4번 문제 답안

정답)

AES의 마스터 키는 **0x8417ff2160a86452ee23d0147f35d8e**이며, 복호화한 문서 속 문제의 정답은 d = **0xbea4fd03c804ea0160a096f4c6b438d54ab78458e124e8f68d42cd7010807a1b**이다.

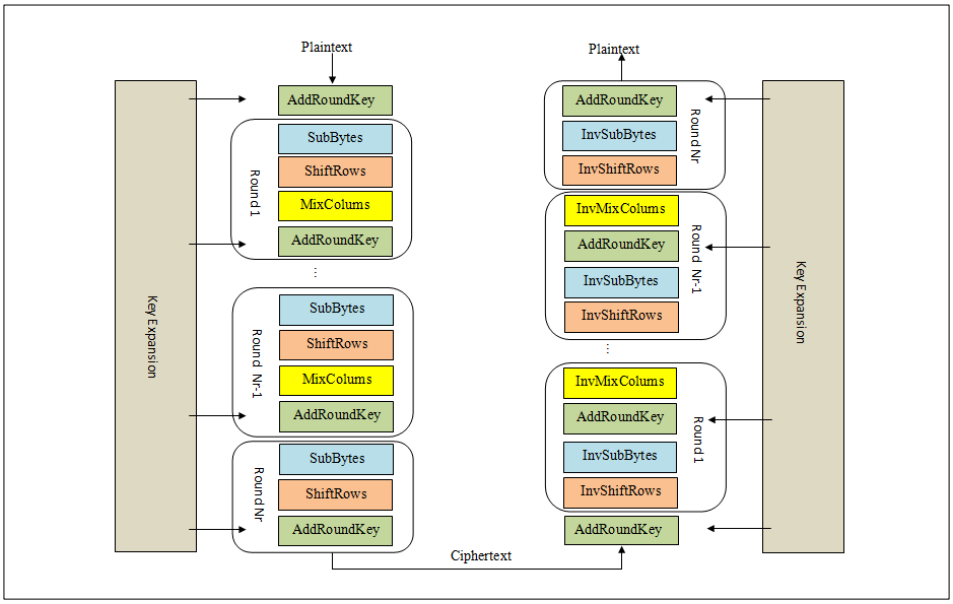
풀이)

[I . Background]

풀이를 위해 몇 가지 기본 개념이 필요하다.

1. AES-128

AES 암호화 방식으로 데이터를 128비트(16바이트) 블록단위로 암호화/복호화를 하며, 총 10라운드로 구성된다. 라운드는 SubBytes, ShiftRows, MixColumns, AddRoundKey로 구성되며, 암호화/복호화의 논리 일치를 위해 마지막 라운드의 MixColumns는 생략된다.



[그림 1. AES의 암호화/복호화. AES-128의 경우 (Nr=10)]

출처 https://www.commonlounge.com/discussion/e32fdd267aaa4240a4464723bc74d0a5

복호화 과정에서 라운드는 InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns 순으로 구성되는데, 사전 작업을 통해 AddRoundKey와 InvMixColumns의 순서를 바꿀 수 있다.

