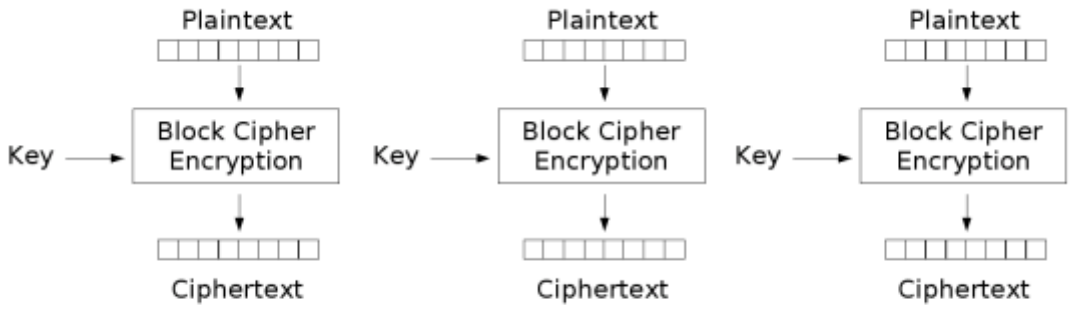
Introduction to ecb, cbc mode and ciphertext stealing

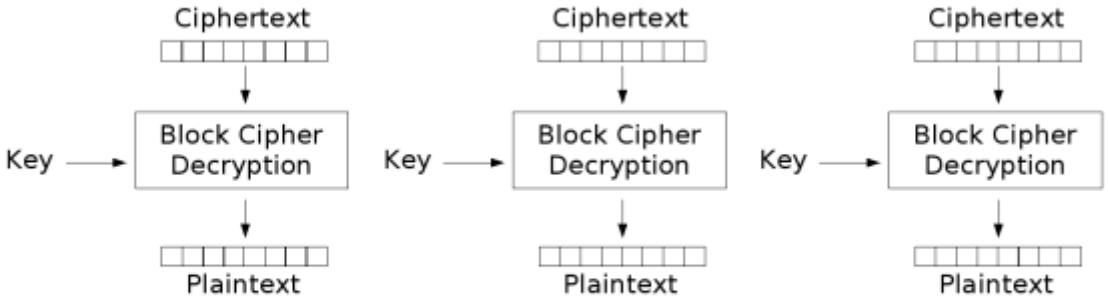
ecb, cbc 모드란?

ecb mode

ECB(electronic codebook) 모드란 가장 간단한 블록 암호 운용 모드이며, 암호문을 블록 사이즈 단위로 나눠 블록마다 같은 key로 암호화를 하는 방식이다.



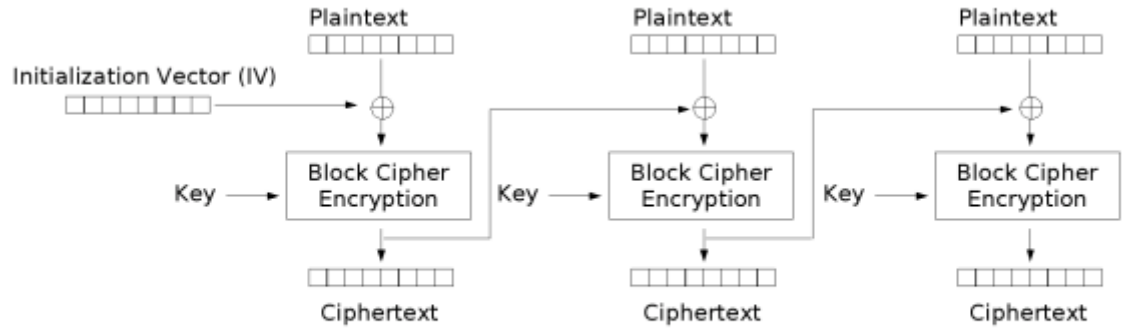
Electronic Codebook (ECB) mode encryption



