



Code Engn

www.CodeEngn.com

2014 CodeEngn Conference 10



숨겨진 베일을 벗겨라

암호 알고리즘 사용의 취약점 nurilab

[강의 내용]

- 디지털 포렌식
- 리버스 엔지니어링
- 암호 알고리즘 사용 사례
- 디지털 포렌식 관점
- 리버스 엔지니어링 해보자
- 어떻게 이렇게 만들었지?

 07.05



디지털 포렌식

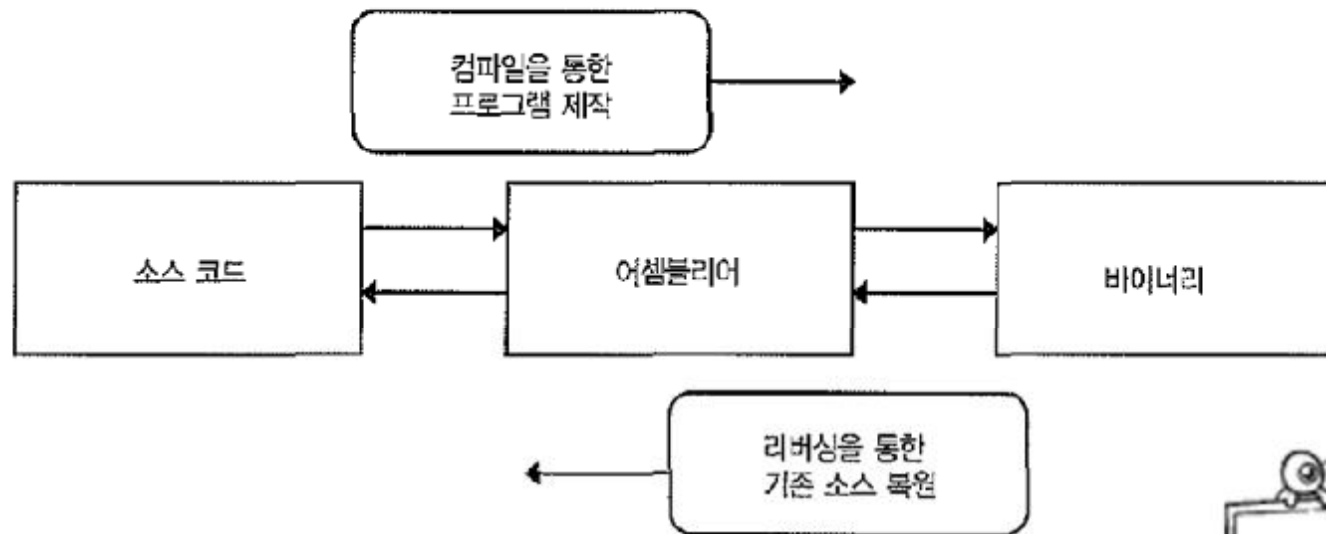
- 디지털 포렌식
 - 특정 사건과 관련된 사람의 디지털 기기등에서 법적 증거를 찾는 행위
- 대표적인 디지털 포렌식
 - 디지털 기기에서 삭제되거나 숨겨진 파일의 복구
 - 사건과 관련된 사람의 디지털 기기 사용 흔적 추적
 - 암호가 설정된 문서의 복구
 - 등등



리버스 엔지니어링 개요

- 리버스 엔지니어링

- 리버스(reverse)와 엔지니어링(engineering) 의 합성어
- 줄여서 리버싱(reversing) 이라고 함
- 거꾸로 분석하는 것을 의미



- 리버싱을 하는 사람

- 리버스엔지니어(reverse engineer) 또는 리버서(reverser)라고도 함

- 크래커 (Cracker)
 - 보통 법을 어기면서까지 자신의 영리만을 위해서 크랙을 만들어서 배포
- 해커 (Hacker)
 - 프로그램을 구현하는 과정에서 실수할 수 있는 부분들을 분석하여 허점을 찾아내 프로그램의 보안이 강화 되기를 비라면서 자신이 찾은 취약점들을 세상에 공개하여 방어 기술의 발전에 공헌



- 법제처 종합법령정보센터 (<http://www.moleg.go.kr>)를 방문
 - 정보 통신과 관련된 법률을 볼 수 있음

The screenshot shows the Korea Legal Information Service (KOLIS) website. The main content area displays the '컴퓨터프로그램 보호법' (Computer Program Protection Act) of 2007. The page includes a search bar, navigation tabs, and a detailed table of contents for the act. The table of contents lists 17 articles, including provisions on the definition of computer programs, the establishment of the Korea Copyright Commission, and the protection of computer programs from unauthorized copying and distribution.

구분	제목	제정·개정·폐지
1.	컴퓨터프로그램 보호법 시행령	[시령 2007.4.5.] [대통령령 제19968호, 2007.4.5.]
2.	컴퓨터프로그램 보호법 시행규칙	[시행 2007.4.5.] [정보통신부령 제214호, 2007.4.5.]
3.	컴퓨터프로그램 보호법	[시령 2007.4.5.] [법률 제8332호, 2006.10.4.]
4.	컴퓨터프로그램 보호법 시행규칙	[시령 2007.4.5.] [정보통신부령 제144호, 2007.4.5.]
5.	컴퓨터프로그램 보호법 시행령	[시령 2003.8.6.] [대통령령 제18263호, 2003.8.6.]
6.	컴퓨터프로그램 보호법	[시령 2003.7.1.] [법률 제8643호, 2002.12.2.]
7.	컴퓨터프로그램 보호법 시행규칙	[시령 2001.7.25.] [정보통신부령 제113호, 2001.7.25.]
8.	컴퓨터프로그램 보호법 시행령	[시령 2001.7.17.] [대통령령 제17308호, 2001.7.17.]
9.	컴퓨터프로그램 보호법	[시령 2001.7.17.] [법률 제6357호, 2001.1.1.]
10.	컴퓨터프로그램 보호법 시행규칙	[시령 2000.8.19.] [정보통신부령 제103호, 2000.8.19.]
11.	컴퓨터프로그램 보호법 시행령	[시령 2000.8.5.] [대통령령 제15942호, 2000.8.5.]
12.	컴퓨터프로그램 보호법	[시령 2000.7.29.] [법률 제6233호, 2000.1.1.]
13.	컴퓨터프로그램 보호법 시행령	[시령 1999.1.1.] [대통령령 제18260호, 1999.1.1.]
14.	컴퓨터프로그램 보호법 시행규칙	[시령 1999.1.1.] [정보통신부령 제59호, 1999.1.1.]
15.	컴퓨터프로그램 보호법	[시령 1999.1.1.] [법률 제5606호, 1998.12.2.]
16.	컴퓨터프로그램 보호법 시행령	[시령 1998.6.7.] [대통령령 제15019호, 1998.6.7.]
17.	컴퓨터프로그램 보호법 시행규칙	[시령 1998.6.7.] [정보통신부령 제59호, 1998.6.7.]

- **컴퓨터 프로그램 보호법**

- 제12조의2항 (프로그램코드역분석)

- ① 정당한 권원에 의하여 프로그램을 사용하는 자 또는 그의 허락을 받은 자가 호환에 필요한 정보를 쉽게 얻을 수 없고 그 획득이 불가피한 경우 당해 프로그램의 호환에 필요한 부분에 한하여 프로그램저작권자의 허락을 받지 아니하고 프로그램코드역분석을 할 수 있다.
 - ②제1항의 규정에 의한 프로그램코드역분석을 통하여 얻은 정보는 다음 각호의 1에 해당하는 경우에는 이를 사용할 수 없다.
 - 1. 호환 목적외의 다른 목적을 위하여 이용하거나 제3자에게 제공하는 경우
 - 2. 프로그램코드역분석의 대상이 되는 프로그램과 표현이 실질적으로 유사한 프로그램을 개발·제작·판매하거나 기타의 프로그램저작권을 침해하는 행위에 이용하는 경우

- **보안 업체가 시행하는 역분석은 법적인 문제가 없는가?**



한글 2.x 암호 체계

- 파워 해킹 테크닉 (p464) 발췌



사례 9. 한글 암호 해독 사건

안기부도 공안사건 해결에 해커를 동원했다는데……

이 사건은 최근 실제로 일어났던 공안 사건의 해결에 결정적 역할을 한 컴퓨터 프로그램의 암호 해독 경위를 놓고 수사당국과 암호체계를 제작한 컴퓨터소프트웨어 업체가 서로 다른 주장을 펼쳐 논란 속에 커다란 관심을 불러 일으킨 사건이었다.