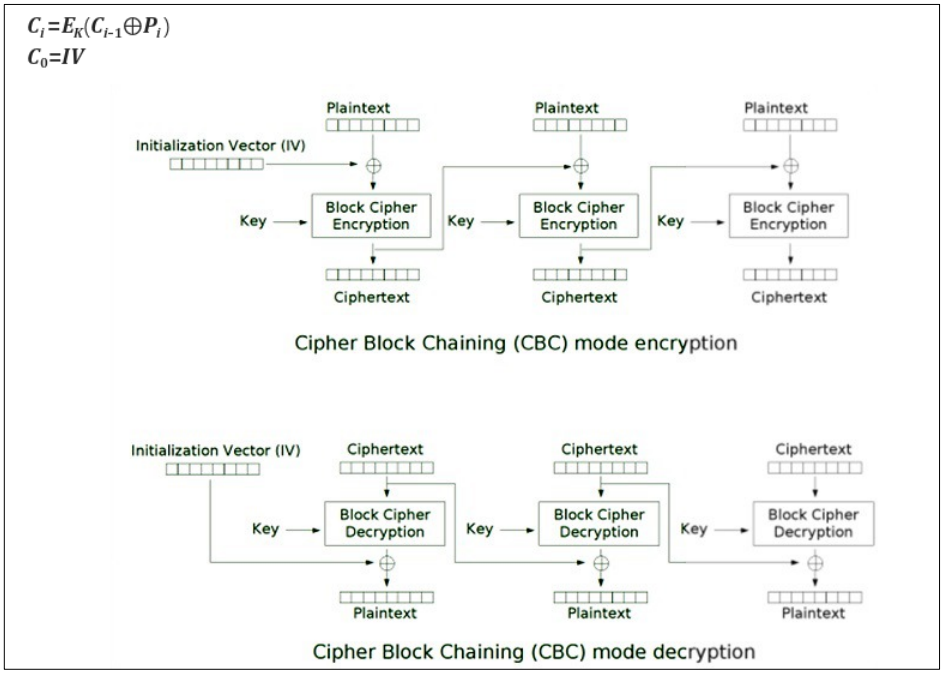
I : input, O : output, A : AddRoundKey에서 더해지는 행렬, M : InvMixColumns에서 곱해지는 행렬이라고 할 때 기존 복호화는 AddRoundKey → InvMixColumns 순서로 진행되므로 식으로 표현하면 아래와 같다.

$$O = M \times (I + A) \Rightarrow O = (M \times I) + (M \times A)$$

4번 문제 답안

분배법칙에 의해 식을 풀어 쓸 수 있고 A에 미리 M을 곱하면 InvMixColumns → AddRoundKey 순서로 바꾸어 계산할 수 있다.

2. CBC 운영모드



[그림 2. CBC 운영모드]

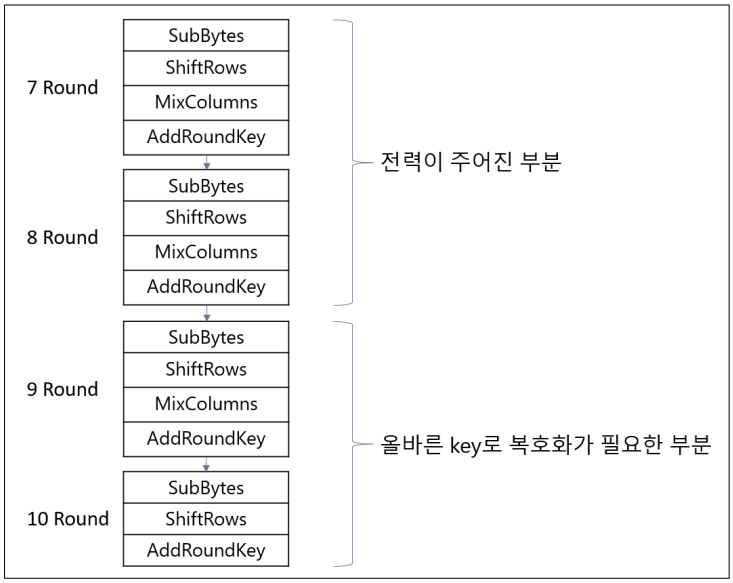
CBC 운영모드의 복호화는 $P_0 = D(C_0) \oplus IV, P_n = D(C_n) \oplus C_{n-1} (n > 0)$으로 계산된다. IV = 0으로 주어졌기 때문에 암호문의 첫 16바이트를 예측한 key로 복호화한 후, 최초 8 bytes가 'D0 CF 11 E0 A1 B1 1A E1'인지 확인하여 올바른 키를 찾을 수 있다.

3. CPA (Correlation Power Analysis)

암호화 연산 도중 발생한 소비 전력을 이용하여 암호키를 찾는 전력 분석 기법이다. 주어진 소비 전력과 예측한 key로 복호화한 데이터 사이의 상관계수를 구하여 상관계수가 크게 나타나는 값을 키로 찾을 수 있다.

