 < N
END
Data Encryption


SHA1(Random 4 + 5F B2 AD 01 0C B9 E1 F6)
20byte
IV 16byte
Random 3
Key 16byte
AES128 CBC Encrypt (32byte)
Random 5 + 00 X 12
Encrypted Hmac Key


Random 5
Key 20byte
HMAC (20byte)
Encrypted Data


SHA1(Random 4 + A0 67 7F 02 B2 2C 84 33)
20byte
IV 16byte
Random 3
Key 16byte
AES128 CBC Encrypt(32byte)
HMAC (20byte) + 00 X 12
Encrypted Hmac Value

무결성 정보 암호화

- 무결성 정보 저장 위치

```
<dataIntegrity
        encryptedHmacKey="eKZOmd26cIaelecW0ccn/I7T49I7/7WLyqBkSepFge8="
        encryptedHmacValue="oFYyMfDjJ3Y5SZW4FC+4IXnIZOPWWQR5dv0b41pwrOQ="
/>
```

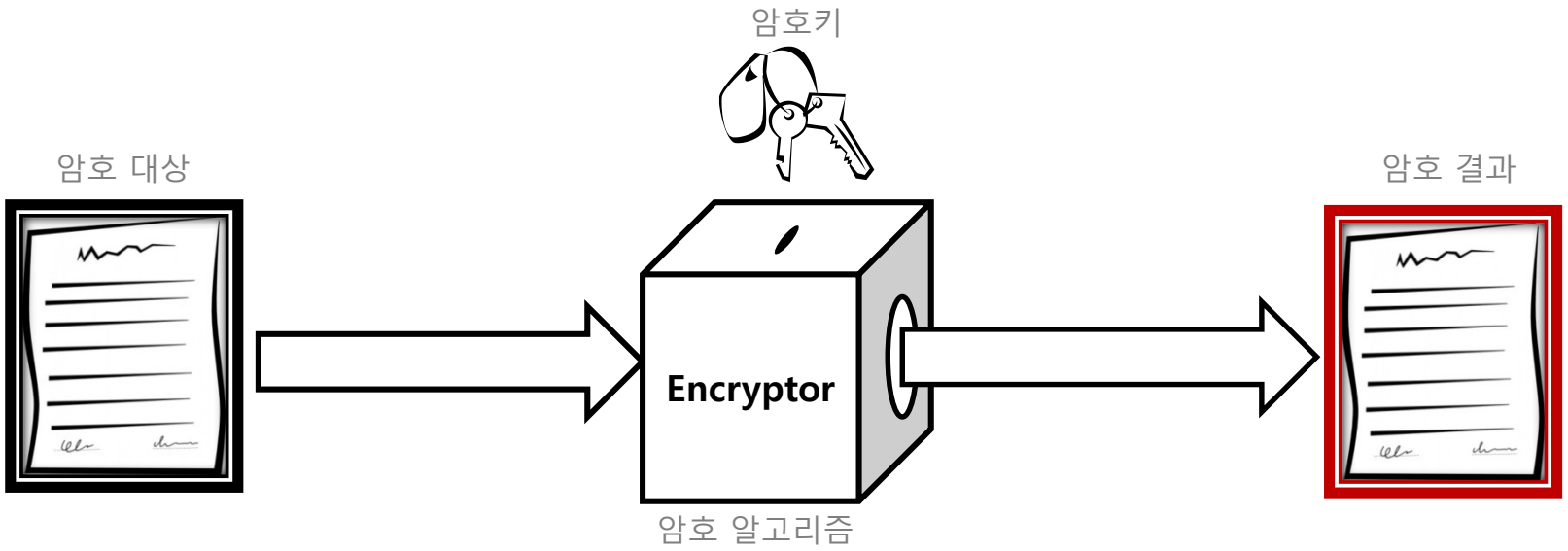
분석과정

- 분석 과정 시작
- IDA Python 기반 분석
- 리버싱 결과 증명

분석과정

분석 목표

- 암호 대상 : 무엇을 암호화 하는가?
- 암호 알고리즘 : 어떤 알고리즘을 사용하는가?
- 암호 키 : 어떤 암호 키를 사용하는가?
- 암호 결과 : 암호 결과가 어떻게 저장되는가?