안기부가 북한의 지령을 받은 지하당 조직 ‘구국전위’ 총책인 안재구(61)씨를 검거하고 안씨가 컴퓨터 디스켓에 담아 두었던 북한공작지도부의 대남 지령문과 ‘구국전의’의 대북보고문 전문을 완전 해독, 증거물로 제시한 것은 1994년 6월이었다.

안씨는 문서작성 소프트웨어인 ‘**한글 2.1**’의 암호체계를 믿고 이 프로그램 안에 ‘**장백산**’이라는 암호를 사용, 평문으로 문서를 보관했던 것인데, 논란의 핵심은 바로 이 ‘**한글 2.1**’ 암호체계의 신뢰성 여부였다.

- 파워 해킹 테크닉 (p464) 발췌

당시 안기부 수사 관계자는 ‘인적 정보를 통해 암호를 찾아낸 것이 아니라 민간인 해커들을 대거 동원해 **기계적인 정공법으로 암호를 풀어냈다**’고 했다. 안씨 등 이 사건 관련자로부터 강압적인 수단을 동원해 암호를 알아낸 것이 아니고 이론적으로 컴퓨터 기술을 이용, 풀어냈다는 것이었다.

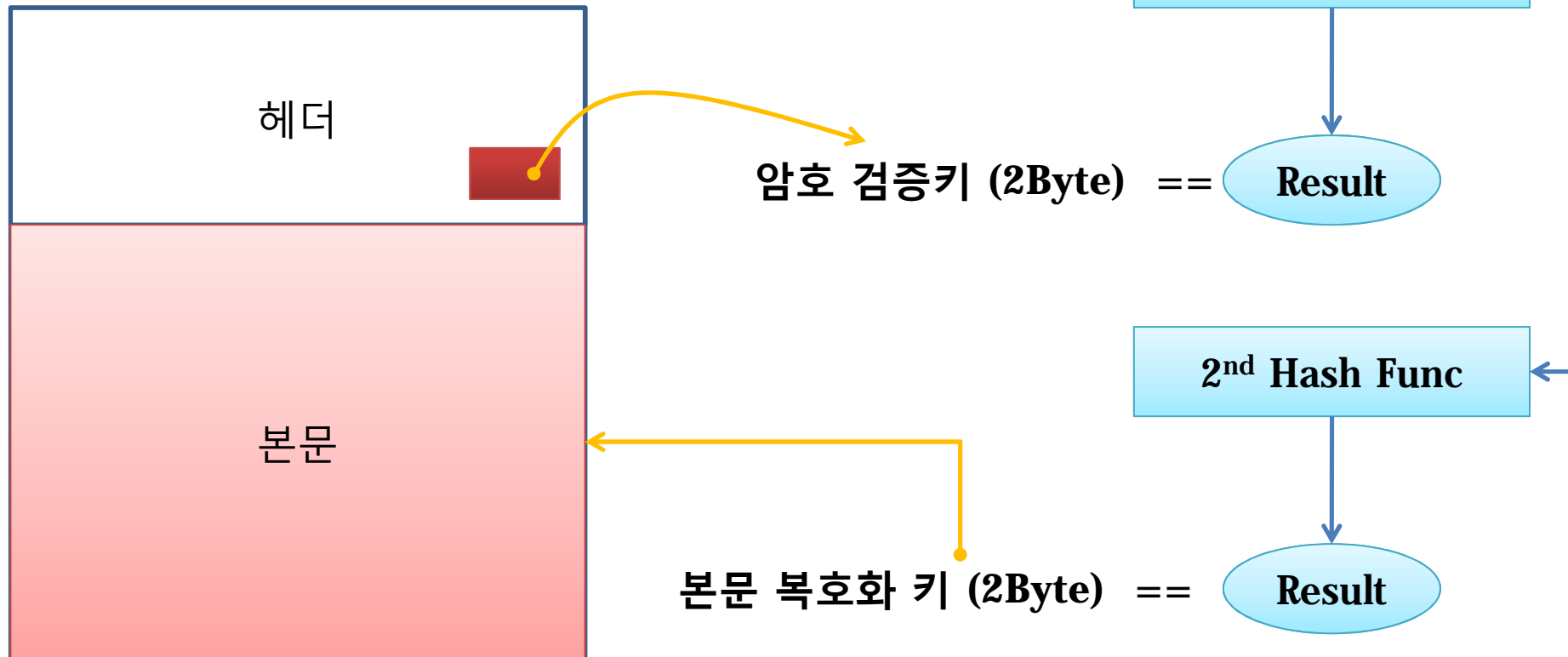
이러한 안기부의 주장을 정면으로 반박하고 나선 것은 대부분의 컴퓨터 사용자들이 쓰고 있는 ‘**한글 2.1**’을 만든 ‘한글과컴퓨터사’였다. 한글과컴퓨터사는 ‘**한글 2.1의 암호체계를 정공법으로 푸는 것은 거의 불가능하다**’고 밝혔다. 암호를 거는 데 42억9천4백96만7천2백95개의 숫자가 조합되어 있기 때문에 1초에 하나씩 암호 키를 확인한다고 해도 133년이 걸린다는 것이 회사측의 주장이었다.



- 한글 2.x 도스용 프로그램



- 한글 헤더
 - 2Byte 암호 검증 키 존재
- 한글 본문
 - 2Byte 본문 복호화 키를 통해 복호화



- 공격 방법의 검토
 - 해시 값을 공격할 것인가?
 - Key Space를 공격할 것인가?

해시 값 공격 범위

- 암호 해독을 위해서는...
 - 암호 검증 키 (2Byte) + 본문 복호화 키 (2Byte) = 4Byte
- 복호화를 위한 경우의 수
 - $2^{32} = 4,294,967,296$

Key Space 공격 범위

- 5글자 ~ 45글자까지 입력 가능
- 키보드 입력 가능 값을 95글자라고 할 경우...
- 5 ~ 10글자의 Key Space = 60,510,648,114,434,700,000

공격하는 것은 무리!!!

• 1st Hash 알고리즘

```

unsigned GetConfirmKey(unsigned InpKey[])
{
    int i = 0;

    _BX = InpKey[i];
    asm xor si, si
    asm xor ax, ax
    asm cmp bx, +00
    asm jnz _1e0f
    exit(0);
_1dfa:
    asm mov ax, si
    asm mov cl, 3
    asm shl ax, cl
    asm push ax
    _BX = InpKey[i];
    asm pop ax
    asm xor ax, bx
    asm add ax, 0a5h
    asm mov si, ax
    asm inc WORD PTR i
_1e0f:
    asm push ax
    _BX = InpKey[i];
    asm pop ax
    asm cmp bx, +0
    asm jnz _1dfa
    asm mov ax, si
    asm mov bx, 0ff00h
    asm xor dx, dx
    asm div bx
    asm add dx, +55h
    asm mov ax, dx

    return(_AX);
}
    
```