문제에서 소비 전력은 AES-128의 10라운드 중 7, 8라운드에 해당하는 전력소비량만이 주어졌다. 따라서 CPA를 통해 키를 찾기 위해서는 암호화에서 8라운드가 끝나는 시점까지 올바르게 복호화를 진행해야 한다. 즉, 아래 그림에서 보이듯 암호화에 9, 10라운드인 SubBytes → ShiftRows → MixColumns → AddRoundKey → SubBytes → ShiftRows → AddRoundKey 7개 연산에 대한 복호화를 진행해야 한다.

4번 문제 답안



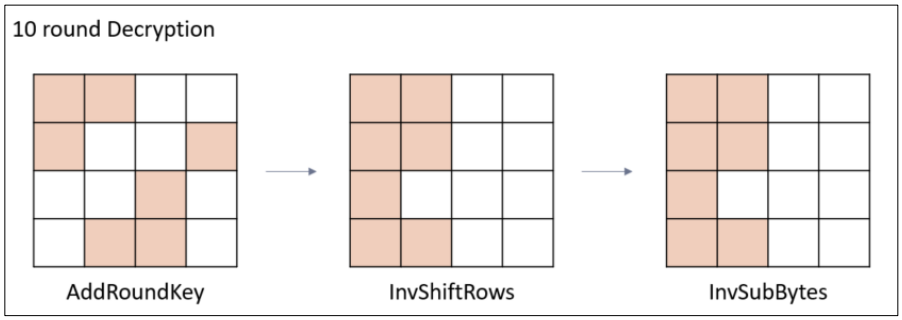
따라서 주어진 암호문을 AddRoundKey(10라운드 키) → InvShiftRows → InvSubBytes → AddRoundKey(9라운드 키) → InvMixColumns → InvShiftRows → InvSubBytes 한 결과와 소비 전력 사이의 상관계수를 계산한다. 이때, AddRoundKey와 InvMixColumns의 순서를 바꾸어 AddRoundKey(10라운드 키) → InvShiftRows → InvSubBytes → InvMixColumns → AddRoundKey(M과 곱한 9라운드 키) → InvShiftRows → InvSubBytes 순서로 복호화를 한다.

[Ⅱ. AES-128의 마스터 키 찾기]

주어진 10라운드 키의 일부를 활용해 9라운드 키의 일부를 찾은 후, 8라운드 키 일부를 전수조사 하여 마스터 키를 찾는다.

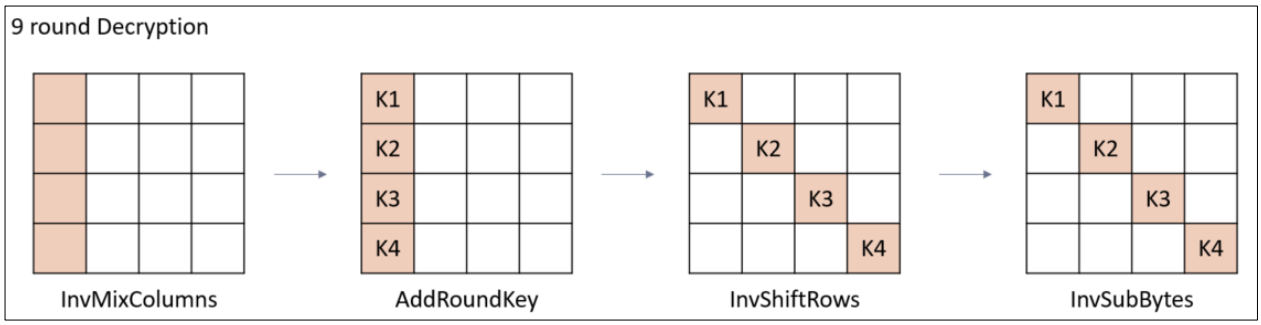
1. 9라운드 키 일부(1, 2, 3, 4번째 바이트) 찾기

주어진 10라운드 키의 일부(1, 2, 5, 8, 11, 12, 14번째 바이트)로 3개의 연산 AddRoundKey(10라운드 키) → InvShiftRows → InvSubBytes에 대한 복호화를 진행하면 아래와 같이 색칠 되어 있는 바이트가 올바르게 복호화 된다.