분석과정

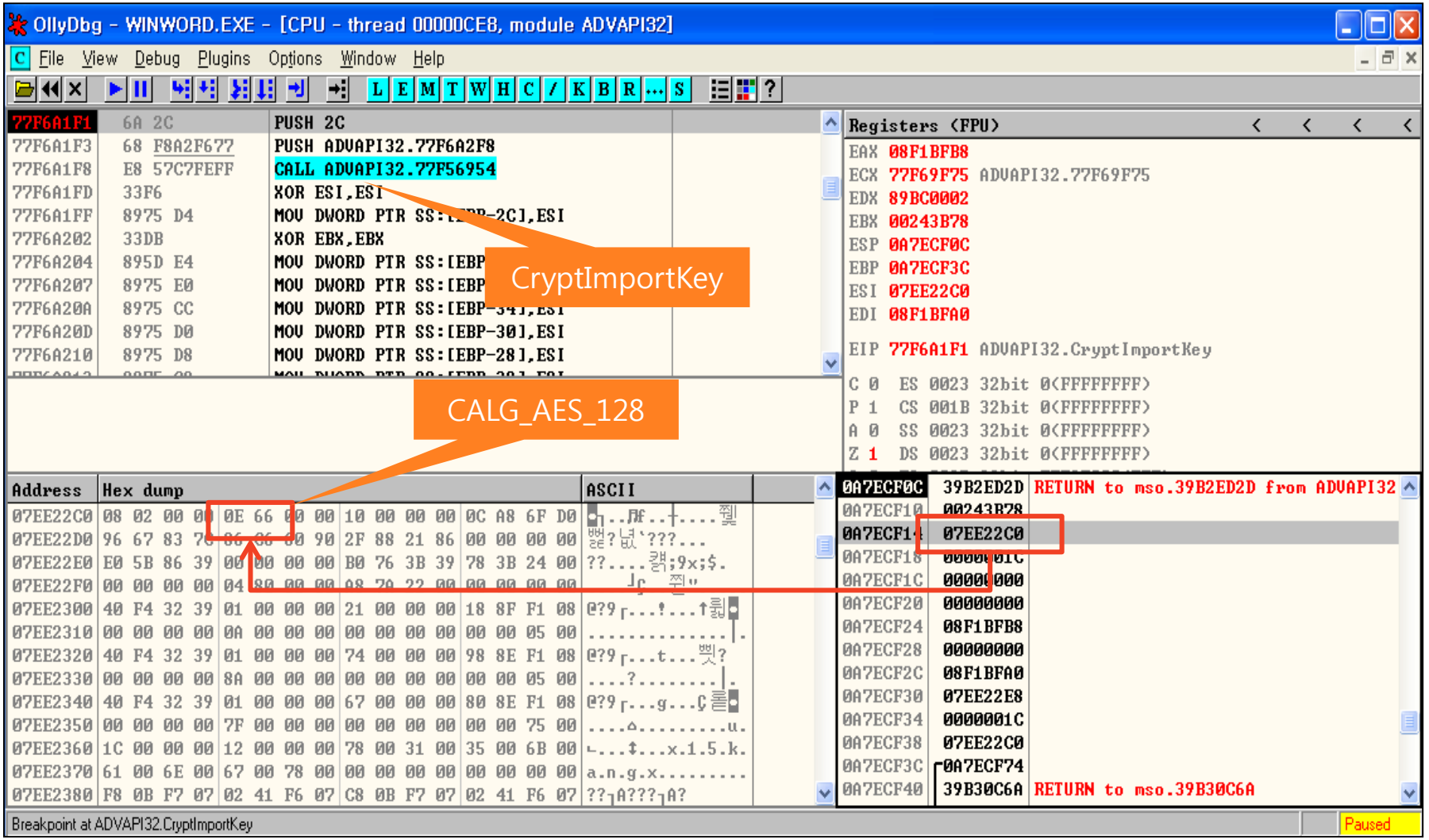
일반적인 CryptAPI 함수 호출 순서

1	CSP(Cryptography Service Provider) 핸들 생성
	CryptAcquireContext(**&hCryptProv**, 0, MS_ENHANCED_PROV, PROV_RSA_FULL, 0)
2	Hash Object 생성
	CryptCreateHash(**hCryptProv**, CALG_MD5, 0, 0, **&hHash**)
3	Password Hash
	CryptHashData(**hHash**, (BYTE *)**szPassword**, strlen(szPassword), 0)
4	Hash 값으로 세션 키 생성
	CryptDeriveKey(hCryptProv, ENCRYPT_ALGORITHM, **hHash**, KEYLENGTH, **&hKey**)
5	pbBuffer의 내용 암호화
	CryptEncrypt(**hKey**, 0, feof(hSource), 0, **pbBuffer**, &dwCount, dwBufferLen)