• 2nd Hash 알고리즘

```

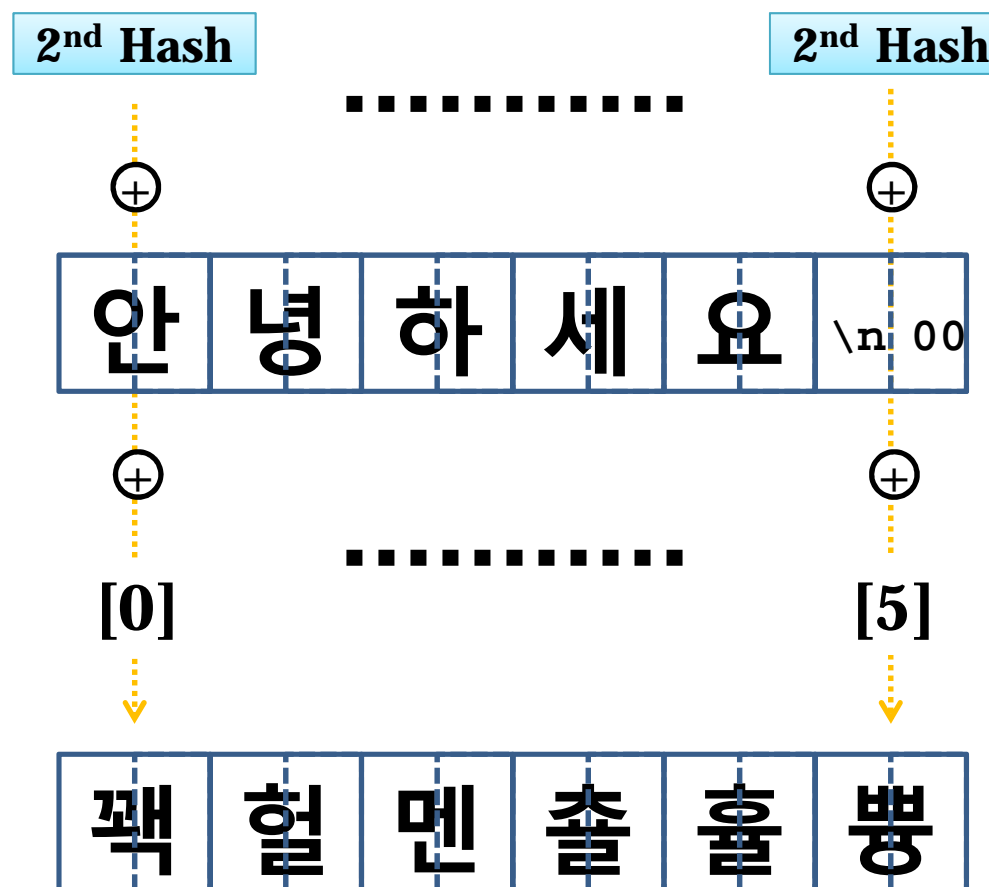
unsigned GetDecodeKey(unsigned InpKey[])
{
    int i = 0;

    _BX = InpKey[i];
    asm xor si, si
    asm xor dx, dx
    asm jmp _0016;
_0007:
    asm mov cx, 3
    asm ror dx, cl
    asm push ax
    _BX = InpKey[i];
    asm pop ax
    asm xor dx, bx
    asm inc WORD PTR i
_0016:
    asm push ax
    _BX = InpKey[i];
    asm pop ax
    asm cmp bx, +00
    asm jnz _0007
    asm mov ax, dx

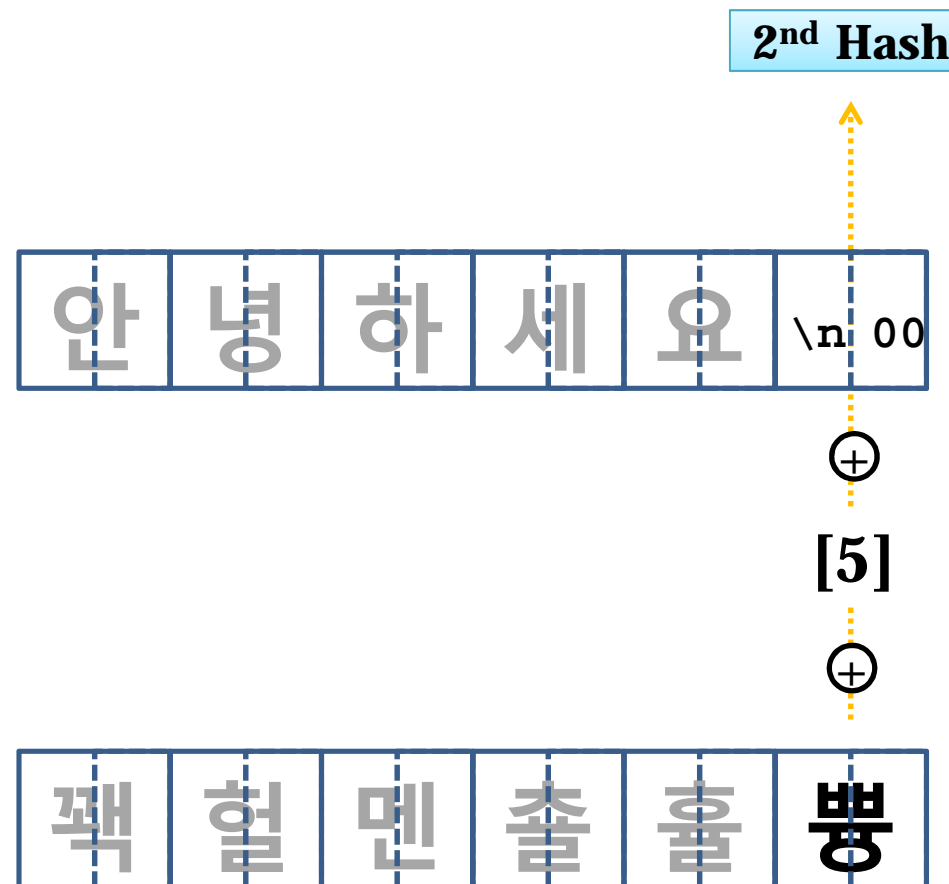
    return (_AX);
}
    
```

출처 : HWP 2.1 형식의 문서파일의 숫자 암호 해독기
(By 임형택)

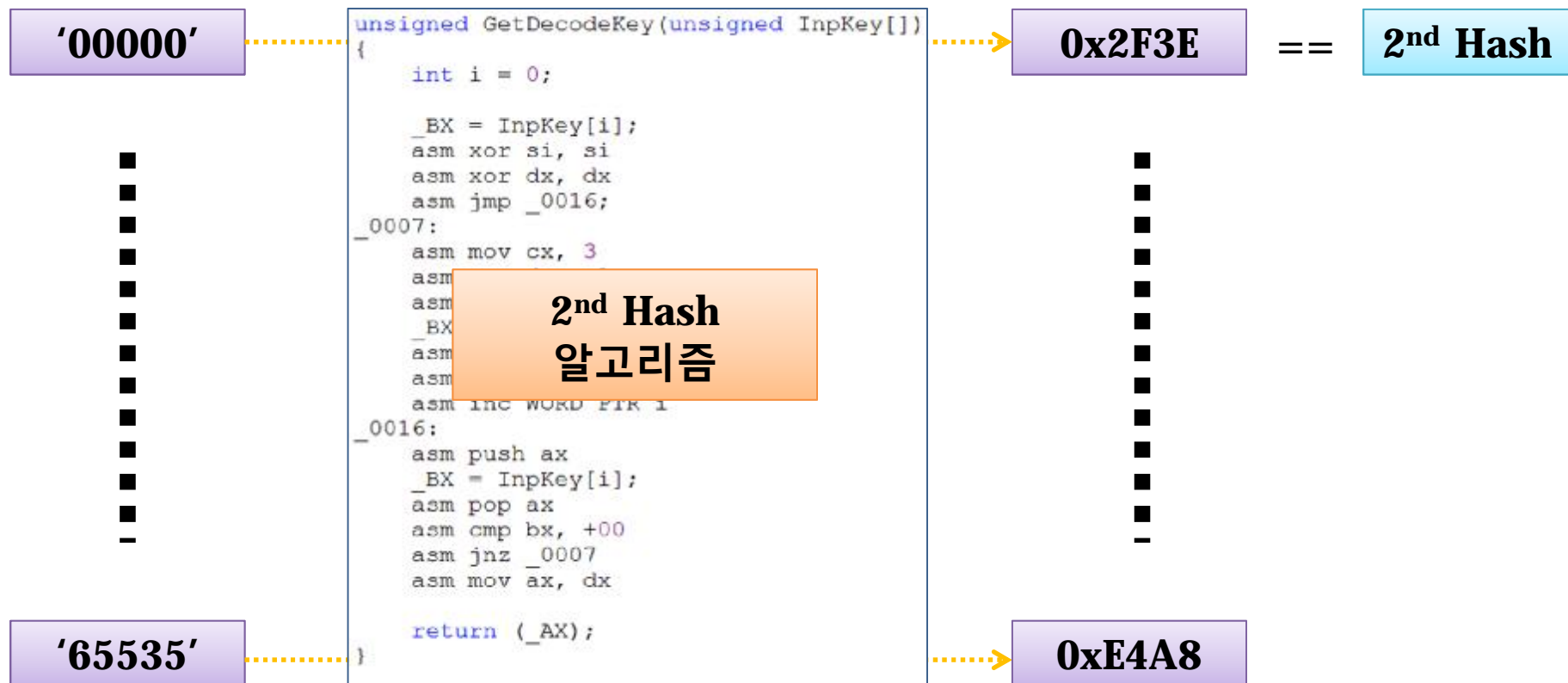
- 본문 암호화 방법
 - 2nd Hash, 본문 Char, 위치정보를 XOR해서 새로운 값 생성



