Block Cipher Mode 분석

(ECB, CBC, CFB, CTR, GCM, CCM)

김경호

https://youtu.be/_mDANPodH8M

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

암호화 모드

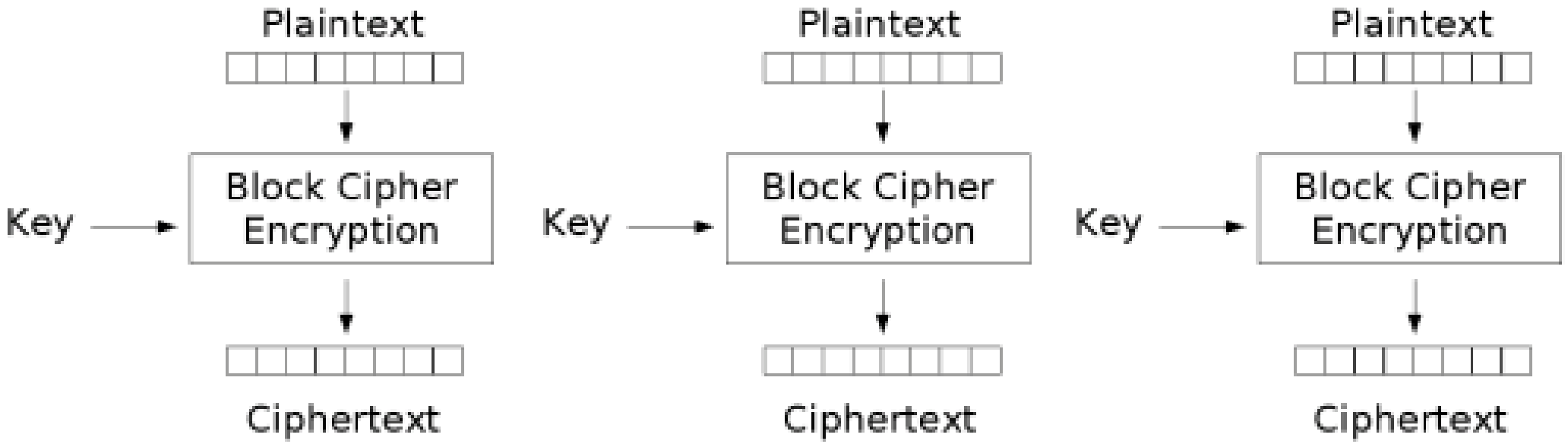
- 연산 방식에 따라 다양한 모드가 존재
  - ECB, CBC, CFB, CTR, GCM, CCM

› **Modes of operation**

Electronic Codebook Mode (ECB), Cipher Block Chaining Mode (CBC), Cipher Feedback Mode (CFB), Counter Mode (CTR), Gallois Counter Mode (GCM), Counter Mode with CBC-MAC (CCM)

ECB(Electronic Code Block)

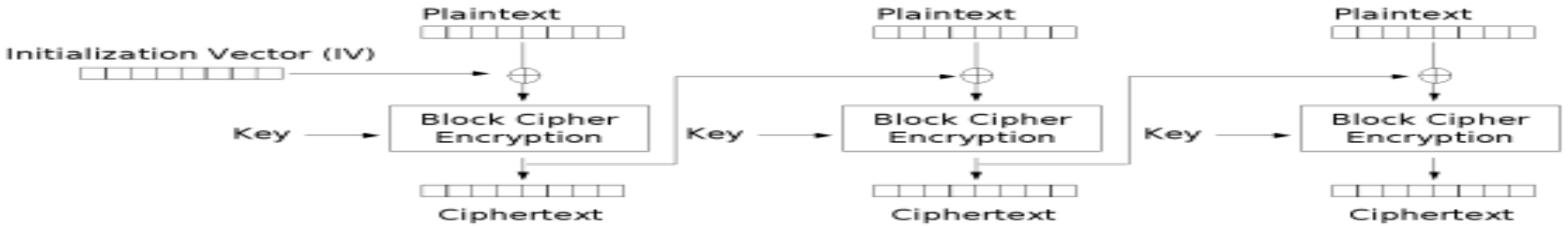
- 가장 Naïve한 암호 알고리즘
- 동일한 키를 사용하기 때문에 한 블록만 해독돼도 모든 데이터 유출