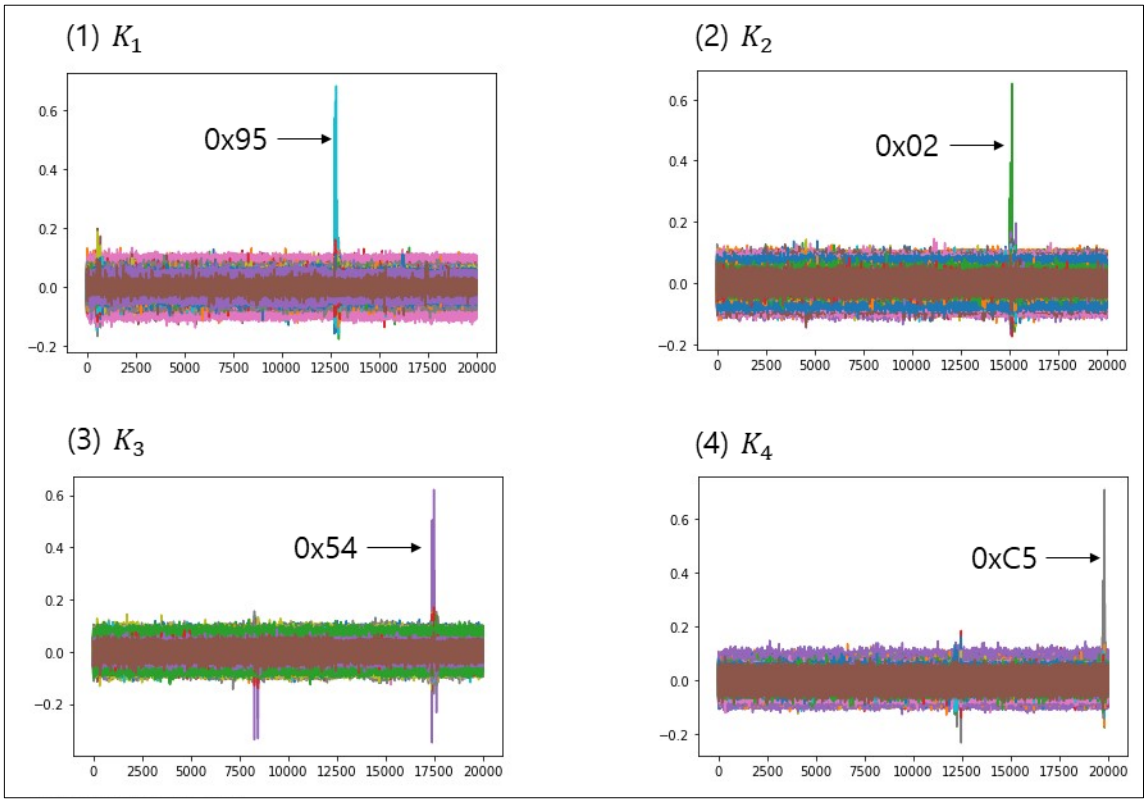


4번 문제 답안

3개 연산을 복호화한 후, InvMixColumns를 하면 데이터의 1, 2, 3, 4번째 바이트만이 올바르게 복호화된다. 따라서 총 4번의 CPA를 통해 9라운드 키의 1, 2, 3, 4번째 바이트를 찾는다. CPA를 할 때 고려해야 할 점 AddRoundKey 이후, InvShiftRows로 위치가 변한다는 것이다. 4번째 바이트를 찾기 위해서는 복호화한 데이터의 16번째 바이트로 상관계수를 비교해야 한다.



K_1, K_2, K_3, K_4에 대한 CPA 결과는 아래와 같다.



4번 문제 답안

찾은 $[K_1, K_2, K_3, K_4] = [0x95, 0x02, 0x54, 0xC5]$는 InvMixColumns에서 곱해지는 행렬과 9라운드 키의 곱이다. 9라운드 키를 찾으려면 InvMixColumns에서 곱해지는 행렬의 역행렬, 즉 MixColumns에서 곱해지는 행렬과 곱해야 한다.