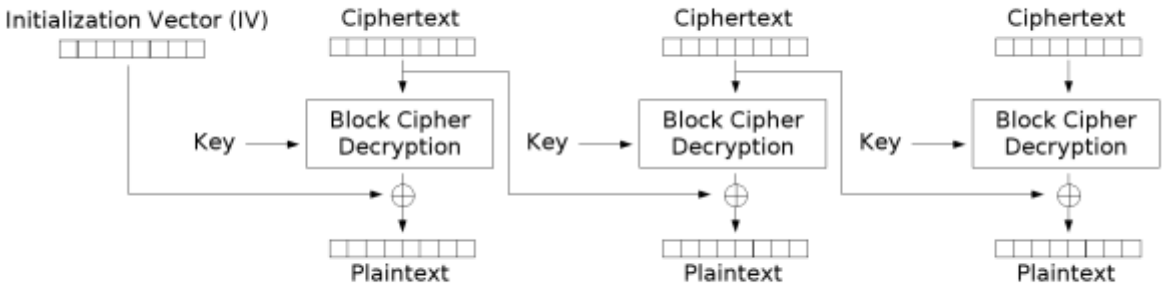
Electronic Codebook (ECB) mode decryption

cbc 모드

CBC(cipher-block chaining)란 초기화 벡터(IV: Initialization Vector)과 첫 평문 블록과 xor 연산을 한 뒤 이를 암호화, 암호화 한 결과를 다음 평문 블록과 xor 후 암호화 하면서 마지막 블록까지 반복하는 운용 모드이다.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

사진 출처: 위키피디아 (https://ko.wikipedia.org/wiki/블록_암호_운용_방식)

문제점

이 두가지 운용 모드로 암/복호화를 진행하기 위한 가장 큰 조건은 바로 모든 블록 사이즈가 같아야 하는 것이다. 암호화 과정에서 블록 사이즈를 맞추기 위해 마지막 블록에 대해서 패딩(padding)을 수행한다.

이게 왜 문제가 될까? 바로 Oracle 관련 공격에 취약하다는 것이다. 이에 대해 알아보기 전에 Oracle에 대해 잠깐 설명을 해보자면

Oracle

사용자와 시스템의 협조적인 관계를 통해 암/복호화를 진행하는 시스템 (~~롤에서 예언자의 영약 떠올린 늙은이는 없거를~~)

Oracle의 한 종류로 Oracle Padding이라는 것이 있는데, 복호화 시스템에 암호문을 넣었을 때, 패딩의 올바름 유무를 보여주는 Oracle을 의미한다.

이제 Oracle이 뭔지는 간략히 알았으니 이를 이용한 ecb, cbc 모드의 공격 방법에 대해서 알아 보자면

ECB Oracle Attack

1. 블록 사이즈를 정의한다.

2. 오프셋을 찾는다.

3. 공격한다.

크게 위의 세 단계로 공격이 이루어지는데 이 시나리오를 설명하자면 아래와 같다.

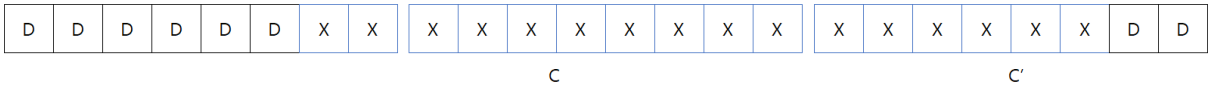
1. 블록 사이즈 정의

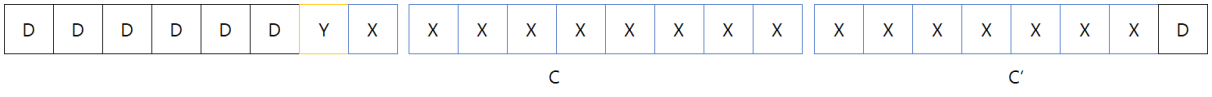
자명하니 넘어간다.