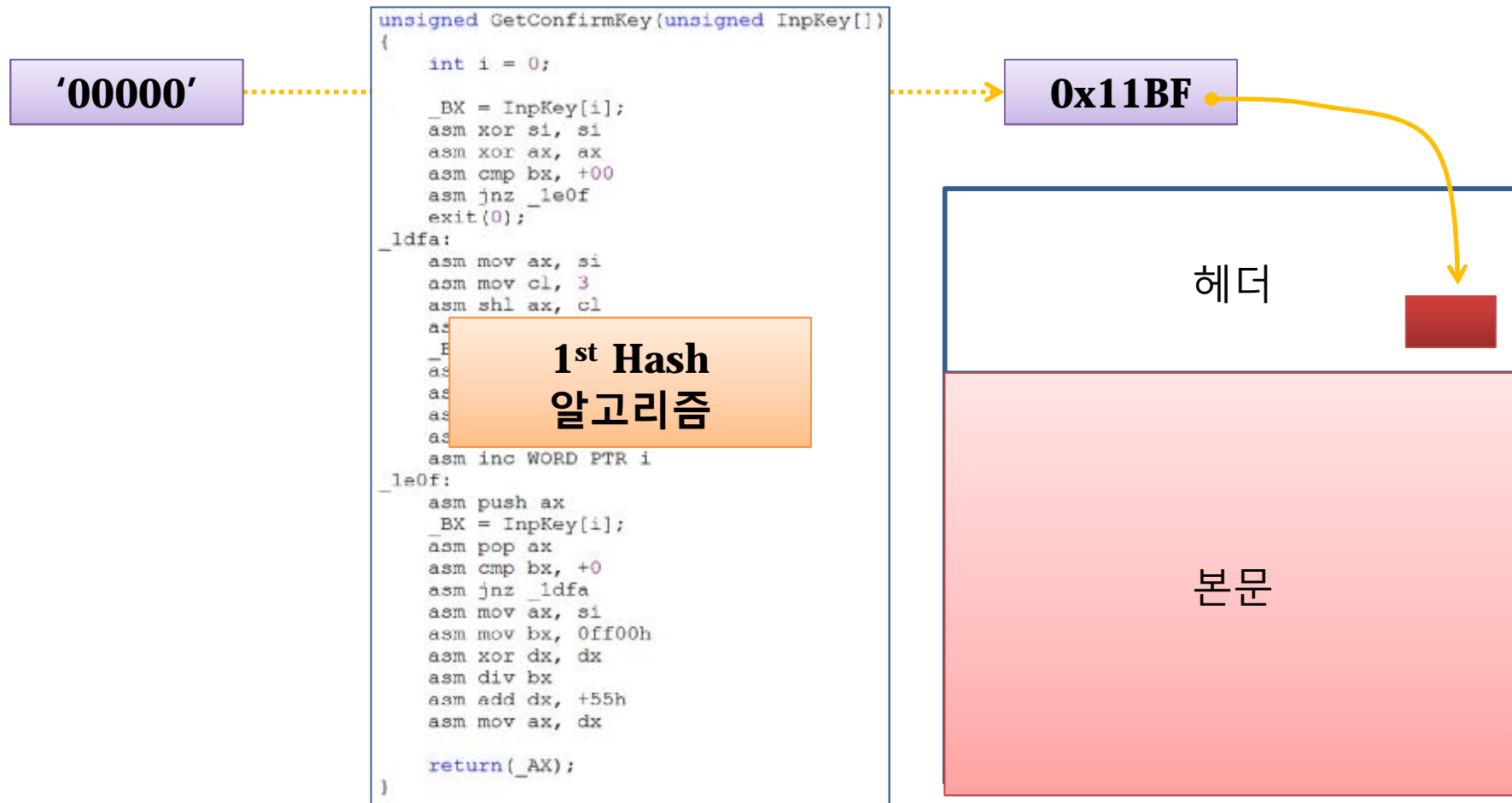
- 한글 2.x 암호 해독 공격 Idea (1)
 - 엔터키에 주목

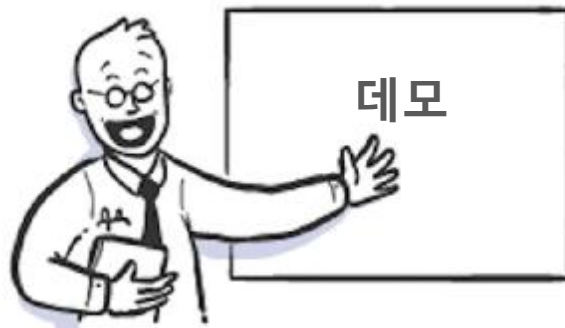


- 한글 2.x 암호 해독 공격 Idea (2)
 - Brute Force 방법으로 Idea(1)에서 알아낸 2nd Hash와 일치하는 Password 찾기



- 한글 2.x 암호 해독 공격 Idea (3)
 - Idea(1)에서 알아낸 2nd Hash와 일치하는 Password의 1st Hash 생성하여 한글 헤더에 기록하기





```
C:\WINDOWS\system32\cmd.exe
Y:\Dropbox\02_??\???HWP21\nphwp2>c:\Python27\python.exe nphwp2.py
-----
NPassware (for HWP 2.1) Ver 0.23 (October 22 2013)
Copyright (C) 2011-2013 NuriLab. All rights reserved.
-----

Usage: NPHwp.exe hwp-file [options]
Options:
    -?,          --help          this help

Y:\Dropbox\02_??\???HWP21\nphwp2>
```