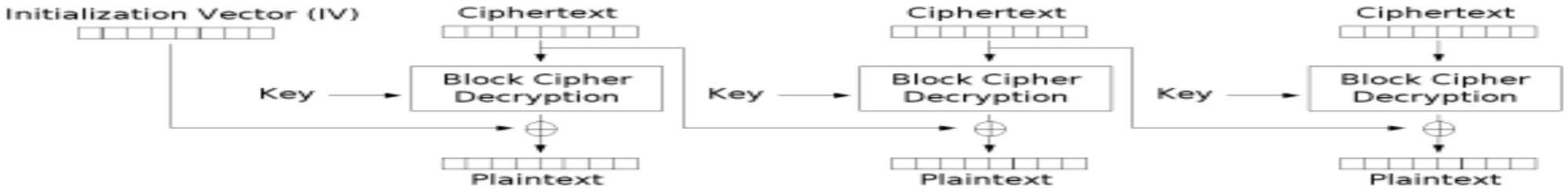
Electronic Codebook (ECB) mode encryption

CBC(Cipher Block Chaining)

- 보안성이 상당히 높은 암호화 모드
- 이전 블록의 암호문이 현재 블록과 XOR 연산 후 암호화
- 이전 블록 연산 값을 사용하기 때문에 병렬처리 불가능



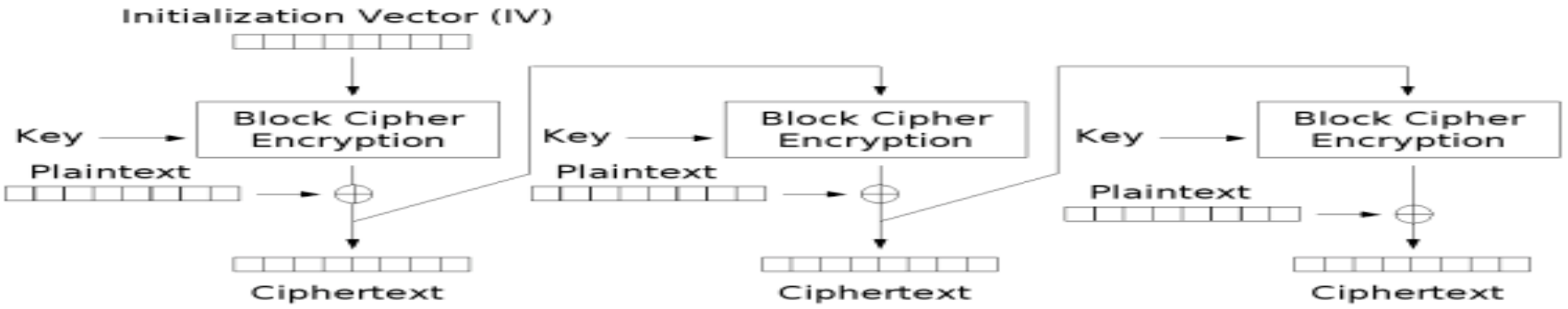
Cipher Block Chaining (CBC) mode encryption



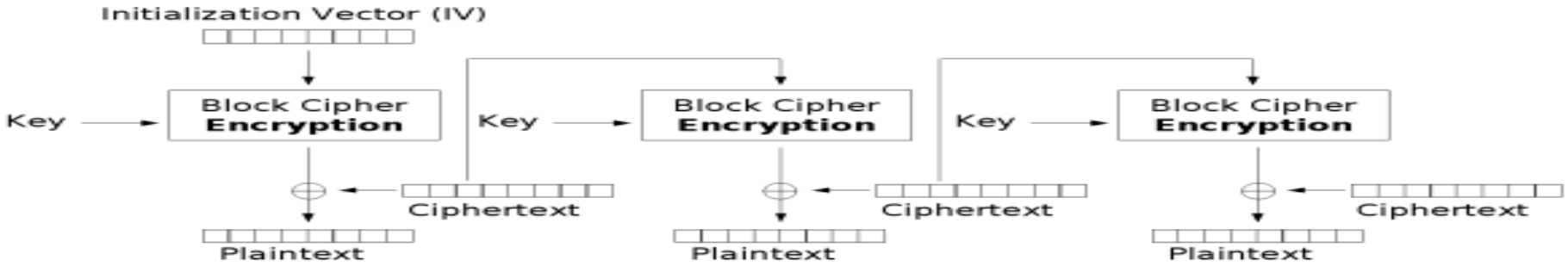
Cipher Block Chaining (CBC) mode decryption

CFB(Cipher FeedBack)