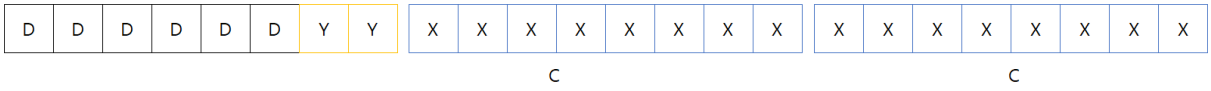
2. 오프셋 찾기

실제로 암호화 되는 평문은 input + secret + padding이 아닌 dummy + input + secret + padding이다. input이 시작하는 위치(오프셋)을 찾기 위해 x * (blocksize * 2) 만큼의 데이터를 넣고 앞에 y를 1 byte씩 추가하면서 찾을 수 있다. (뒤에 두 암호문 블록의 xor 결과가 0이면 오프셋을 찾은 것)

아래의 예제는 블록을 8 byte로 가정한 후 오프셋을 찾는 공격 과정이다.



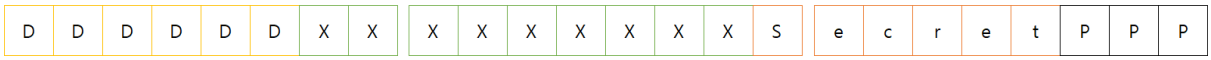


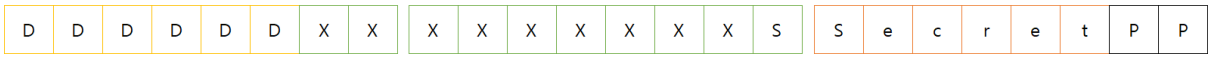


3. ~~신나게~~ 공격한다.

이제 오프셋, 블록 사이즈 다 알았으니 공격을 한다.

위에서 오프셋을 찾는 과정도 블록마다 같은 key를 사용하는 ecb 모드의 특성 때문에 가능했던 방법이인데, 공격 과정도 똑같이 할 수 있다. 암호화 되는 평문이 dummy + input + secret + padding 이라고 가정하면 공격 시나리오는 아래와 같다.





두 번째 블록의 아웃풋은 같게 나올 것이며 xor 연산을 하면 당연히 000000..이렇게 나올 것이다. 이 경우, secret message의 한 바이트를 유추해내는 데 성공했다는 의미이며, 이를 반복하면서 1 byte씩 유추하면서 secret message를 leak할 수 있다.

CBC Oracle Padding Attack

ECB모드에서 봤듯, 각 블록마다 같은 key로 암호화를 수행하는 행위는 위험한 행위이다.

이 단점을 보완하기 위해 cbc모드를 사용하는데, 그렇다고 cbc모드도 과연 안전할까?

바로 Oracle Padding Attack에 취약하다는 점이다.

Oracle이 Padding을 검증하는 방법을 알면 쉽다.

초기화 벡터(IV)의 마지막 1 byte부터 0×00 ~ 0xff까지 브루트포싱 하면서 valid padding값을 유추해내는 식으로 공격해 IV를 leak하는 방법이다.