분석과정

암호 알고리즘 확인

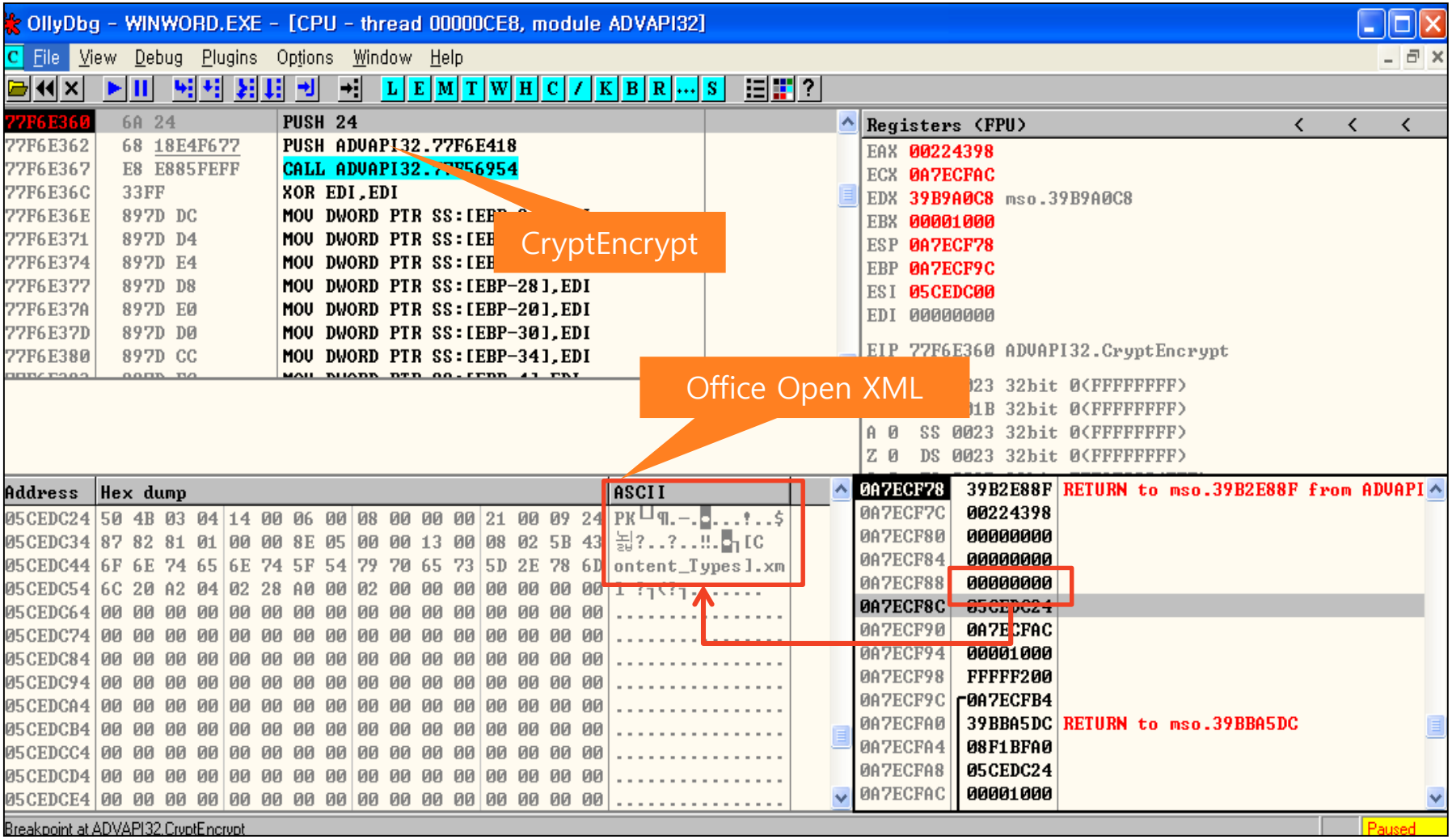
- 확인 결과 : AES128



분석과정

암호화 대상 데이터 확인

- 확인 결과 : Office Open XML 형식



분석과정

Ollydbg 동적 분석의 한계

```
39B313AB  8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
39B313AE  8B08          MOV ECX,DWORD PTR DS:[EAX]
39B313B0  50            PUSH EAX
39B313B1  FF51 08       CALL DWORD PTR DS:[ECX+8]      CryptCreateHash
39B313B4  8BF8          MOV EDI,EAX                    CryptDestroyHash
39B313B6  85FF          TEST EDI,EDI
39B313B8  7C D9         JL SHORT mso.39B31393
39B313BA  8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
39B313BD  8B08          MOV ECX,DWORD PTR DS:[EAX]
39B313BF  6A 04         PUSH 4
39B313C1  8D55 08       LEA EDX,DWORD PTR SS:[EBP+8]
39B313C4  52            PUSH EDX
39B313C5  50            PUSH EAX
39B313C6  FF51 10       CALL DWORD PTR DS:[ECX+10]     CryptHashData
39B313C9  8BF8          MOV EDI,EAX
39B313CB  85FF          TEST EDI,EDI
39B313CD  7C C4         JL SHORT mso.39B31393
39B313CF  FF75 F4       PUSH DWORD PTR SS:[EBP-C]
39B313D2  8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
39B313D5  8B08          MOV ECX,DWORD PTR DS:[EAX]
39B313D7  53            PUSH EBX
39B313D8  50            PUSH EAX
39B313D9  FF51 10       CALL DWORD PTR DS:[ECX+10]     CryptHashData
39B313DC  8BF8          MOV EDI,EAX
39B313DE  85FF          TEST EDI,EDI
39B313E0  7C B1         JL SHORT mso.39B31393
39B313E2  FF75 F4       PUSH DWORD PTR SS:[EBP-C]
39B313E5  8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
39B313E8  8B08          MOV ECX,DWORD PTR DS:[EAX]
39B313EA  53            PUSH EBX
39B313EB  50            PUSH EAX
39B313EC  FF51 14       CALL DWORD PTR DS:[ECX+14]     CryptGetHashPram
39B313EF  8BF8          MOV EDI,EAX
39B313F1  85FF          TEST EDI,EDI
39B313F3  7C 9E         JL SHORT mso.39B31393
39B313F5  FF45 08       INC DWORD PTR SS:[EBP+8]
39B313F8  8B46 04       MOV EAX,DWORD PTR DS:[ESI+4]
39B313FB  8B4D 08       MOV ECX,DWORD PTR SS:[EBP+8]
39B313FE  3B48 08       CMP ECX,DWORD PTR DS:[EAX+8]   If ecx < 100000
39B31401  72 A8         JB SHORT mso.39B313AB
```

동적으로 맵핑 되는 함수주소

매번 변경되는 Random 값...

F8과 F9의 거리...

1회 분석 시 약 2시간 소요

여긴 어디... 나는 누구...?

실 패

분석과정

- F8과 F9의 결과물

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
CryptGenRandom	CryptCreateHash			CryptGetHashParam		CryptImportKey			CryptHashData		CryptDestroyHash	cryptdestroykey	cryptEncrypt		cryptdecrypt	
									00 18 2b c8	ED 74 CD 05 BF C6 E2 25 57 7A 82 78 6F B5 04 01 91 84 88 02						
									00 18 2b c8	D7 AA 0F 6D 30 61 34 4E						
				00 18 2b c8	14 00 00 00											
				00 18 2b c8	5D 6D 43 9C 2E 45 56 C8 53 61 3E 58 A1 07 F2 C8 31 F0 AF 87											
												00 23 99 f0				
						00 19 4d d8	08 02 00 00 0E 66 00 00 10 00 00 00 5D 6D 43 9C 2E 45	f0 99 23 00								
											00 18 2b c8					
														B2 C7 1B 50 0E 0F 65 97 2F 78 35 51 23 09 79		
													00 23 99 f0	36 79 AD FC 34 00 00 00 00 00 00 00 00 00 00 00 00		
														1C F8 40 B4 95 D0 7E ED 9A 9		
											09 5e d9 20					
	00 19 4d d8	8004	20 d9 5e 09													
									09 5e d9 20	ED 74 CD 05 BF C6 E2 25 57 7A 82 78 6F B5 04 01 91 84 88 02						
										14 6E 0B E7 AB AC D0 D6							
				09 5e d9 20	14 00 00 00											
				09 5e d9 20	A6 AE 75 95 5E 35 87 FB A6 58 4F 18 4B B9 BB D5 6F 2F C8 45											
												00 23 99 f0				
						00 19 4d d8	08 02 00 00 0E 66 00 00 10	f0 99 23 00								
											09 5e d9 20					
													00 23 99 f0	36 63 26 19 D8 21 B7 89 1D		
														52 07 1A 48 3A 13 08 D1 89		
	09 60 08 e8	8004	20 d9 5e 09													
									09 5e d9 20	ED 74 CD 05 BF C6 E2 25 57 7A 82 78 6F B5 04 01 91 84 88 02						
									09 5e d9 20	14 6E 0B E7 AB AC D0 D6						
				09 5e d9 20	14 00 00 00											
				09 5e d9 20	A6 AE 75 95 5E 35 87 FB A6 58 4F 18 4B B9 BB D5 6F 2F C8 45											
						09 60 08 e8	08 02 00 00 0E 66 00 00 10	c8 2b 18 00								
											09 5e d9 20					
	00 1d 44 c0	8004	20 d9 5e 09													
									09 5e d9 20	83 BC 4C F3 65 09 A5 33 F4 DA 54 30 A1 AC 3D 69						
									09 5e d9 20	00 00 00 00						
												00 23 1a d0				
						00 1d 44 c0	08 02 00 00 0E 66 00 00 10	d0 1a 23 00								
				09 5e d9 20	49 2C 92 22 F7 BA 3E FB 87 8A 9D C9 26 BD B6 74 DE 64 01 9C											
											09 5e d9 20					
	00 1d 44 c0	8004	20 d9 5e 09						09 5e d9 20	83 BC 4C F3 65 09 A5 33 F4 DA 54 30 A1 AC 3D 69						
									09 5e d9 20	00 00 00 00						
												00 23 1a d0				
						00 1d 44 c0	08 02 00 00 0E 66 00 00 10 00 00 00 36 63 26 19 D8 21 B7	d0 1a 23 00								

분석과정 – IDA Python

- IDA Python 중 Break Point Hook 기능의 기본 구조

1. MyDbgHook 설정
2. Break Point 설정
3. Event 처리 코드 작성

- 끝 -

```
from idaapi import *
from idc import *
import struct

class MyDbgHook(DBG_Hooks):
```

Event 처리 코드 작성 부분

```
        return 0

# Remove an existing debug hook
try:
    if debughook:
        print("----< Removing previous hook ... >----")
        debughook.unhook()
    else:
        print "----< MS Office 2010 Hook >----"
except:
    pass

# Install the debug hook
debughook = MyDbgHook()
debughook.hook()

#CryptGenRandom(hProv, dwLen, *pbBuffer)
AddBpt( LocByName( "advapi32_CryptGenRandom" ) )

#CryptHashData
AddBpt( LocByName( "advapi32_CryptHashData" ) )
```