- 평문, 암호문 길이가 같기때문에 패딩 필요 없음
- CBC와 동일하게 이전 블록 값을 사용하기 때문에 병렬처리 X



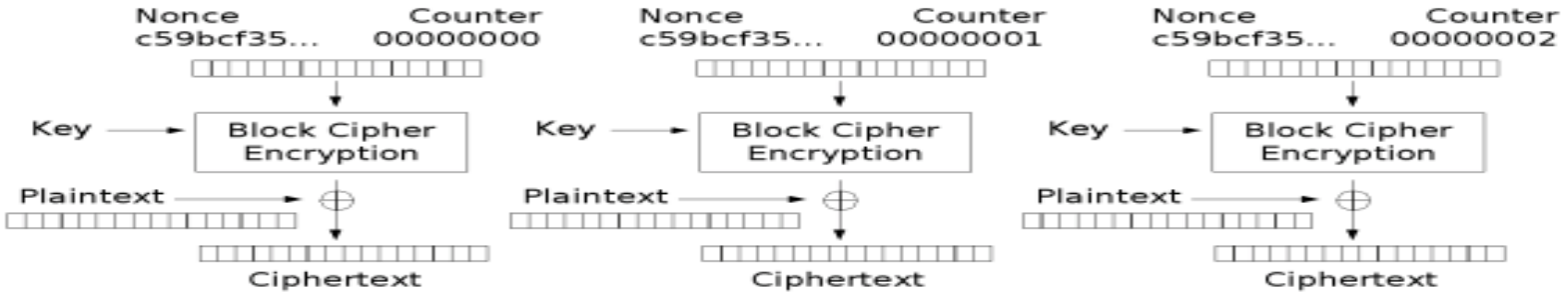
Cipher Feedback (CFB) mode encryption



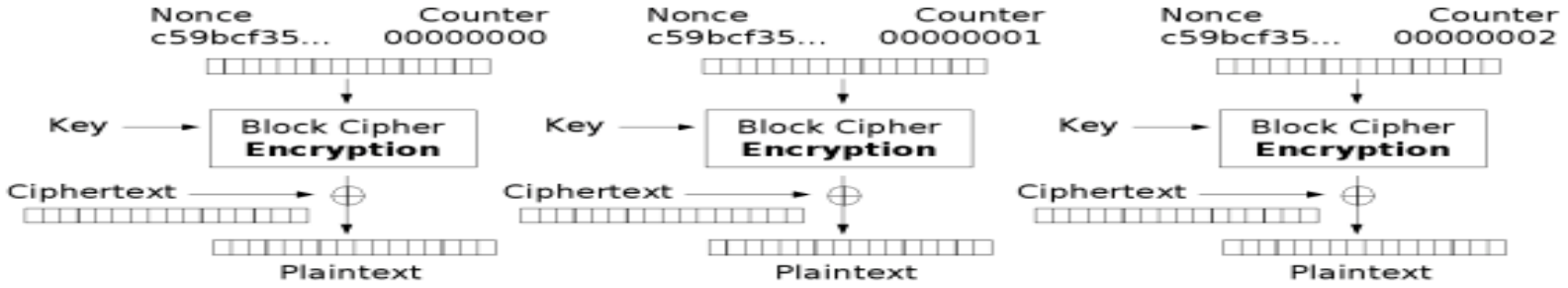
Cipher Feedback (CFB) mode decryption

CTR(Counter)

- Nonce와 Counter로 이루어진 IV 값을 이용한 암호화
- 블록당 1씩 증가하는 Counter 값을 이용
- 병렬 처리가 가능하기때문에 빠른 암호화 속도