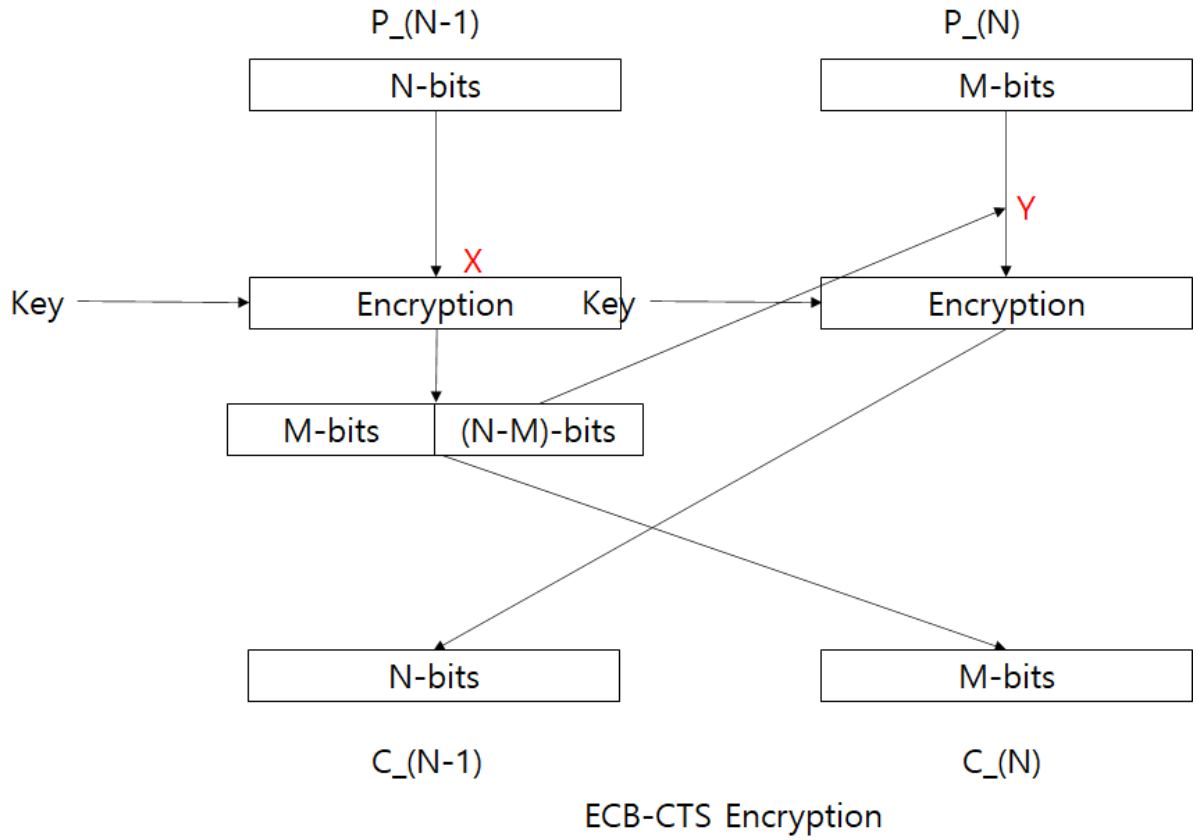
암호문 훔치기 (Ciphertext Stealing 이하 CTS)

cbc모드가 Oracle Padding Attack에 취약하다고 했는데, 이에 대한 대비책 또한 없는 건 아니다.

CTS 기술은 복잡도를 증가시켜 암호문 공격을 어렵게 할 뿐만 아니라 평문과 암호문의 크기가 같아진다는 장점이 있다. 암/복호화 과정은 아래와 같다. ~~CBC 그리고 ECB그리자~~