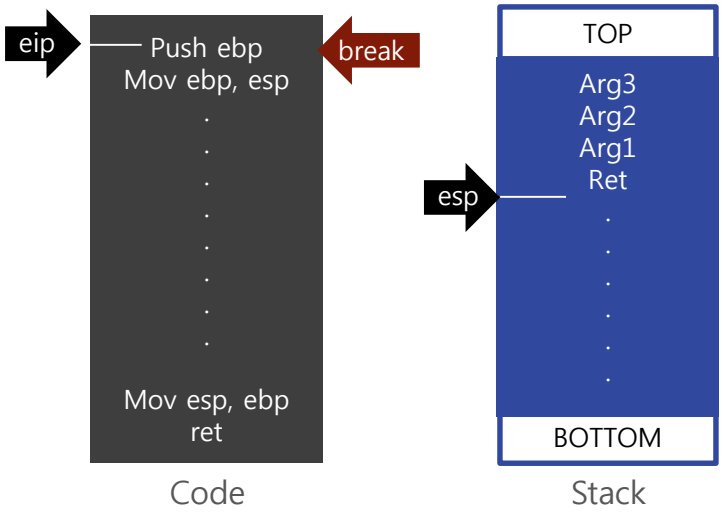
분석과정 – IDA Python

2가지 함수 종류

- 입력 값 확인이 필요한 함수 [ex : func(a, b, c)]
- 결과 값 확인이 필요한 함수 [ex : func(a, **&b**, c)]

입력 값 확인이 필요한 함수

- 함수 시작 점에 브레이크 포인트 설정
- esp를 기준으로 입력 값에 접근하여 확인 후 재실행



입력 값 확인이 필요한 함수

```
#CryptHashData(hHash, *pbData, dwDataLen, dwFlags)
elif eip == LocByName( "advapi32_CryptHashData" ):
    esp = cpu.Esp
    arg1 = esp + 4
    arg2 = esp + 8
    arg3 = esp + 12
    arg4 = esp + 16
    print "CryptHashData( 0x%08x, " % self.read_addr(arg1),
    print "0x%08x, " % self.read_addr(arg2),
    print "0x%08x, " % self.read_addr(arg3),
    print "0x%08x) " % self.read_addr(arg4)

    print "[0x%08x] " % self.read_addr(arg2),
    tmp = dbg_read_memory(self.read_addr(arg2), self.read_addr
    #for i in tmp:
    #    print "%02x" % ord(i),
    print tmp.encode('hex')
    print "\n"

    ResumeProcess()
```

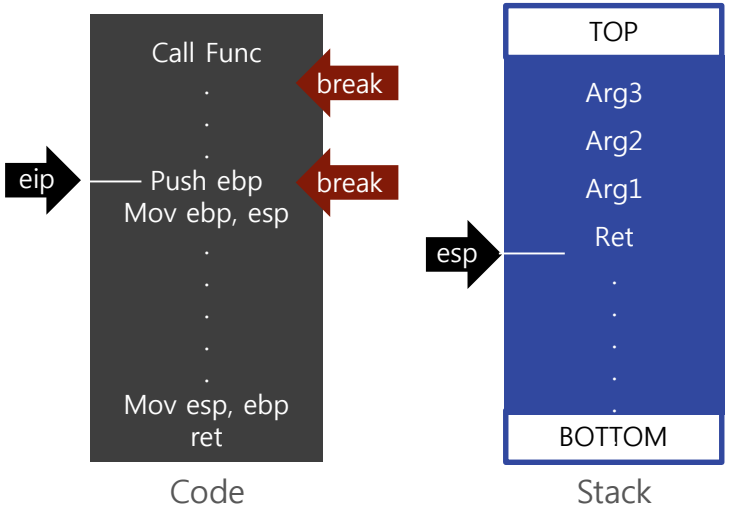
분석과정 – IDA Python

2가지 함수 종류

- 입력 값 확인이 필요한 함수 [ex : func(a, b, c)]
- 결과 값 확인이 필요한 함수 [ex : func(a, **&b**, c)]

결과 값이 필요한 함수

- 함수 시작 점에 브레이크 포인트 설정
- esp를 기준으로 결과 값이 저장될 메모리 주소 확인
- Ret에 브레이크 포인트 설정 후 재실행
- 결과 값 확인 후 브레이크 포인트 해제 후 재실행