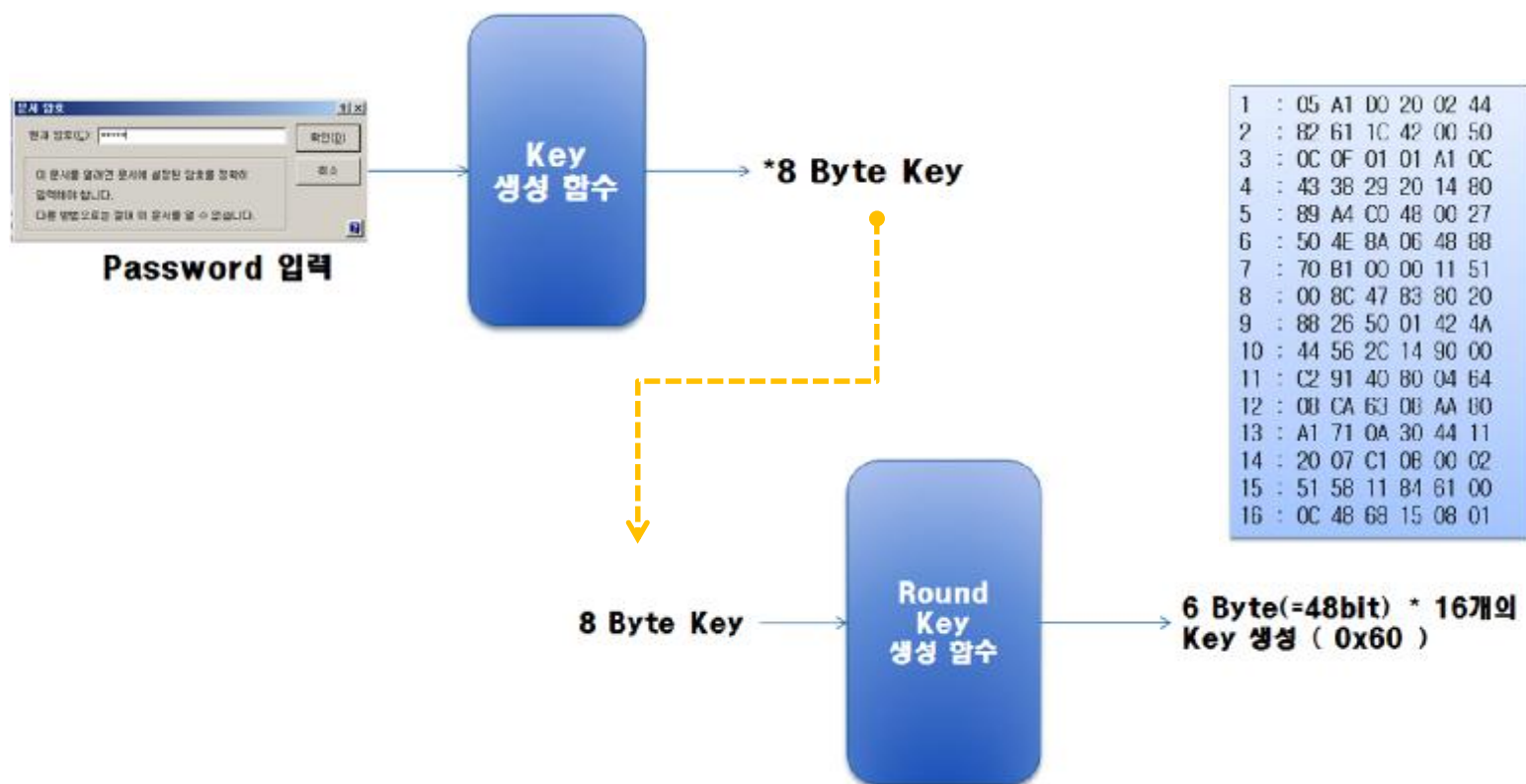
```
C:\WINDOWS\system32\cmd.exe
Y:\Dropbox\02_??\???HWP21\nphwp2>c:\Python27\python.exe nphwp2.py README_K.HWP
-----
NPassware (for HWP 2.1) Ver 0.23 (October 22 2013)
Copyright (C) 2011-2013 NuriLab. All rights reserved.
-----

[+] HWP file      : README_K.HWP
[-] Compressed   : 1
[-] Header Key   : 4D33
[-] Enter Key    : D29E
[+] Start Attack .....
[-] Found Decode Key : 0C64 (Password : 56562)
[-] Found Decode Key : 4C64 (Password : 56542)
[-] Found Decode Key : 8C64 (Password : 56522)
[-] Found Decode Key : CC64 (Password : 56502)
[+] End Attack .....

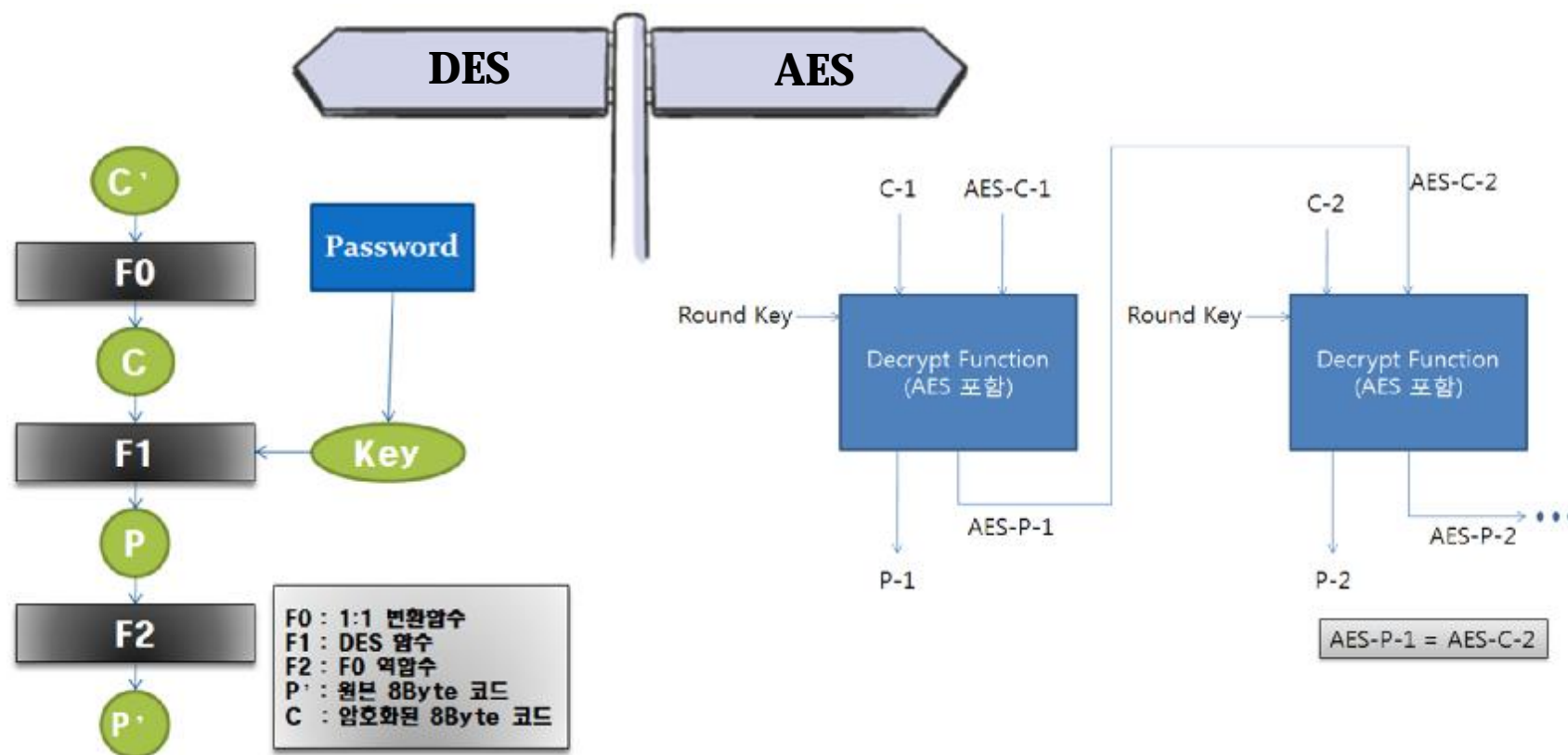
Y:\Dropbox\02_??\???HWP21\nphwp2>
```

한글 최신 버전의 암호 체계

- 한글 최신 버전의 암호화 방식 (1)
 - 암호화 검증키가 헤더에 저장되지 않음
 - Password를 8Byte Key로 변환
 - DES, AES 사용