~~나 귀찮아서 그림이 더럽다.~~

ECB-CTS

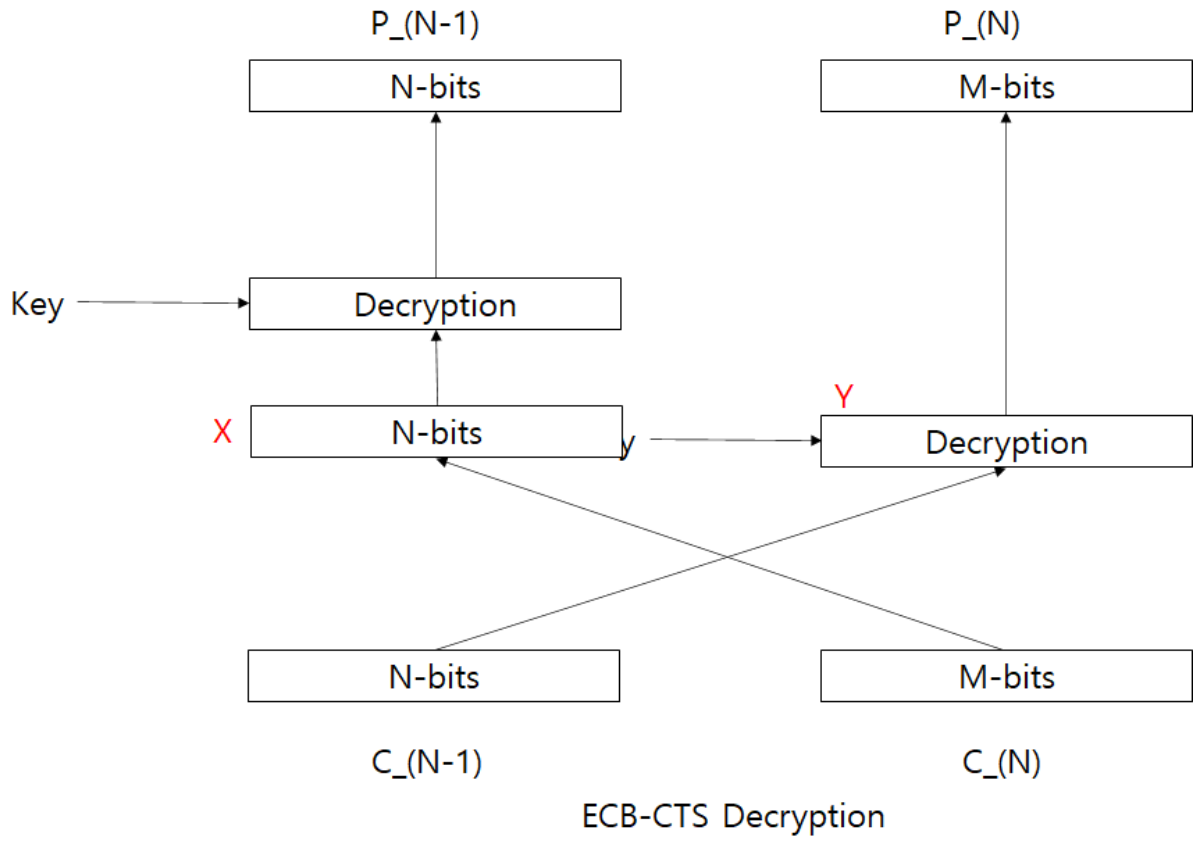


ECB-CTS Encryption

$$X = E(P_{N-1}, key)$$

$$Y = P_N | tail_{N-M}(X)$$

$$C_{N-1} = E(Y, key) \quad C_N = head_M(X)$$

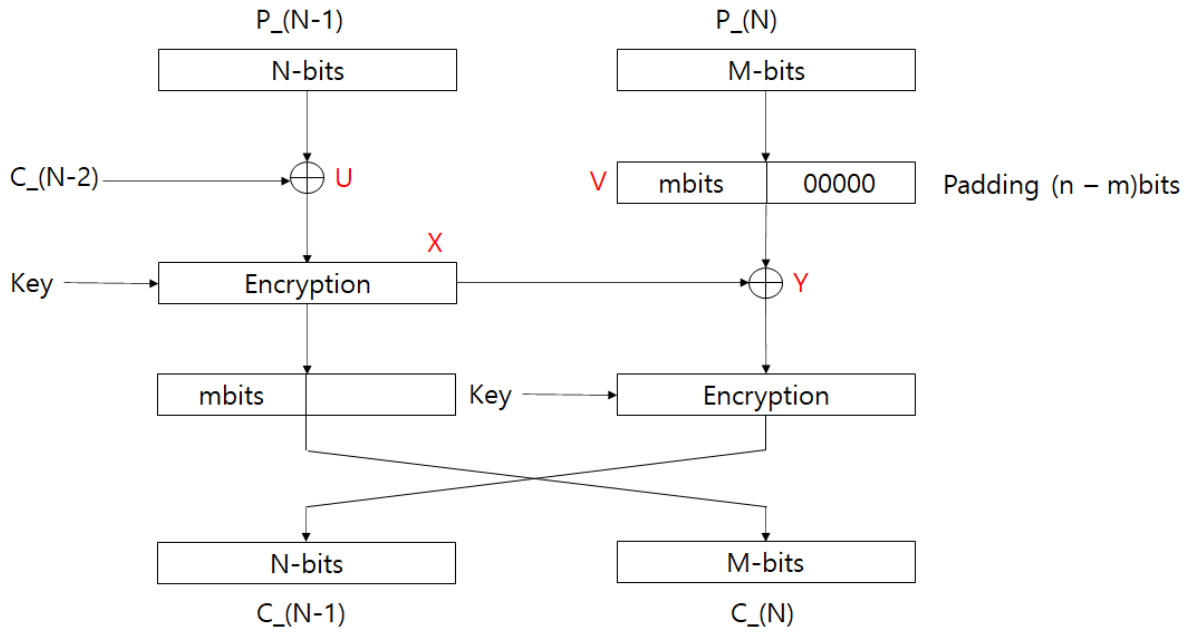


ECB-CTS Decryption

$P_{N-1} = D(X, key)$

$Y = D(C_{N-1}, key) \quad P_N = head_N(Y)$

CBC-CTS