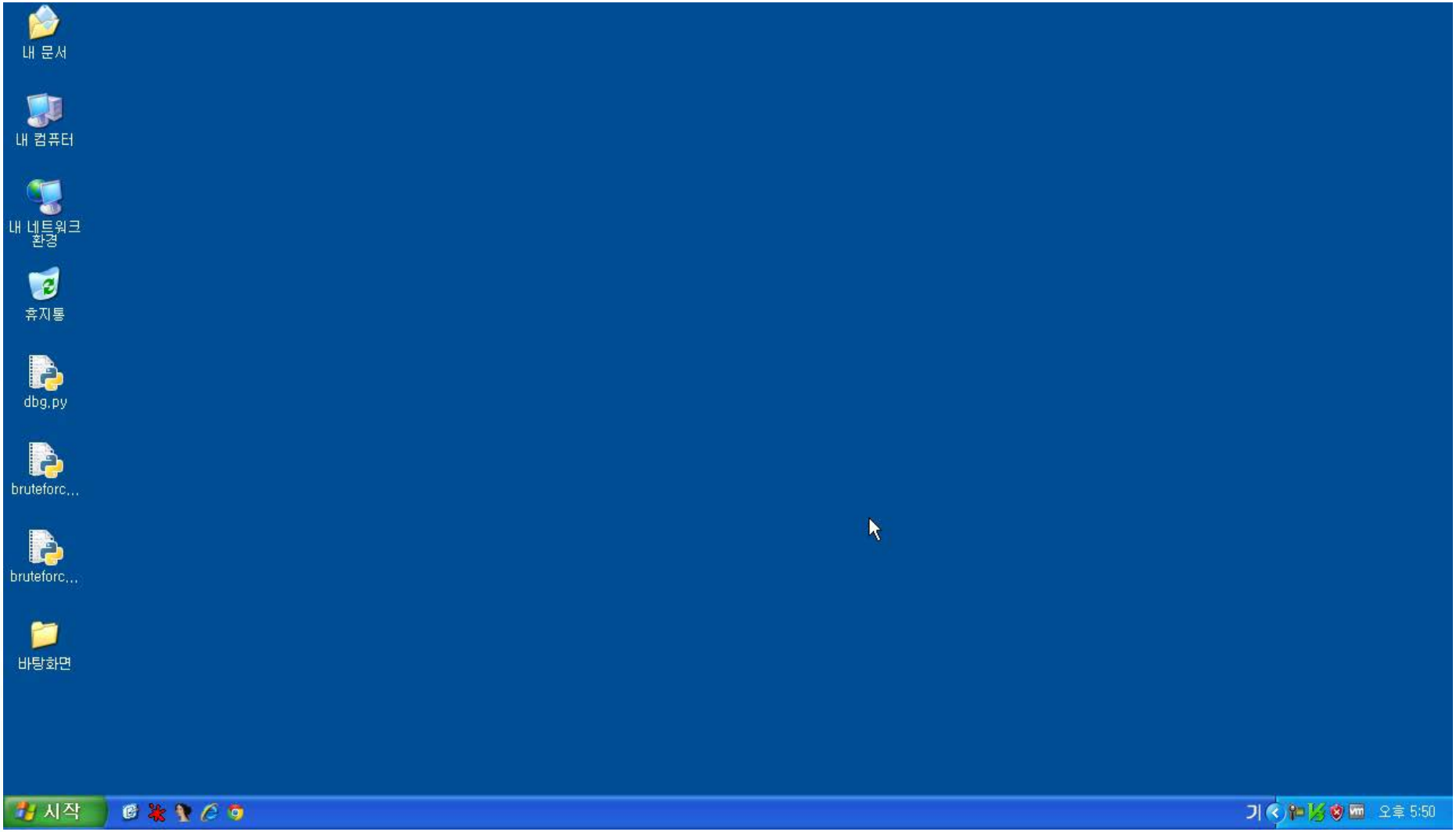


결과 값이 필요한 함수

```
#CryptGenRandom(hProv, dwLen, *pbBuffer)
elif eip == LocByName( "advapi32_CryptGenRandom" ):
    esp = cpu.Esp
    arg1 = esp + 4
    arg2 = esp + 8
    arg3 = esp + 12
    print "CryptGenRandom( 0x%08x, " % self.read_addr(arg1),
    print "0x%08x, " % self.read_addr(arg2),
    print "0x%08x) " % self.read_addr(arg3)
    self.CryptGenRandomRet = self.read_addr(esp)
    self.CryptGenRandomSize = self.read_addr(arg2)
    self.CryptGenRandomValue = self.read_addr(arg3)
    AddBpt(self.CryptGenRandomRet)
    ResumeProcess()
#CryptGenRandom Result
elif eip == self.CryptGenRandomRet:
    tmp = dbg_read_memory(self.CryptGenRandomValue, self.Crypt
    print "[0x%08x]" % self.CryptGenRandomValue,
    for i in tmp:
        print "%02x" % ord(i),
    print "\n"
    DelBpt(self.CryptGenRandomRet)
    ResumeProcess()
```

