Chap 3.

1. 원격에 있는 두 사용자가 동전 던지기를 하기 위해 다음과 같은 프로토콜을 사용할 수 있다.
   - 단계 1. Alice는 매우 큰 양의 정수 X를 임의로 선택하여 그것의 해시값 $H(X)$를 Bob에게 전달한다.
   - 단계 2. Bob은 짝수, 홀수 중 하나를 선택하여 Alice에게 자신의 선택을 알린다.
   - 단계 3. Alice는 X를 Bob에게 전달한다. Bob의 추측이 맞으면 Bob이 이긴 것이 된다.

   이 프로토콜과 관련하여 다음 각각에 대해 답하시오.

   ① 선택 가능한 X의 범위가 이 프로토콜의 안전성에 어떤 영향을 주는지 설명하시오.
   ② Bob이 무엇을 할 수 있으면 항상 이길 수 있는지 제시하고, 해시함수의 어떤 특성 때문에 그것이 가능하지 않은지 제시하시오.
   ③ Alice가 무엇을 할 수 있으면 항상 이길 수 있는지 제시하고, 해시함수의 어떤 특성 때문에 그것이 가능하지 않은지 제시하시오.

   위 프로토콜을 블록방식의 대칭 암호알고리즘을 사용하여 다음과 같이 변경하였다.

   - 단계 1. Alice는 블록 크기의 랜덤한 값 X와 대칭키 K를 임의로 선택하여 X를 K로 채우기 없이 ECB 모드로 암호화한 암호문을 Bob에게 전달한다.
   - 단계 2. Bob은 짝수, 홀수 중 하나를 선택하여 Alice에게 자신의 선택을 알려준다.
   - 단계 3. Alice는 K를 Bob에게 전달한다.

   ④ 이 프로토콜에서 Alice가 무엇을 할 수 있으면 항상 이길 수 있는지 제시하고, 이 측면에서 이 프로토콜의 안정성을 논하시오.

A - 1. X의 범위가 너무 작을 경우, B가 가능한 모든 X값을 전부 대입하여 해시값을 알아낸 뒤에 X가 짝수인지 홀수인지 알아낼 수 있기 때문에, 안전성이 낮아진다.
A - 2. Bob은 H(X) = Y를 만족하는 X값을 해시함수를 통해 역으로 구할 수 있게 되면 항상 이길 수 있다. 하지만 해시함수는 역방향으로는 값을 구하는것이 계산적으로 어렵기 때문에 가능하지 않다.
A - 3. Alice는 H(X) = H(Y)를 만족하는 X값과 Y값을 찾을 수 있게 되면 항상 이길 수 있다. 하지만 해시함수는 이런 충돌을 찾는것을 계산적으로 어렵기 때문에 가능하지 않다.
A - 4. ECB에서 서로 다른 블럭의 평문이 다른 값일 때, 같은 암호문인 C1, C2가 나오게 되는 경우를 발견한다면 Alice가 항상 이길 수 있다. 이 암호 프로토콜의 문제점은 서로 다른 블럭에서의 같은 내용의 평문이 담겨있다면, 두 블럭은 같은 암호문으로 암호화된다는 문제점이 있다. 이는 정보의 노출 문제이다.

3. 중재 서명방식은 서명자의 서명키가 노출되었을 때 발생하는 문제점을 극복하기 위한 방식이다. 이 문제는 노출되지 않더라도 서명자의 부정행위를 방지할 수도 있다. 15장에서 자세히 소개하는 블록체인 기술은 첨가만 가능한 분산 데이터베이스를 제공하여 준다. 이 기술이 중재 서명방식 대신에 어떻게 사용할 수 있는지 자신의 생각을 제시하시오.

A. 블록체인 기술은 삭제, 수정이 불가능하고 첨가만 가능하기 때문에, 만일 서명자가 서명 시간을 조작하려는 시도 자체가 불가능해진다. 따라서 중재 서명방식을 사용하던 이유인, 서명 시간 조작의 방지를 블록체인 기술의 특징인 첨가만 가능하다는 것이 대신하게 된다.

5. 대칭 암호알고리즘의 복호화 속도를 일부러 느리게 만들었다. 대칭 암호알고리즘 안전성에 어떤 영향을 주는지 설명하시오.

A. 대칭암호 알고리즘의 복호화속도를 일부러 느리게 만든것은, 만일 서로 다른 길이의 평문이 암호화되는 과정에서 서로 다른 시간을 갖게 된다면, 이를 이용하여 평문의 길이를 유추할 수 있다. 따라서 모든 평문을 암호화할 때의 시간을 같게 조정하기 위해서 일부러 느리게 만들어주어 대칭 암호 알고리즘의 안전성을 보장한다.

6. 대칭키 길이가 32비트인 암호알고리즘을 전사공격하는데 하루가 소요된다고 가정하자. 그러면 복호화 속도가 같은 대칭키 길이가 40비트인 암호알고리즘을 전사공격하는데 소요되는 시간을 제시하시오.

A. 대칭키 길이가 32비트인 암호 알고리즘을 전사공격하는데 24시간이 걸린다. 24시간은 60 * 60 * 24 * 1000 = 86400000밀리초이다. 32비트의 경우 2^32가지의 경우의 수가 있기 때문에 하나의 경우를 시도하는데 걸리는 시간은 86400000 / 2^32 밀리초가 된다. 문제에서 40비트의 경우를 물어보았기 때문에 (하나의 경우를 시도하는데 걸리는 시간) * (총 경우의 수)를 계산하면 걸리는 시간을 구할 수 있다. (86400000 / 2^32) * 2^40 = 22118400000 (ms) 이므로 256시간이 걸린다.