CBC-CTS Encryption

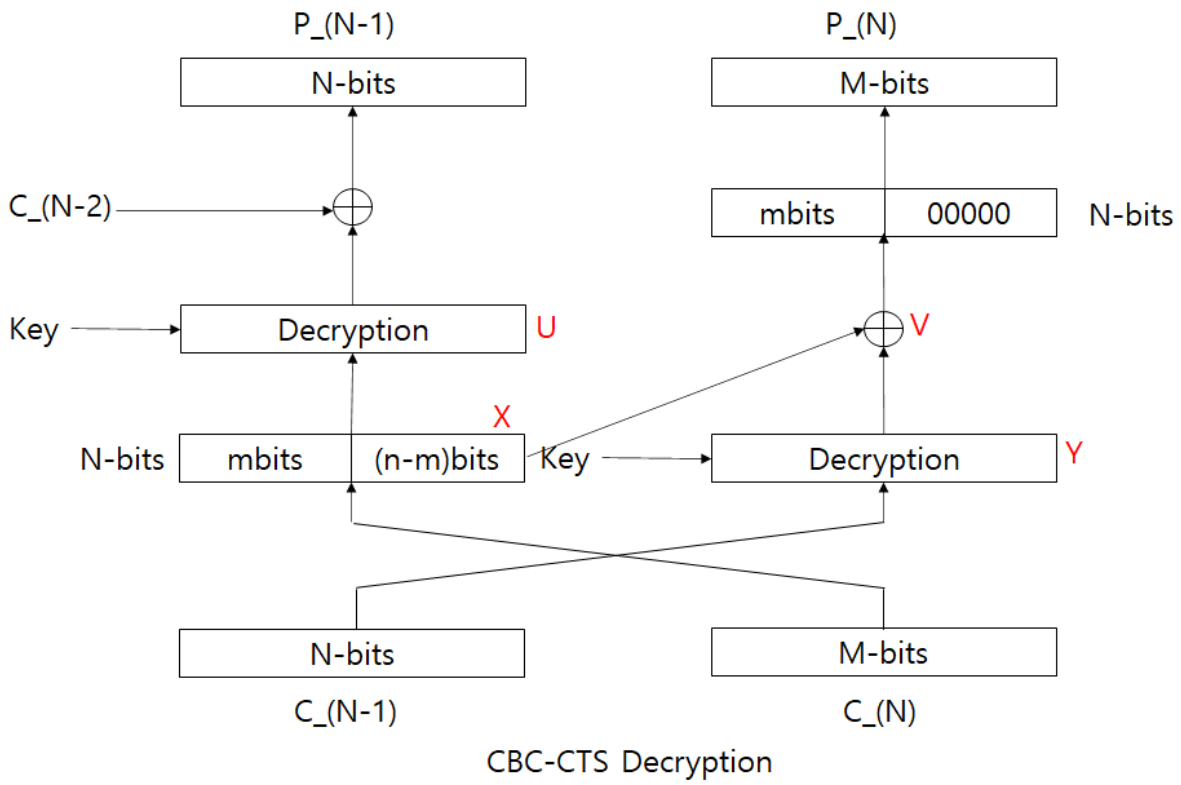
$$U = P_{N-1} \oplus C_{N-2}$$

$$X = E(U, key)$$

$$V = pad(P_N)$$

$$Y = V \oplus X$$

$$C_{N-1} = E(Y, key) \quad C_N = haed(X)$$



CBC-CTS Decryption

$$Y = D(C_{N-1}, key)$$

$$V = X \oplus Y$$

$$U = D(X, key)$$

$$P_{N-1} = U \oplus C_{N-2} \quad P_N = head(V)$$