분석과정 – IDA Python

- IDA Python 스크립트를 활용한 디버깅



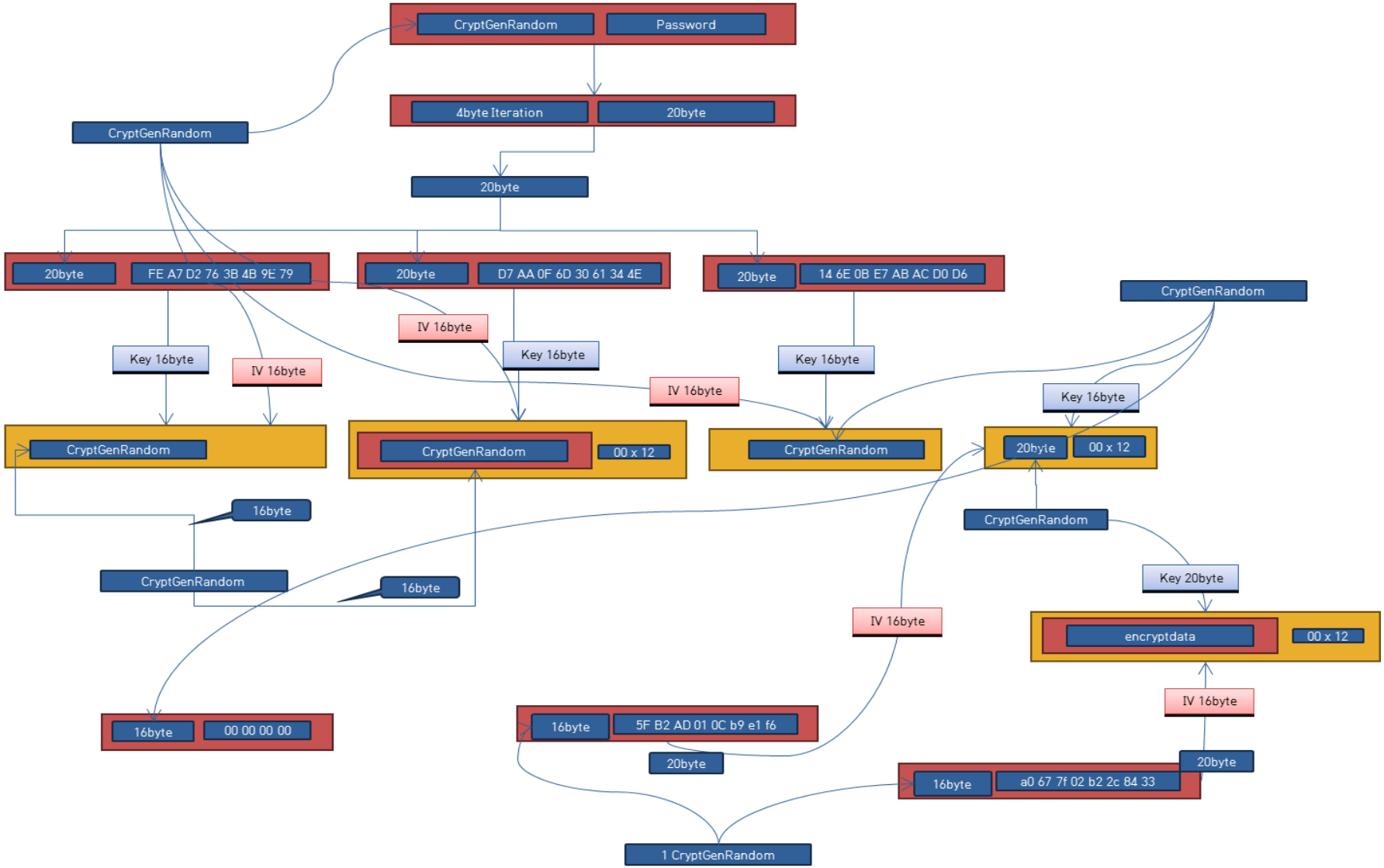
분석과정 – IDA Python

IDA Python 로그 결과 확인

```
[0x065c8c24] 01 10 00 01 48 2b f4 01 10 00 01 10 00 01 10 00 01 10 50 8c 00 a4 95 62 28 61 08 04 40 00 04 40 00 04

[0x065c8c24]Exception in DBG Hook function:

Traceback (most recent call last):
  File "C:/Documents and Settings/Owner/???? ????/dbg_final.py", line 106, in dbg_bpt
    for i in tmp:
TypeError: 'NoneType' object is not iterable
CryptCreateHash( 0x0a0a9f20,  0x00008004,  0x00000000,  0x00000000,  0x00124c94 )
[0x00124c94] 0x0022cc80

CryptHashData( 0x0022cc80,  0x08e31e20,  0x00000010,  0x00000000)
[0x08e31e20]  43510e97454b5069614e6c78da2d2ed4


CryptHashData( 0x0022cc80,  0x00124d38,  0x00000004,  0x00000000)
[0x00124d38]  0b000000


CryptDestroyKey( 0x0a0a45b8 )

CryptImportKey( 0x0a0a9f20,  0x09c00420,  0x0000001c,  0x00000000,  0x00000000,  0x088641f8 )
[0x09c00420] 08 02 00 00 0e 66 00 00 10 00 00 00 13 99 bf a3 ea f9 e1 6a 01 aa 1e af ba 02 3a 68
[0x088641f8] 0x0a0a45b8

CryptGetKeyParam( 0x0a0a45b8,  0x00000008,  0x00124cf8,  0x00124ce4 )
[0x00124cf8] 80 00 00 00

CryptSetKeyParam( 0x0a0a45b8,  0x00000004,  0x00124ce8,  0x00000000 )
[0x00124ce8] 01 00 00 00

CryptGetHashParam( 0x0022cc80,  0x00000002,  0x00c10000,  0x00124cb8,  0x00000000 )
[0x00c10000] ba 18 68 ef 4b 07 02 7f ce 7d ef cc 2f 2c 45 b2 bc 5b 63 4f

CryptSetKeyParam( 0x0a0a45b8,  0x00000001,  0x00c10000,  0x00000000 )
[0x00c10000] ba 18 68 ef 4b 07 02 7f ce 7d ef cc 2f 2c 45 b2 bc 5b 63 4f 52 00 50 00 52 00 4f 00 46 00 49 00
```