7. 전사공격은 가능한 모든 키를 검사하는 것을 말한다. 대표적인 대칭 암호알고리즘인 DES의 키 길이는 56 비트이고, AES의 키 길이는 128비트이다. DES를 전사공격한다는 것은 2^{56}개의 가능한 모든 키를 검사하는 것을 말하며, 평균적으로 2^{55}번 시도를 하면 키를 찾을 수 있다. 전사공격을 실제 하기 위해서는 임의의 키로 복호화하였을 때 해당 복호화가 성공적인지 여부를 공격자가 확인할 수 있어야 한다. 다음 각각에 대해 답변하시오.

   ① 공격자가 소유한 하나의 프로세서가 1초에 10^7개의 복호화를 할 수 있다고 가정하였을 때와 10^{13}개의 복호화를 할 수 있다고 가정하였을 때 DES와 AES를 전사공격하기 위해 소요되는 시간을 계산하시오.

   ② 공격자가 같은 프로세서를 1,000개 보유가 있을 때 전사공격하기 위해 소요되는 시간을 계산하시오.

   실제 컴퓨터 프로그래밍을 이용하여 소요되는 시간을 년, 일, 시간, 분, 초로 제시하시오.

A - 1.

<python code>

```
des_length = 2**56
aes_length = 2**128

p1 = 10**7
p2 = 10**13

Y = 365 * 24 * 60 * 60
D = 24 * 60 * 60
H = 60 * 60
M = 60

# DES & p1
des1 = des_length / p1
print(f"DES & Process1(10^7) : {des1 // Y}년 {(des1 % Y) // D}일 {((des1 % Y) % D) // H}시간 {(((des1
% Y) % D) % H) // M}분 {(((des1 % Y) % D) % H) % M}초")

# DES & p2
des2 = des_length / p2
print(f"DES & Process2(10^13) : {des2 // Y}년 {(des2 % Y) // D}일 {((des2 % Y) % D) // H}시간 {(((des2
% Y) % D) % H) // M}분 {(((des2 % Y) % D) % H) % M}초")

# aes & p1
aes1 = aes_length / p1
print(f"AES & Process1(10^7) : {aes1 // Y}년 {(aes1 % Y) // D}일 {((aes1 % Y) % D) // H}시간 {(((aes1
% Y) % D) % H) // M}분 {(((aes1 % Y) % D) % H) % M}초")

# DES & p1
aes2 = aes_length / p2
print(f"AES & Process2(10^13) : {aes2 // Y}년 {(aes2 % Y) // D}일 {((aes2 % Y) % D) // H}시간 {(((aes2
% Y) % D) % H) // M}분 {(((aes2 % Y) % D) % H) % M}초")
```

<result>

```
DES & Process1(10^7) : 228.0년 179.0일 23.0시간 50.0분 3.7927932739257812초
DES & Process1(10^13) : 0.0년 0.0일 2.0시간 0.0분 5.759403792793819초
AES & Process1(10^7) : 1.0790283070806014e+24년 228.0일 18.0시간 37.0분 52.0초
AES & Process1(10^13) : 1.0790283070806015e+18년 244.0일 23.0시간 34.0분 24.0초
```

A - 2.
<python code>

```
des_length = 2**56
aes_length = 2**128

p1 = (10**7) * 1000
p2 = (10**13) * 1000

Y = 365 * 24 * 60 * 60
D = 24 * 60 * 60
H = 60 * 60
M = 60

# DES & p1
des1 = des_length / p1
print(f"DES & Process1(10^7) : {des1 // Y}년 {(des1 % Y) // D}일 {((des1 % Y) % D) // H}시간 {(((des1 % Y) % D) % H) // M}분 {(((des1 % Y) % D) % H) % M}초")

# DES & p2
des2 = des_length / p2
print(f"DES & Process2(10^13) : {des2 // Y}년 {(des2 % Y) // D}일 {((des2 % Y) % D) // H}시간 {(((des2 % Y) % D) % H) // M}분 {(((des2 % Y) % D) % H) % M}초")

# aes & p1
aes1 = aes_length / p1
print(f"AES & Process1(10^7) : {aes1 // Y}년 {(aes1 % Y) // D}일 {((aes1 % Y) % D) // H}시간 {(((aes1 % Y) % D) % H) // M}분 {(((aes1 % Y) % D) % H) % M}초")

# DES & p1
aes2 = aes_length / p2
print(f"AES & Process2(10^13) : {aes2 // Y}년 {(aes2 % Y) // D}일 {((aes2 % Y) % D) // H}시간 {(((aes2 % Y) % D) % H) // M}분 {(((aes2 % Y) % D) % H) % M}초")
```

<result>

DES & Process1(10^7) : 0.0년 83.0일 9.0시간 35.0분 59.40379279386252초
DES & Process2(10^13) : 0.0년 0.0일 0.0시간 0.0분 7.2057594037927934초
AES & Process1(10^7) : 1.0790283070806014e+21년 155.0일 3.0시간 56.0분 48.0초
AES & Process2(10^13) : 1079028307080601.0년 144.0일 15.0시간 57.0분 52.0초