- 한글 최신 버전의 암호화 방식 (2)
 - 복호화시에는 DES, AES 방법이 다름



한글 개인정보보호 암호 체계

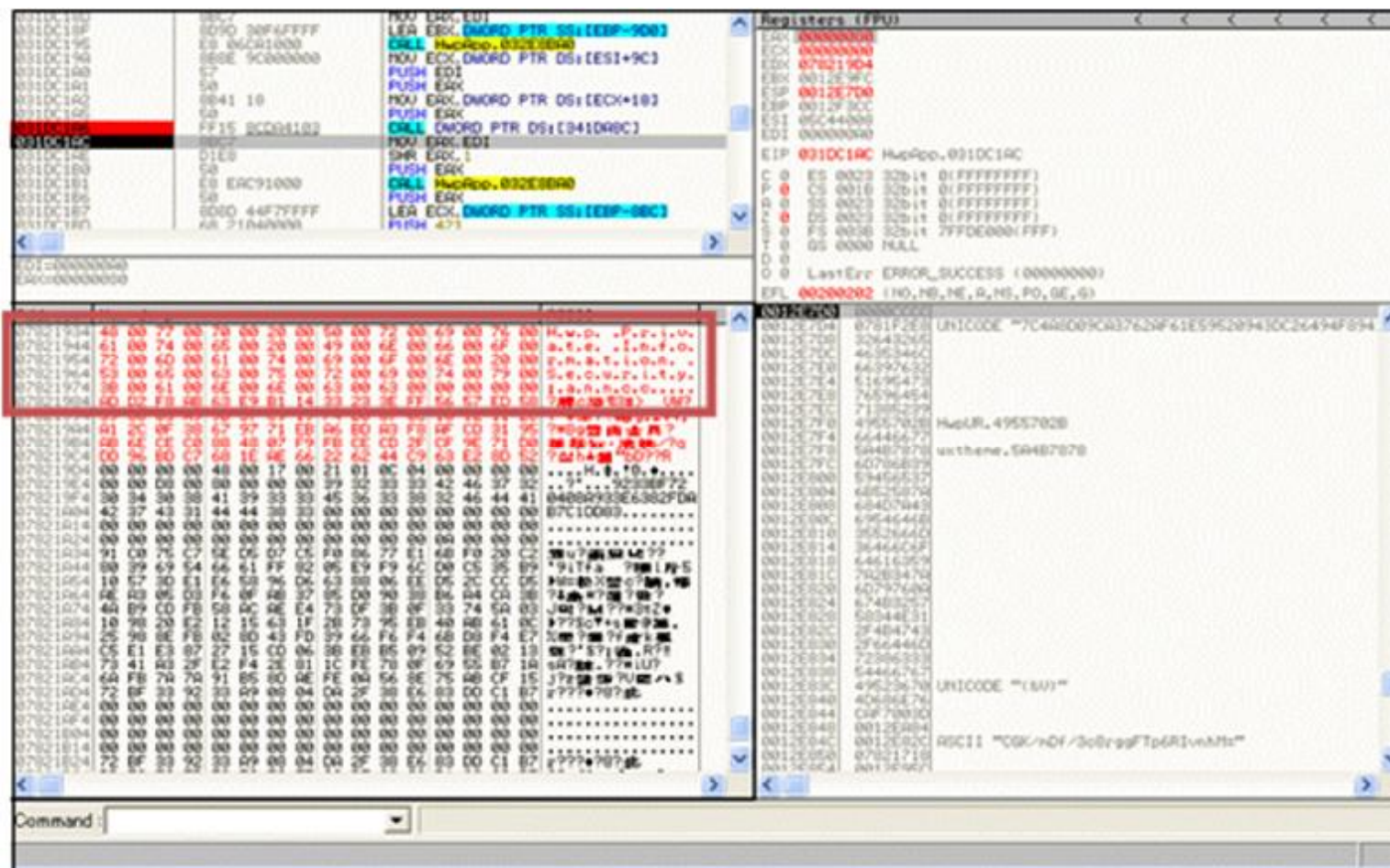
- 최신 한글 버전에 탑재된 개인정보보호
 - 사용자가 원하는 영역만 암호화 가능



- 사용자가 암호화 한 영역은 한글 포맷에서는 Base64로 저장되어 있음

000054E0	00 3A 00 65 00 32 00 64 00 32 00 6C 00 34 00 35	..e.2.d.2.1.4.5
000054F0	00 46 00 32 00 76 00 39 00 66 00 73 00 54 00 69	.F.2.v.9.f.s.T.i
00005500	00 51 00 54 00 64 00 59 00 76 00 39 00 52 00 38	.Q.T.d.Y.v.9.R.8
00005510	00 71 00 2B 00 70 00 55 00 49 00 77 00 66 00 44	.q.+p.U.I.w.f.D
00005520	00 66 00 78 00 78 00 4B 00 5A 00 39 00 6B 00 78	.f.x.x.K.Z.9.k.x
00005530	00 6D 00 37 00 65 00 45 00 59 00 7A 00 58 00 52	.m.7.e.E.Y.z.X.R
00005540	00 6B 00 43 00 7A 00 4D 00 68 00 6B 00 64 00 54	.k.C.z.M.h.k.d.T
00005550	00 69 00 6D 00 66 00 52 00 35 00 6F 00 6C 00 46	.i.m.f.R.5.o.l.F
00005560	00 36 00 59 00 63 00 61 00 64 00 7A 00 34 00 2B	.6.Y.c.a.d.z.4.+
00005570	00 7A 00 0A 00 76 00 79 00 6D 00 57 00 32 00 4B	.z...v.y.m.W.2.K
00005580	00 67 00 31 00 4E 00 34 00 58 00 43 00 47 00 4B	.g.1.N.4.X.C.G.K
00005590	00 2F 00 6D 00 44 00 66 00 2F 00 33 00 63 00 38	./m.D.f./3.c.8
000055A0	00 72 00 67 00 67 00 46 00 54 00 70 00 36 00 52	.r.g.g.F.T.p.6.R
000055B0	00 49 00 76 00 6E 00 68 00 4D 00 3D 00 20 00 4D	.I.v.n.h.M.=. .M

- 해당 Base64와 암호의 해시값이 적용되어 암호가 풀리면 특수 문자 존재
– “Hwp Private Information Security:”



- 암호문이 정상인지를 확인용 문자열
 - “Hwp Private Information Security:”
- 특수 문자열 뒤에 사용자가 적용한 개인정보 문자열 등장
 - Hwp Private Information Security:**010-6236-8888**