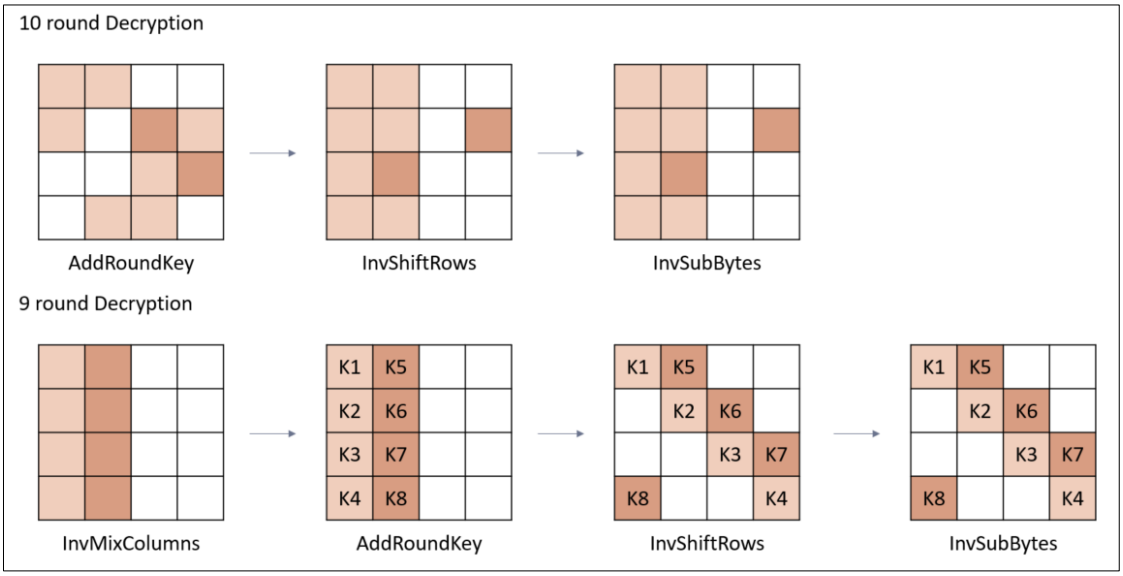
$$\begin{bmatrix} 2&3&1&1 \\ 1&2&3&1 \\ 1&1&2&3 \\ 3&1&1&2 \end{bmatrix} \times \begin{bmatrix} 0x95 \\ 0x02 \\ 0x54 \\ 0xC5 \end{bmatrix} = \begin{bmatrix} 0xA6 \\ 0xA8 \\ 0x6B \\ 0x63 \end{bmatrix}$$

따라서 **9라운드 키 1, 2, 3, 4번째 바이트는 0xA6, 0xA8, 0x6B, 0x63**이다. 그리고 Key Scheduling을 통해 **10라운드 키의 10, 15번째 바이트가 0xA5, 0x3E** 임을 알 수 있다.

9라운드	A6	A8	6B	63	*	*	*	*	*	*	*	*	*	*	*	*
10라운드	22	E5	*	*	E9	*	*	4C	*	A5	5B	2C	*	9B	3E	*