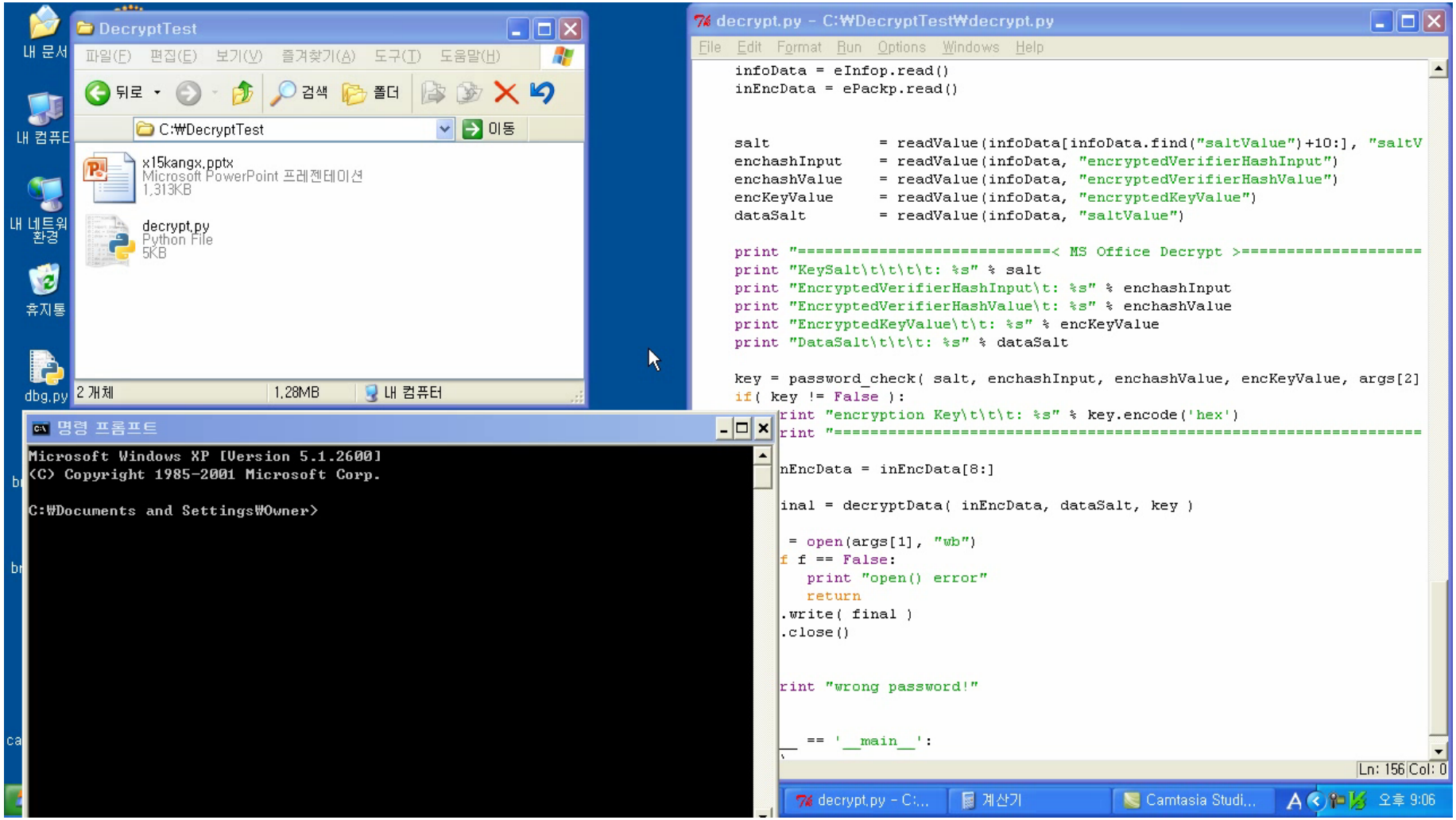
분석과정

- Python 파이썬 로그 결과 도식화



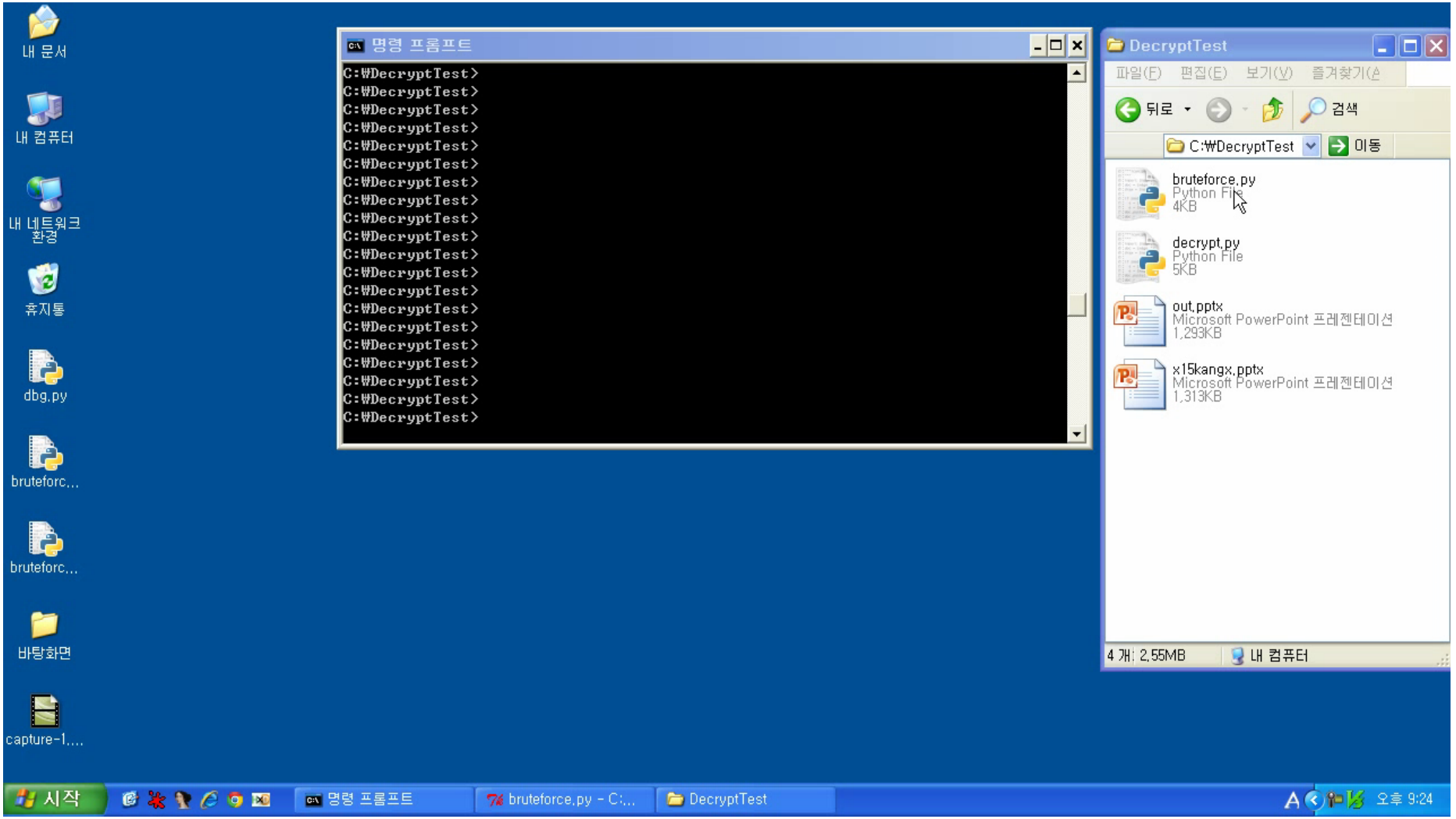
분석과정

- Python으로 구현한 오피스 암호 해제 시연



분석과정

- Python으로 구현한 암호화된 오피스파일 Bruteforce 시연



MS office 2010 vs MS office 2013

MS Office 2010 vs MS Office 2013

MS Office 2013 XML 블록 내용

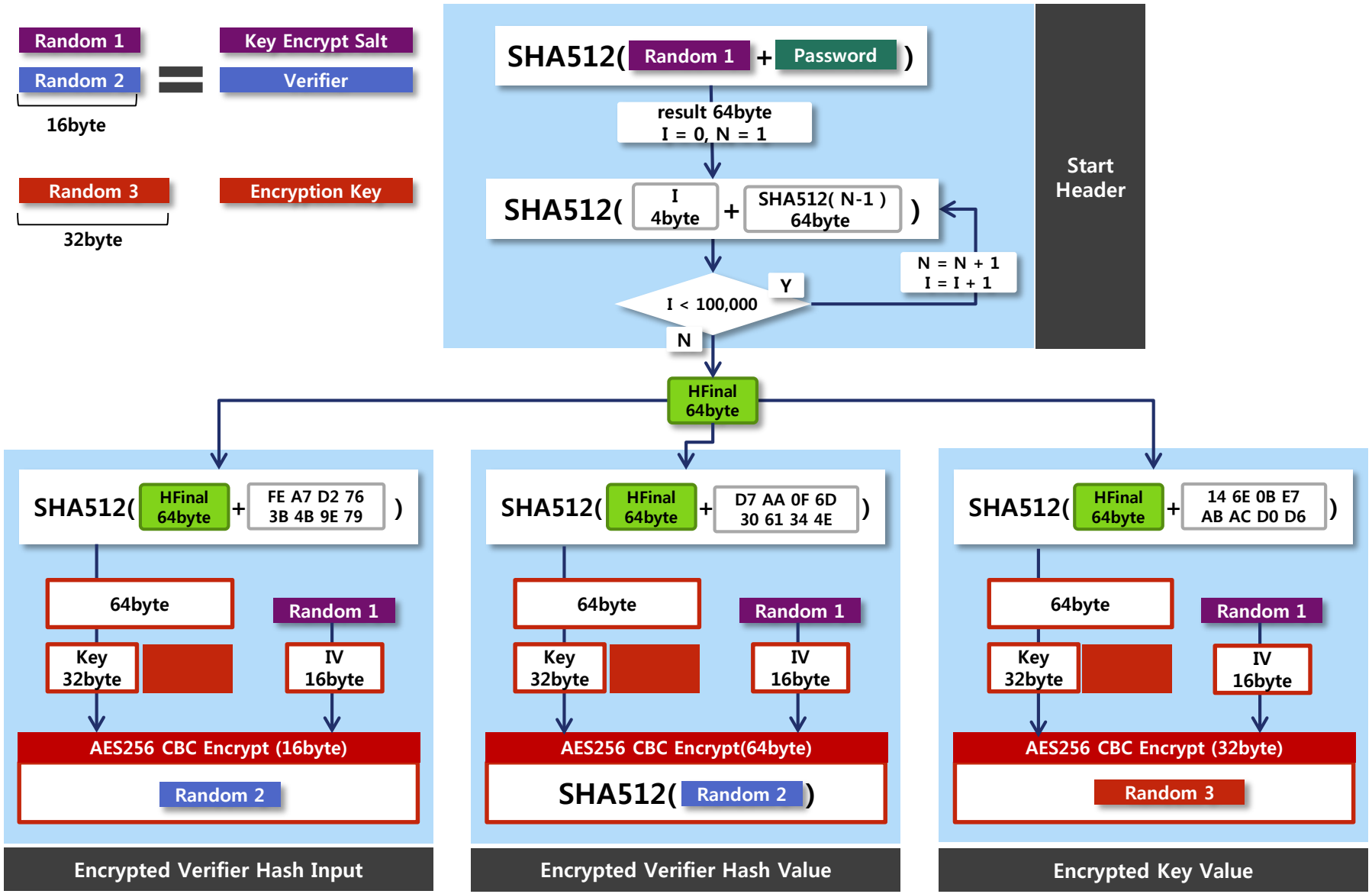
```
<keyData
        saltSize="16"
        blockSize="16"
        keyBits="256"
        hashSize="64"
        cipherAlgorithm="AES"
        cipherChaining="ChainingModeCBC"
        hashAlgorithm="SHA512"
        saltValue="Lui0tyCEde7DQJ7E54wHzoQ=="
/>


<dataIntegrity
        encryptedHmacKey="a734SEI851x3M/baDOYKwuQVhxZJqmMb3QQgUIjnKRbpmLZvLfIw6aknH7j/SBMJSJjKUpHLtidUK/wHZABwiQ=="
        encryptedHmacValue="qqa8/fXErGHWXoVTMXsiu/Dr9ufnZ5ckHBIVscMWR9DOiA92zNC6T4cb90FV1LyuEePn4XeOaiuQpMa6XUinrw=="
/>

<keyEncryptors>
        <keyEncryptor uri="http://schemas.microsoft.com/office/2006/keyEncryptor/password">
        <p:encryptedKey
                spinCount="100000"
                saltSize="16"
                blockSize="16"
                keyBits="256"
                hashSize="64"
                cipherAlgorithm="AES"
                cipherChaining="ChainingModeCBC"
                hashAlgorithm="SHA512"
                saltValue="wEtoAD5kBiMf1GcFXFUxFg=="
                encryptedVerifierHashInput="v87/9fmj75r/v9Rzk7vdyw=="
                encryptedVerifierHashValue="MeBgGtLQhE/bafxcS6u4WlA6cMAeDdlUdgXIUsIDdLBXXvWpQrARuvNJzWoLLp4Zidkuf2rqvU
                encryptedKeyValue="3yLnG9+P4JMoo4/84HRVV8Of1ttxV8ieS7X/IFj2GBo="
        />
        </keyEncryptor>
</keyEncryptors>
```