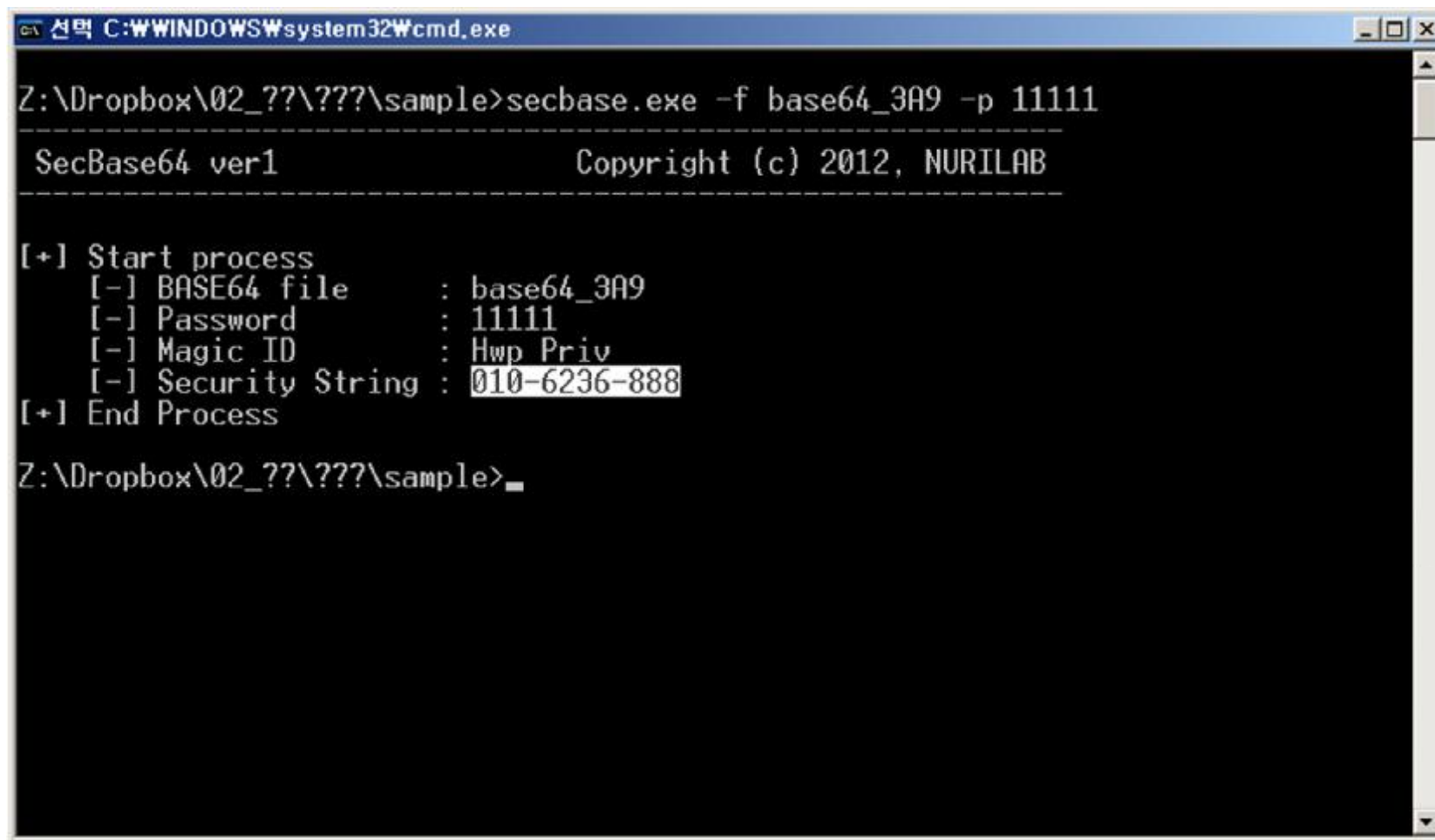
- 한글 문서에서 Base64 영역 추출 프로그램

```
C:\WINDOWS\system32\cmd.exe
Z:\Dropbox\02_??\???\sample>hwp2base64.exe -f 11111.hwp -o
-----
Hwp2Base64 ver1                      Copyright (c) 2012, NURILAB
-----

[+] Start process
[+] File name : 11111.hwp
[+] Scan BASE64 image :
    [-] Property      : 12
    [-] PPS Nname     : Section0
    [-] Position BASE64 : 3A9 (dumped base64_3A9)
[+] End process

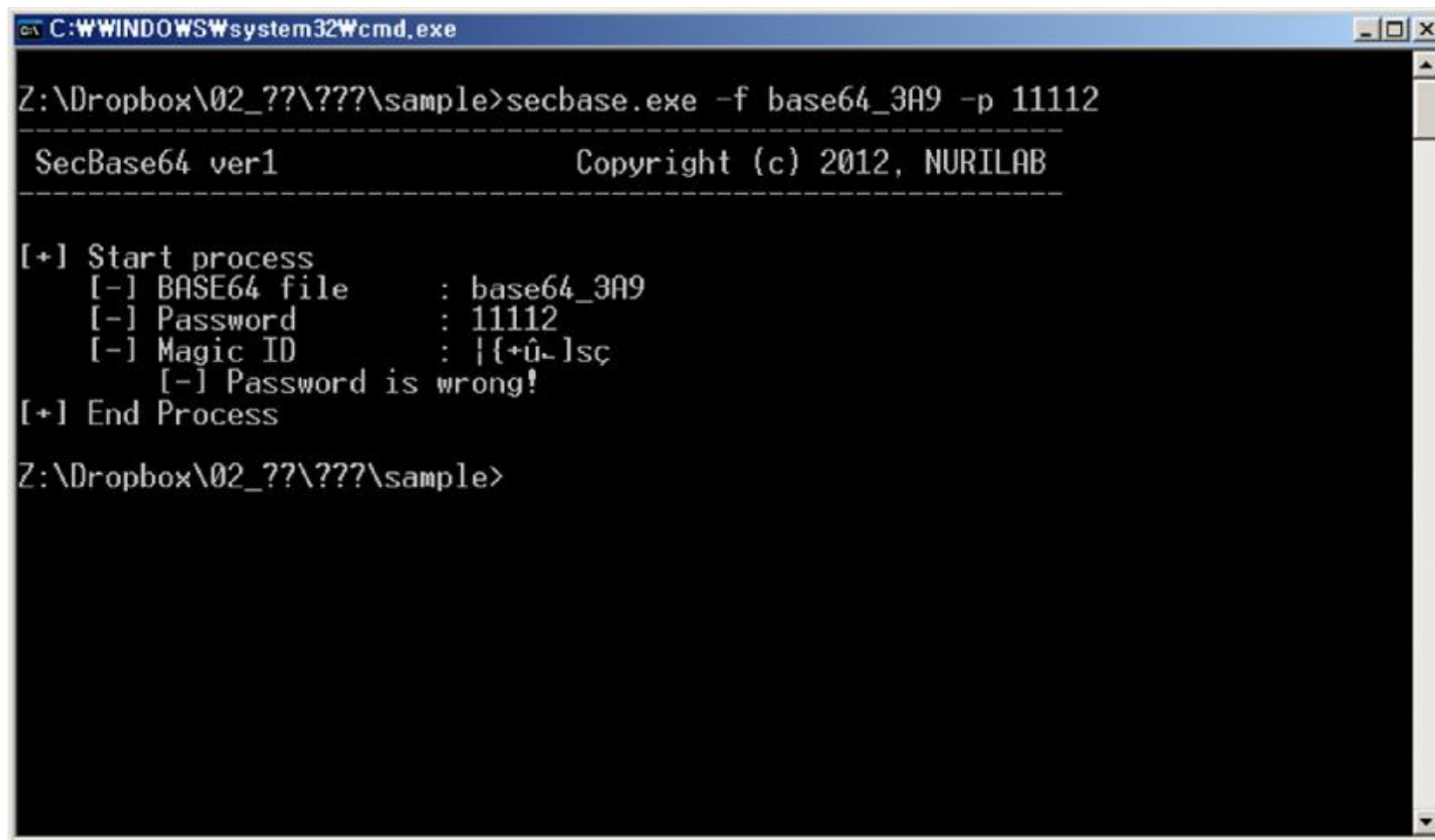
Z:\Dropbox\02_??\???\sample>
```


- 추출된 Base64에 대한 암호문 공격 (정상 암호일 경우)



```
C:\WINDOWS\system32\cmd.exe
Z:\Dropbox\02_??\???\sample>secbase.exe -f base64_3A9 -p 11111
-----
SecBase64 ver1                      Copyright (c) 2012, NURILAB
-----
[+] Start process
  [-] BASE64 file      : base64_3A9
  [-] Password         : 11111
  [-] Magic ID         : Hwp Priv
  [-] Security String  : 010-6236-888
[+] End Process
Z:\Dropbox\02_??\???\sample>
```

- 추출된 Base64에 대한 암호문 공격 (잘못된 암호일 경우)



```
C:\WINDOWS\system32\cmd.exe

Z:\Dropbox\02_??\???\sample>secbase.exe -f base64_3A9 -p 11112

-----
SecBase64 ver1                      Copyright (c) 2012, NURILAB
-----

[+] Start process
  [-] BASE64 file      : base64_3A9
  [-] Password         : 11112
  [-] Magic ID         : |{+û-}sç
      [-] Password is wrong!
[+] End Process

Z:\Dropbox\02_??\???\sample>
```

- 암호가 동일한 다른 문서의 Base64 덤프 내용 비교

Fairdell HexCmp

File Edit Search View Actions Options Help

First File - Z:\Dropbox\02_회사\누리협\한글 개인정보\sample\base64_3EF

OFFSET	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	44	72	55	9A	25	4A	46	29	2F	20	4B	BB	F2	03	E4	DC	DrU1%JF)/ K>ò.äÜ
00000010	FC	21	3D	A6	5A	D4	85	CA	F2	9B	E7	3D	4A	9B	13	B9	ü = Z0!Èò!ç=J!.
00000020	50	2A	3A	17	34	CE	53	7A	C4	AA	98	C0	20	09	76	AB	P*: 4!SzÄ!Ä .v<<
00000030	D4	74	4E	45	D7	42	0C	26	CF	CF	DC	98	AB	BC	B0	30	ÔtNExB.&I!U!<<4*0
00000040	DE	83	59	8F	4A	CF	C6	DF	CB	4F	E7	19	B5	14	57	38	p!V!J!ÄBÈOç.p.W8
00000050	92	6A	D8	A9	BD	B3	4A	A9	C4	70	A8	5E	56	66	1C	C4	'j004³J0Äp'^Vf.Ä
00000060	C0	59	7A	38	6E	D0	FB	AA	4B	E2	E0	21	06	B6	3A	C6	ÄYz8nDû³Kää!.¶:Ä