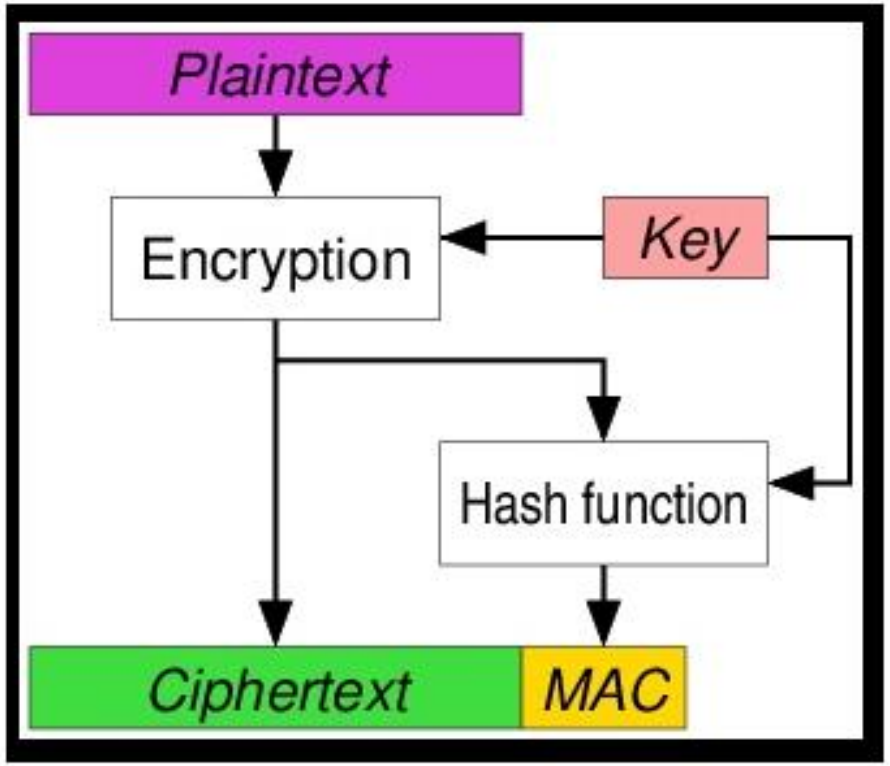
Counter (CTR) mode encryption



Counter (CTR) mode decryption

Authenticated Encryption

- 기존의 암호화 알고리즘에 인증을 추가시킨 암호화
- 암호문의 내용을 조작해서 보낸 경우를 막기 위함
- 기밀성 + 무결성
- MAC 값이 동일한 경우만 인증



CCM(CTR + CBC-MAC)

- 인증값인 T가 동일한 경우 무결성 확보

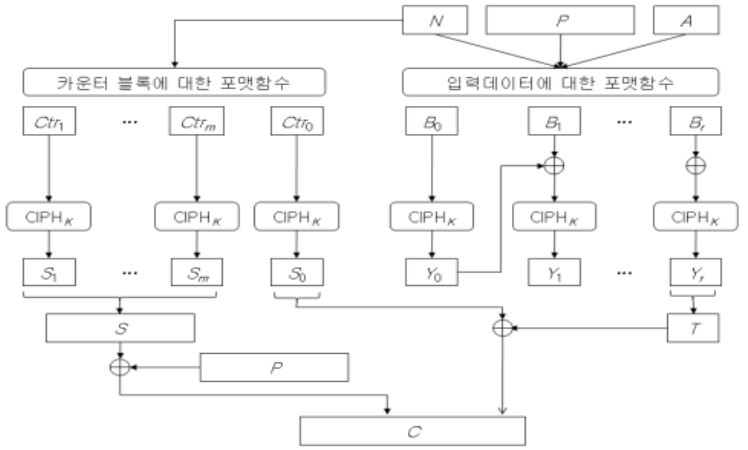
- Input -> Nonce, Payload, A(인증값)
- Output -> Encryption + TAG

- S0와 CBC-MAC 결과물 XOR

Steps:

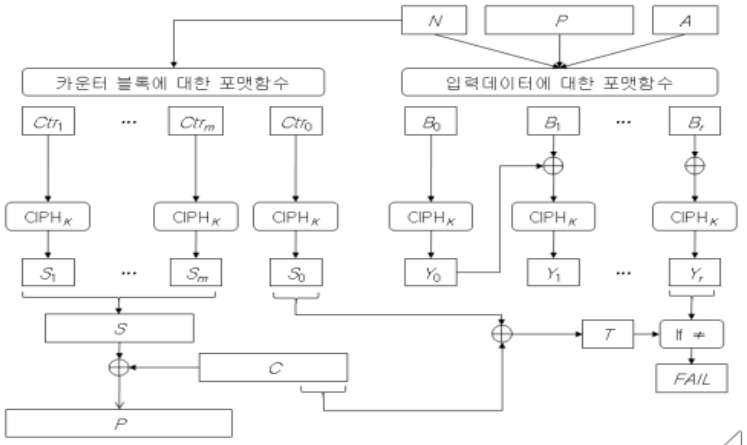
1. Apply the formatting function to (N, A, P) to produce the blocks $B_0, B_1, \ldots, B_r$.
2. Set $Y_0 = \text{CIPH}_K(B_0)$.
3. For $i = 1$ to r, do $Y_i = \text{CIPH}_K(B_i \oplus Y_{i-1})$.
4. Set $T = \text{MSB}_{Tlen}(Y_r)$.
5. Apply the counter generation function to generate the counter blocks $Ctr_0, Ctr_1, \ldots, Ctr_m$, where $m = \lceil Plen/128 \rceil$.
6. For $j = 0$ to m, do $S_j = \text{CIPH}_K(Ctr_j)$.
7. Set $S = S_1 \| S_2 \| \ldots \| S_m$.
8. Return $C = (P \oplus \text{MSB}_{Plen}(S)) \| (T \oplus \text{MSB}_{Tlen}(S_0))$.