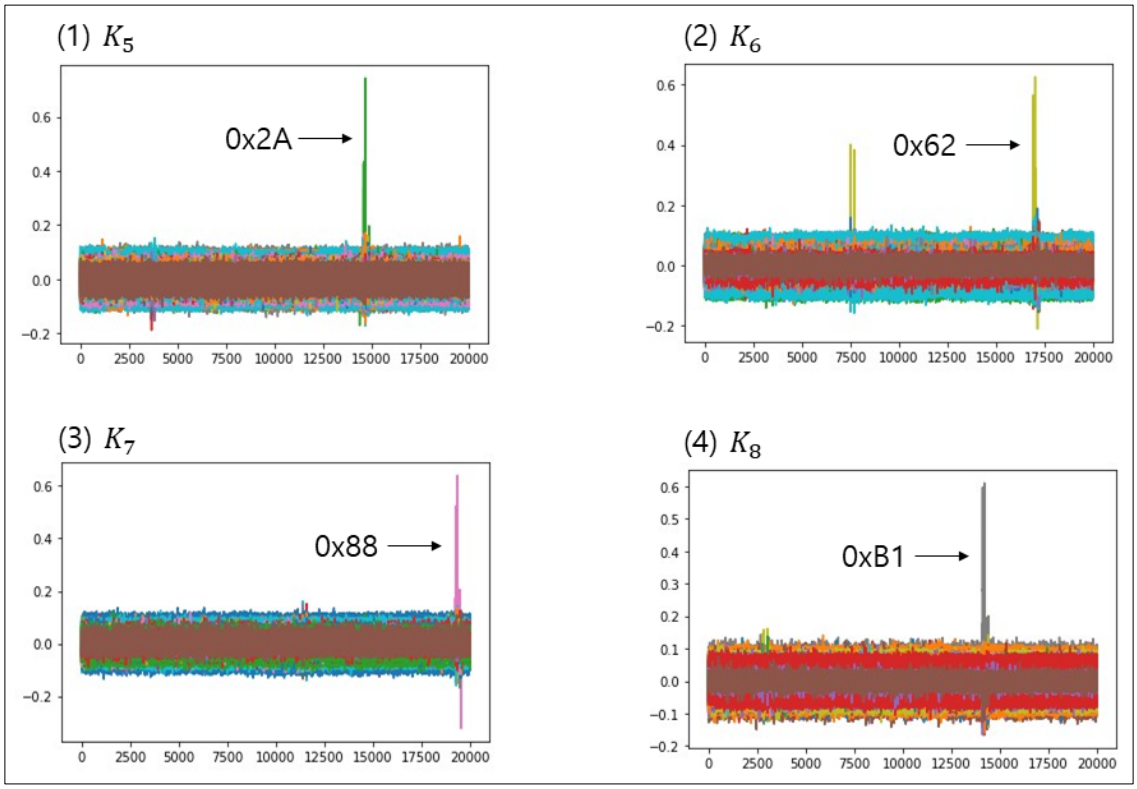
2. 9라운드 키 일부(5, 6, 7, 8번째 바이트) 찾기

1번 과정에서 10라운드 키의 10, 15번째 바이트를 알게 되어 위와 같은 방식으로 9라운드 키의 5, 6, 7, 8번째 바이트도 구할 수 있다.



4번 문제 답안

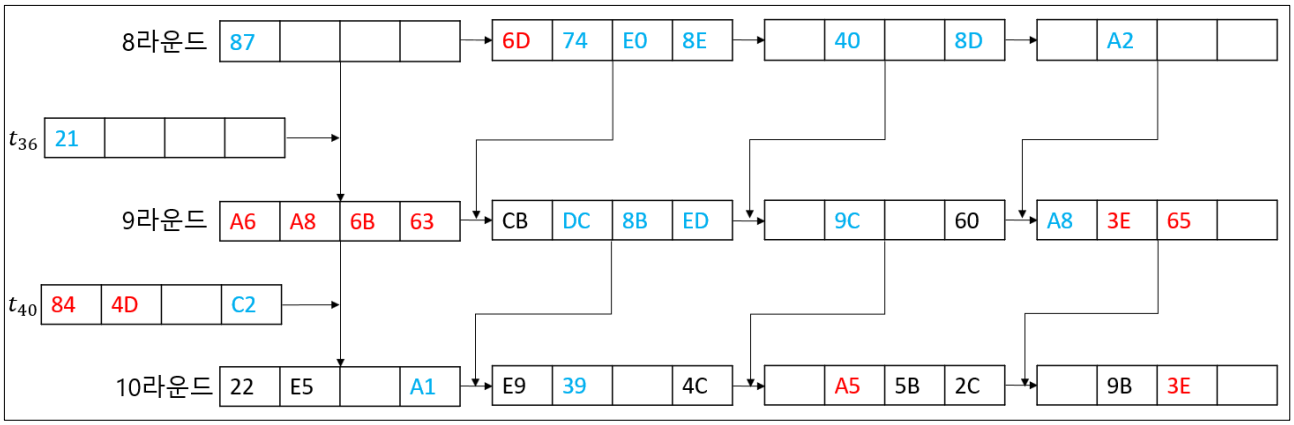
K_5, K_6, K_7, K_8에 대한 CPA 결과는 아래와 같다.



찾은 $[K_5, K_6, K_7, K_8] = [0x2A, 0x62, 0x88, 0xB1]$에 마찬가지로 MixColumns에 곱해지는 행렬과 곱해 **9라운드 키 5, 6, 7, 8번째 바이트는 0xCB, 0xDC, 0x8B, 0xED** 임을 알 수 있다.