MS Office 2010 vs MS Office 2013

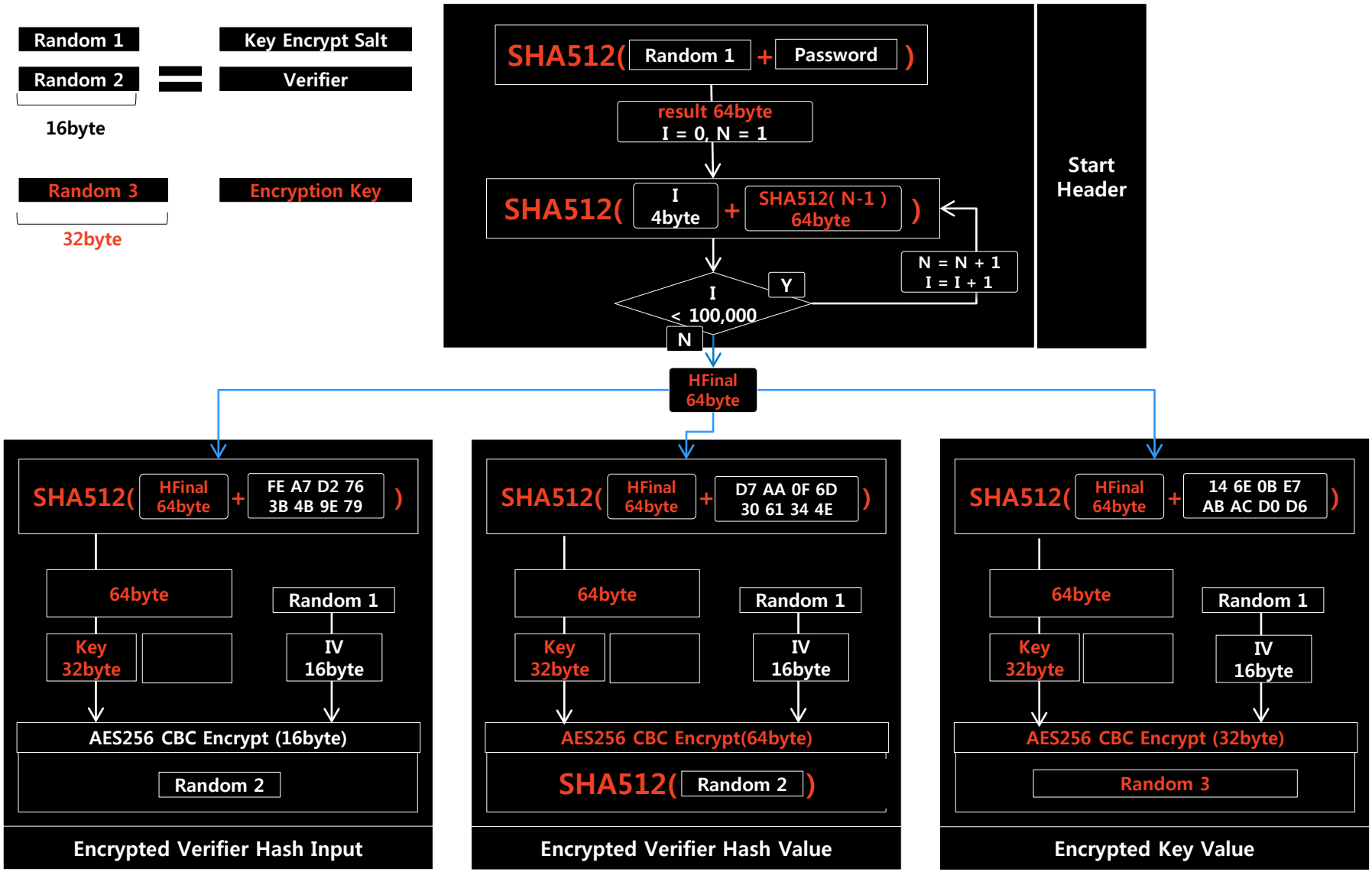
MS Office 2010과 MS Office 2013의 키 길이 및 해쉬 알고리즘 차이

구 분		MS Office 2010	MS Office 2013
KeyData	saltSize	16	16
	bloackSize	16	16
	KeyBits	**128**	**256**
	hashSize	**20**	**64**
	cipherAlgoritm	AES	AES
	cipherChaining	ChainingModeCBC	ChainingModeCBC
	hashAlgoritm	**SHA1**	**SHA512**
	saltValue	16byte	16byte
dataIntegrity	encryptedHmacKey	**32byte**	**64byte**
	encryptedHmacValue	**32byte**	**64byte**
KeyEncryptors	spinCount	100000	100000
	saltSize	16	16
	blockSize	16	16
	keyBits	**128**	**256**
	hashSize	**20**	**64**
	cipherAlgorithm	AES	AES
	cipherChaining	ChainingModeCBC	ChainingModeCBC
	hashAlgorithm	**SHA1**	**SHA512**
	SaltValue	16byte	16byte
	encryptedVerifierHashInput	16byte	16byte
	encryptedVerifierHashValue	**32byte**	**64byte**
	encryptedKeyValue	**16byte**	**32byte**

암호키 암호화(MS Office 2013)



암호키 암호화(MS Office 2010 ⇒ 2013)



MS Office 2013 관련 내용 이하 생략

결론 2-2

결론2-2

- 이번에 발표된 내용을 기반으로 MS Office 관련 연구에 도움이 되길 바랍니다.
  - Ex) 향상된 Bruteforce Tool 개발 등.
- IDA Python을 활용하여 진행하시는 리버싱에 도움이 되길 바랍니다.
- 관련 연구 진행 시 내용 공유 부탁 드립니다.

Q & A

Thanks for Listening

x15kangx@nate.com