(그림 1) CCM 모드 암호화



출처 : TTA

(그림 2) CCM 모드 복호화



출처 : TTA

CCM(CTR + CBC-MAC)

- Format Function
  - 논스, 부가 인증 데이터, 평문 값으로 데이터 블록 생성
  - t = 인증값 T의 바이트 길이
  - Q = 평문의 바이트 길이

비트 색인	7	6	5	4	3	2	1	0
내용	0	Adata	$[(t-2)/2]_3$			$[q-1]_3$		

바이트 색인	0	1 … (15-q)	(16-q) … 15
내용	0	N	Q

01101110 00010011 11010100 10100011 01011101 01110001 10100101 00000000
00000000 00000000 00000000 00000000 00000000 00000000 01000100 00000001

- $0 < a < 2^{16} - 2^8$ -> $a = [a]_{16}$
- $2^{16} - 2^8 \leq a < 2^{32}$ -> $a = 0xFF \| 0xFE \| [a]_{32}$
- $2^{32} \leq a < 2^{64}$ -> $a = 0xFF \| 0xFF \| [a]_{64}$

CCM(CTR + CBC-MAC)

① 블록 B0, B1,…, Br을 생성하기 위하여 (N, P, A)에 포맷 함수 적용

② $Y_0=CIPH_K(B_0)$

③ for i=1, i < r

$Y_i=CIPH_K(B_i \oplus Y_{i-1})$

④ $T=MSB_{Tlen}(Y_r)$

⑤ 카운터 블록 Ctr0, Ctr1,…, Ctrm을 생성하기 위하여 포맷 함수 적용

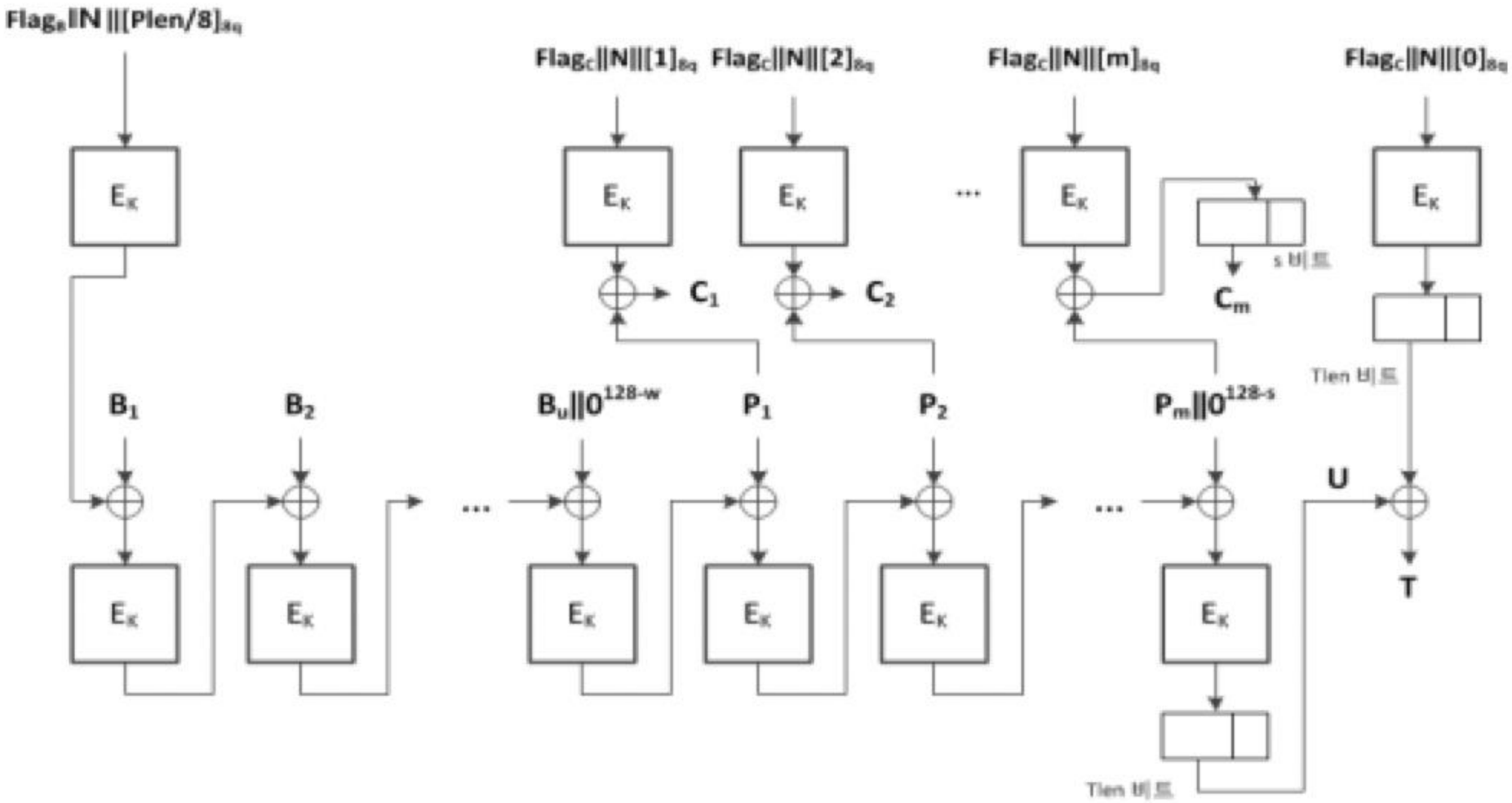
⑥ for j=0, j < m

$S_j=CIPH_K(Ctr_j)$

⑦ S=S1||S2||…||Sm을 계산한다.