3. 8라운드 키 일부 전수조사

현재까지 찾은 라운드 키들을 통해 알 수 있는 값들을 정리해 전수조사가 필요한 부분을 분류한다.



[그림 10. 1, 2번 과정을 통해 알 수 있는 라운드 키]

4번 문제 답안

1, 2번 과정을 통해서 10라운드 키를 모두 알 수 없기 때문에 라운드 키 일부를 전수조사할 필요가 있다. Key Scheduling에 의해, 8라운드의 13, 16번째 바이트만 알면 10라운드 키 전체를 알 수 있고, 10라운드 키를 사용해 마스터 키를 알 수 있다. 따라서 2바이트 전수조사를 통해 마스터 키를 찾는다.

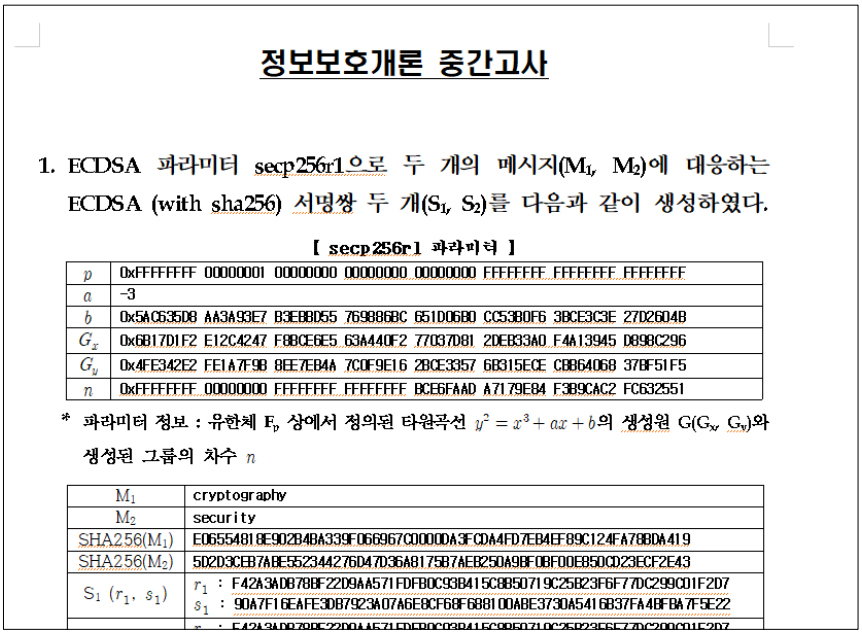
아래는 이 과정을 Pseudocode로 표현한 것이다.

```
a1:= 8라운드 키의 13번째 바이트
a2:= 8라운드 키의 16번째 바이트

for a1 in [0, 255]:
    for a2 in [0, 255]:
        1. a1, a2로 10라운드 키 계산
        2. 10라운드 키로 마스터 키 계산
        3. 마스터 키로 exam.hwp.enc 파일 복호화
        4. if 복호화 파일 최초 8바이트 == 'D0 CF 11 E0 A1 B1 1A E1' : return [a1, a2]
```

4. exam.hwp.enc 파일 복호화

문제에서 사용된 AES-128의 마스터 키는 **0x8417ff2160a86452ee23d0147f35d8e**이고, 이를 이용해 복호화를 진행한다.



정보보호개론 중간고사

1. ECDSA 파라미터 secp256r1으로 두 개의 메시지(M_1, M_2)에 대응하는 ECDSA (with sha256) 서명쌍 두 개(S_1, S_2)를 다음과 같이 생성하였다.

【 secp256r1 파라미터 】

p	0xFFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF
a	-3
b	0x5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B
G_x	0x6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296
G_y	0x4FE342E2 FE1A7F9B 8EE7EB4A 7CDF9E16 2BCE3357 6B315ECE CBB64068 37BF51F5
n	0xFFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551

* 파라미터 정보 : 유한체 F_p 상에서 정의된 타원곡선 $y^2 = x^3 + ax + b$의 생성원 $G(G_x, G_y)$와 생성된 그룹의 차수 n