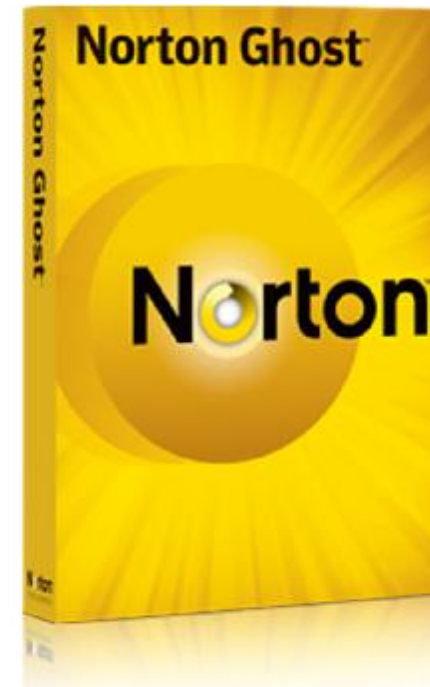
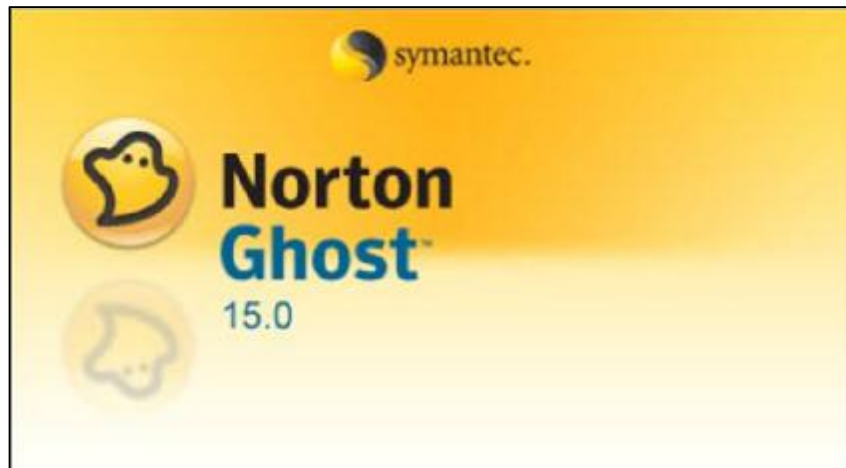
Second File - Z:\Dropbox\02_회사\누리협\한글 개인정보\sample\base64_B47_

OFFSET	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	44	72	55	9A	25	4A	46	29	2F	20	4B	BB	F2	03	E4	DC	DrU1%JF)/ K>ò.äÜ
00000010	FC	21	3D	A6	5A	D4	85	CA	F2	9B	E7	3D	4A	9B	13	B9	ü = Z0!Èò!ç=J!.
00000020	50	2A	3A	17	34	CE	53	7A	C4	AA	98	C0	20	09	76	AB	P*: 4!SzÄ!Ä .v<<
00000030	D4	74	4E	45	D7	42	0C	26	CF	CF	DC	98	AB	BC	B0	30	ÔtNExB.&I!U!<<4*0
00000040	DE	83	80	53	AD	7D	28	F7	E5	9F	23	50	74	DC	5B	FC	p!IS-)(-ä!#PtÜ(ü
00000050	50	1A	FC	A5	D6	C0	A1	6B	3A	47	E0	23	A5	1F	F2	F5	P.ü#0Ä!k:Gä#¶.ôö
00000060	E4	80	90	C7	17	D0	E2	19	8F	40	86	5C	A3	AA	D1	D5	ä!!ç.Ää. !@!n!NÖ
00000070	DF	60	18	09	FB	49	2E	53	C9	55	1B	1B	C4	E0	2C	93	B'. ä! .SEU..Ää. !
00000080	CA	A5	AB	26	1C	3D	28	69	CB	37	04	6E	66	B6	C2	4E	E¶<<.= (iE7.nf¶ÄN
00000090	D5	94	3C	12	82	9C	A5	D7	81	E5	32	8A	EF	17	7A	A3	Ö!k. !!¶x!ä2!i. zê
000000A0	FF	9F	5C	79	86	7C	C5	D6	82	3A	43	C8	E3	11	03	33	y!y! ÄÖ! :CEÄ..3
000000B0	84	E5	74	9F	99	3F	4F	26	34	69	1B	99	BB	E2	37	CA	!ä! !?0&4i. !>>Ä7E
000000C0	01	34	59	B2	BB	56	53	44	E4	41	8C	21	D6	FE	0D	B0	.4Y'>>VSDäA! !Öp.*
000000D0	FB	88	F7	EC	21	83	3B	30	38	DC	7D	67	DB	F1	ED	F6	ä!÷i! !:08Ü!qÜX!ö

Shift: HEX : 0x0 DEC : 0

노턴 고스트 암호 체계

- 노턴 고스트는 디스크 백업본 생성 시 암호 입력을 할 수 있음
- 복원 시 암호 필요
- 백업본 생성시 History 파일 생성
 - 여기에 ID와 Password Key 기록



- 노턴 고스트는 암호의 정보를 로그에 암호화 해서 기록하고 있음

<pre>#include <stdio.h> struct a1 { int t1; wchar_t *ID; int t2; int t3; int t4; int t5; int t6; }; // user arg // wchar_t *ID = L"{20376B5D-9BB1-4068-A63F-E3 // char key[] = "\x50\x56\xab\x06\x81\xea\x59\ // user arg wchar_t *ID = L"{5A4E8E70-0ED6-42A7-9D74-67D48 char key[] = "\xe3\x20\x01\x79\x15\x81\x3f\x98 // mai int me (DV st ch C:\Documents and Settings\inca\? ch key : 123456 me me C:\Documents and Settings\inca\? // in me //</pre>	<pre><Triggered vt="19">0</Triggered> <ID vt="8">{5A4E8E70-0ED6-42A7-9D74-67D48DACF82A}</ID> <State vt="19">0</State> <Timestamp vt="7">9/ 9/2012 9:40:52</Timestamp> - <Location vt="9" clsid="{C7C5381C-FA3F-4C57-AB92-9274526DBFC8}"> <Name vt="8">folder</Name> <DisplayName vt="8">&Local file</DisplayName> <CountQuota vt="11">-1</CountQuota> <SizeQuota vt="11">0</SizeQuota> <Remote vt="11">-1</Remote> <FileSpec vt="8">anncc-a28677fac_E_Drive</FileSpec> <Extension vt="8">.v2i</Extension> <GUIDValue vt="8">\\?\Volume{5f0d676f-c4eb-11e0-8e00-806d617 <DisplayPath vt="8">C:\Documents and Settings\Administrator\바탕 <IsDedupeLocation vt="11">0</IsDedupeLocation> </Location> <JobID vt="8">{DE97AAF5-0733-4C28-A23D-27536048F866}</JobID> <RetargetTag vt="11">0</RetargetTag> <Rules vt="8" /> <Result vt="8" /> <SpanCount vt="19">1</SpanCount> <Verified vt="11">0</Verified> <Indexed vt="11">0</Indexed> <SystemFilesIndexed vt="11">0</SystemFilesIndexed> <IncludesACL vt="11">0</IncludesACL> <OriginalDriveLetter vt="8">E:\</OriginalDriveLetter> <DisplayJobName vt="8">Drive Backup of ghost (E:\)</DisplayJobName> <QuiescedState vt="19">1</QuiescedState> <Obsolete vt="11">0</Obsolete> <ElapsedTime vt="21">13</ElapsedTime> <Size vt="21">445468</Size> <MinVolumeSize vt="21">8193024</MinVolumeSize> <EncryptedPW2 vt="8">5800500044005D0001000000D08C9DDF0115D1118C7A00C0- <AESPassword vt="8">e320c17915813f9881170563</AESPassword></pre>
---	---

결론

- 암호화 알고리즘에 대한 공격
 - 암호화 알고리즘 공격은 불가능하다.
- 암호화 알고리즘 사용상의 취약점을 찾아야 함
 - 암호를 비교하기 위한 특수 값들이 존재
 - 특수 값이 공격의 취약점이 됨
- 해당 공격의 유효성만 확인되면...
 - Rainbow Table 생성하여 공격의 속도를 줄임

A close-up photograph of a fountain pen with a silver-colored nib writing the words "Thank you" in a cursive script on a piece of textured, light brown paper. The pen is positioned on the right side of the frame, with its nib touching the paper at the end of the word "you". The lighting is warm and focused on the writing area.

Thank you

Code Engn

www.CodeEngn.com

2014 CodeEngn Conference 10