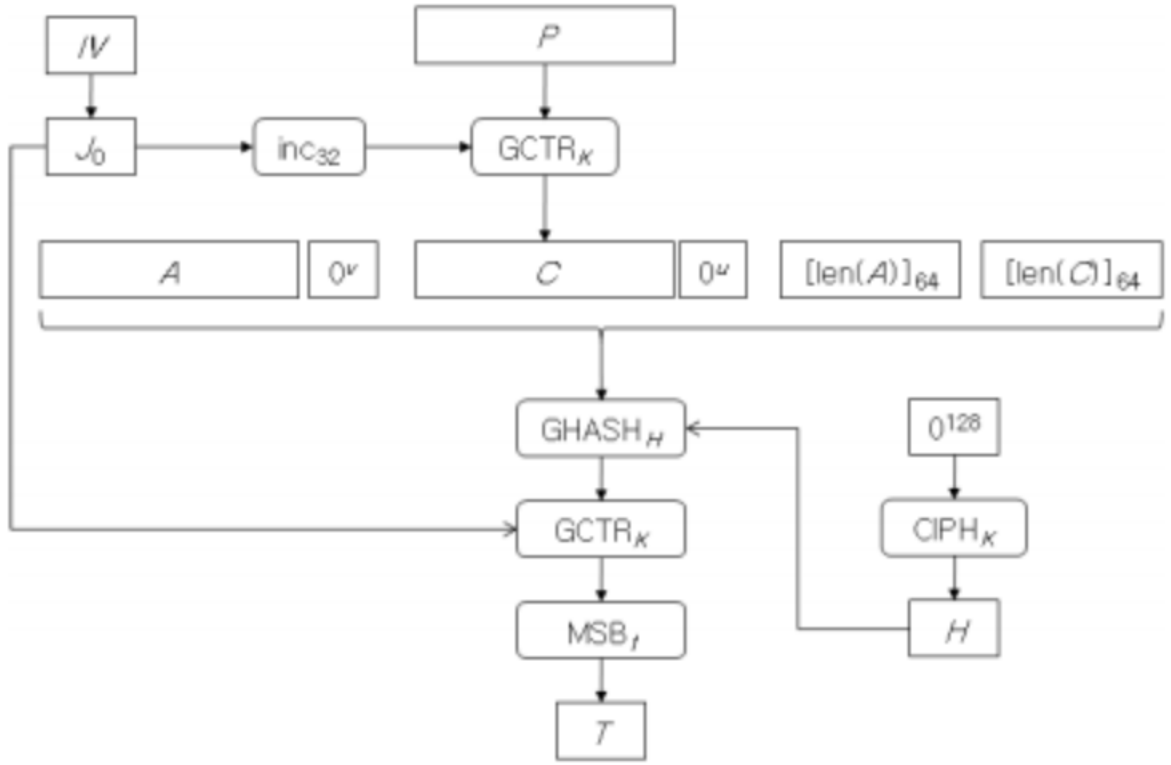
⑧ $C=(P \oplus MSB_{Plen}(S)) \,||\, (T \oplus MSB_{Tlen}(S_0))$

CCM(CTR + CBC-MAC)



GCM(Galois/Counter)

- 유한체 GF(2^{128}) 곱 연산으로 메세지 인증
- 가장 많이 사용하는 Mode
- CTR + Galois
- GHASH, GCTR



GCM(Galois/Counter)

- 유한체 GF(2^{128}) 곱 연산(**X • Y**)

① 블록 X를 비트열 $x_0x_1\ldots x_{127}$로 표기한다.

② $Z_0=0^{128}$와 $V_0=Y$을 초기화한다.