M_1	cryptography
M_2	security
SHA256(M_1)	E0655481 8E902B4B A339FD66 967CD00D A3FCDA4F D7EB4EF8 9C124FA7 8BDA419
SHA256(M_2)	5D2D3CEB 7ABE5523 4427 6D47D36A 8175B7AE B250A9BF 0BF00E85 0CD23ECF 2E43
S_1 (r_1, s_1)	r_1 : F42A3ADB 78BF22D9 AA571FDF B0C93B41 5C8B5071 9C25B23F 6F77DC29 9CD1F2D7 s_1 : 9DA7F16E AFE3DB79 23A07A6E 8CF68F68 810DABE3 730A5416 B37FA4BF BA7F5E22

[그림 11. 복호화한 문서의 일부]

복호화 파일의 SHA256 값 : 0x185463b03cf29b0276b8d7747297455e71e82c037442e92f7cc86de601dcc867

4번 문제 답안

[Ⅲ. 파일(exam.hwp) 속 문제 풀이]

1. ECDSA 문제 분석

문제에서 주어진 M_1, M_2에 대한 각각의 SHA256 값을 h_1, h_2라 하자. 개인키 d에 대해 ECDSA 서명은 아래와 같이 생성된다.

(1) 난수 k_1을 뽑아 점 k_1G의 x좌표 $x(k_1G)$에 대해 $r_1 = x(k_1G) \pmod n$라 하면, $s_1 = k_1^{-1}(h_1 + dr_1) \pmod n$를 계산한다. 문제에서 주어진 S_1은 (r_1, s_1)이다.

(2) 난수 k_2을 뽑아 점 k_2G의 x좌표 $x(k_2G)$에 대해 $r_2 = x(k_2G) \pmod n$라 하면, $s_2 = k_2^{-1}(h_2 + dr_2) \pmod n$를 계산한다. 문제에서 주어진 S_2는 (r_2, s_2)이다.

S_1과 S_2를 보면 r_1, r_2가 같다. 따라서 $x(k_1G) \pmod n = r_1 = r_2 = x(k_2G) \pmod n$ 이므로 $k_1 = k_2 \pmod n$ 또는 $k_1 = -k_2 \pmod n$이다.

2. $k_1 = k_2 \pmod n$인 경우

$k = k_1 = k_2 \pmod n$라 하자.

$$s_1 = k^{-1}(h_1 + dr) \pmod n \Rightarrow k = s_1^{-1}(h_1 + dr) \pmod n$$
$$s_2 = k^{-1}(h_2 + dr) \pmod n \Rightarrow k = s_2^{-1}(h_2 + dr) \pmod n$$

이므로

$$s_1^{-1}(h_1 + dr) = s_2^{-1}(h_2 + dr) \pmod n$$

식을 정리하면

$$d = (s_1h_2 - s_2h_1)((s_2 - s_1)r)^{-1} \pmod n$$

이다. 따라서 식을 통하여

d=0x85cd61964a0aaecdf53e19eb254cb1c07fd7ee85b4a7474fea7645cebf2432d4

을 얻는다. 하지만 $dG \neq Q$ 이므로 해당 d는 개인키가 아니다.

3. $k_1 = -k_2 \pmod n$인 경우

$k = k_1 = k_2 \pmod n$라 하자.

$$s_1 = k_1^{-1}(h_1 + dr) \pmod n \Rightarrow k_1 = s_1^{-1}(h_1 + dr) \pmod n$$
$$s_2 = k_2^{-1}(h_2 + dr) \pmod n \Rightarrow k_2 = s_2^{-1}(h_2 + dr) \pmod n$$

이고,

$$k_1 = -k_2 \pmod n$$

이므로 다음 식을 도출할 수 있다.

$$d = -(s_1h_2 + s_2h_1)((s_1 + s_2)r)^{-1} \pmod n$$

4번 문제 답안

식을 통하여

d=0xbea4fd03c804ea0160a096f4c6b438d54ab78458e124e8f68d42cd7010807a1b

을 얻는다. 해당 d는 $dG = Q$를 만족한다.

따라서 개인키 d=0xbea4fd03c804ea0160a096f4c6b438d54ab78458e124e8f68d42cd7010807a1b 이다.