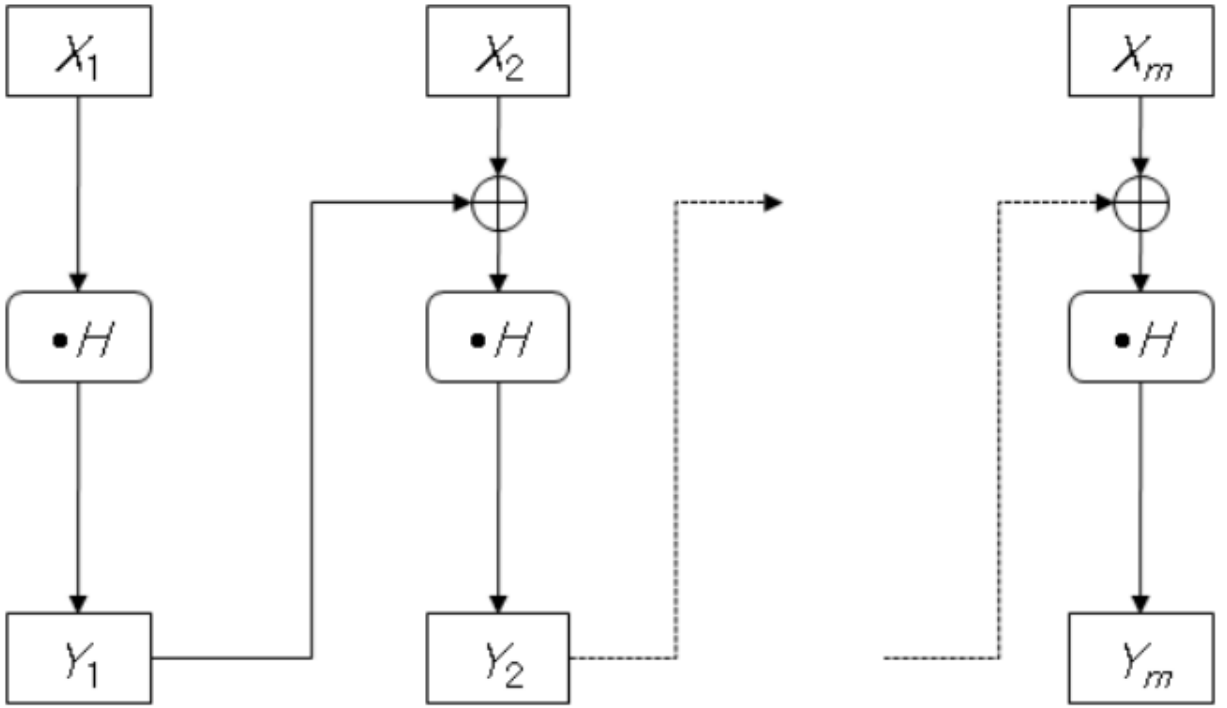
③ $i=0$부터 127까지 블록 Z_{i+1}와 V_{i+1}을 다음과 같이 계산한다.

$$Z_{i+1} = \begin{cases} Z_i & \text{if } x_i=0; \\ Z_i \oplus V_i & \text{if } x_i=1. \end{cases}$$

$$V_{i+1} = \begin{cases} V_i \gg 1 & \text{if } \mathrm{LSB}_1(V_i)=0; \\ (V_i \gg 1) \oplus R & \text{if } \mathrm{LSB}_1(V_i)=1. \end{cases}$$

GCM(Galois/Counter)

- GHASH 함수
  - $Y_0 = 0^{128}$
  - $Y_i=(Y_{i-1} \oplus X_i) \cdot H$



GCM(Galois/Counter)

① $H = CIPH_K(0^{128})$

② if $len(IV)=96$, $J_0=IV||0^{31}||1$
if $len(IV)\neq 96$, $J_0=GHASH_H(IV||0^{s+64}||[len(IV)]_{64})$

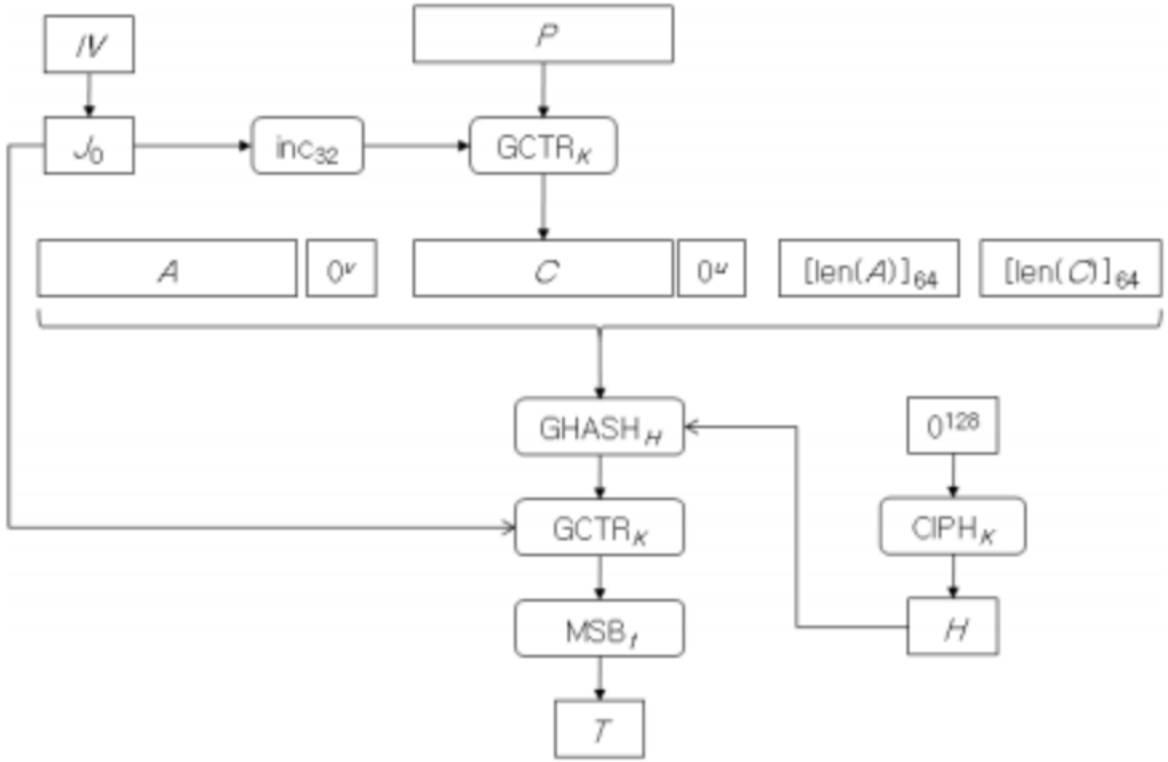
③ $C=GCTR_K(inc_{32}(J_0),P)$

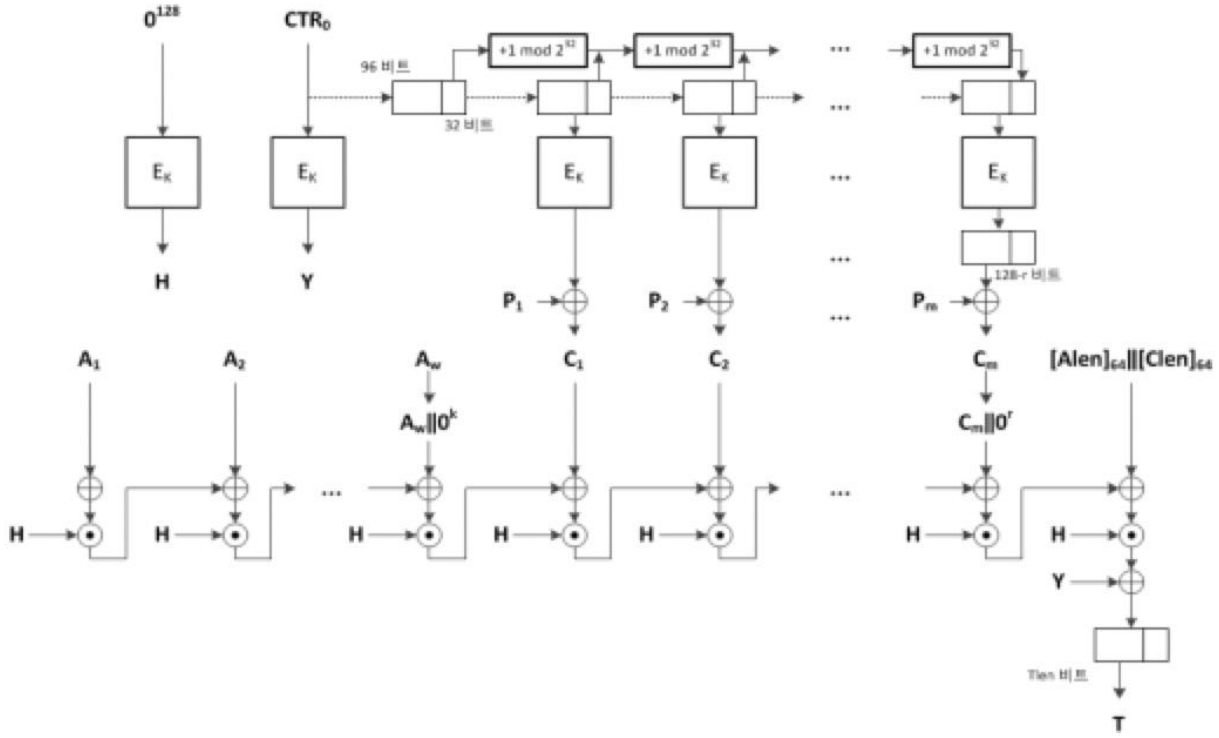
④ $S = GHASH_H(A \,||\, 0^v \,||\, C \,||\, 0^u \,||\, [len(A)]_{64} \,||\, [len(C)]_{64})$

⑤ $T=MSB_{Tlen}(GCTR_K(J_0,S))$

⑥ return (C,T)

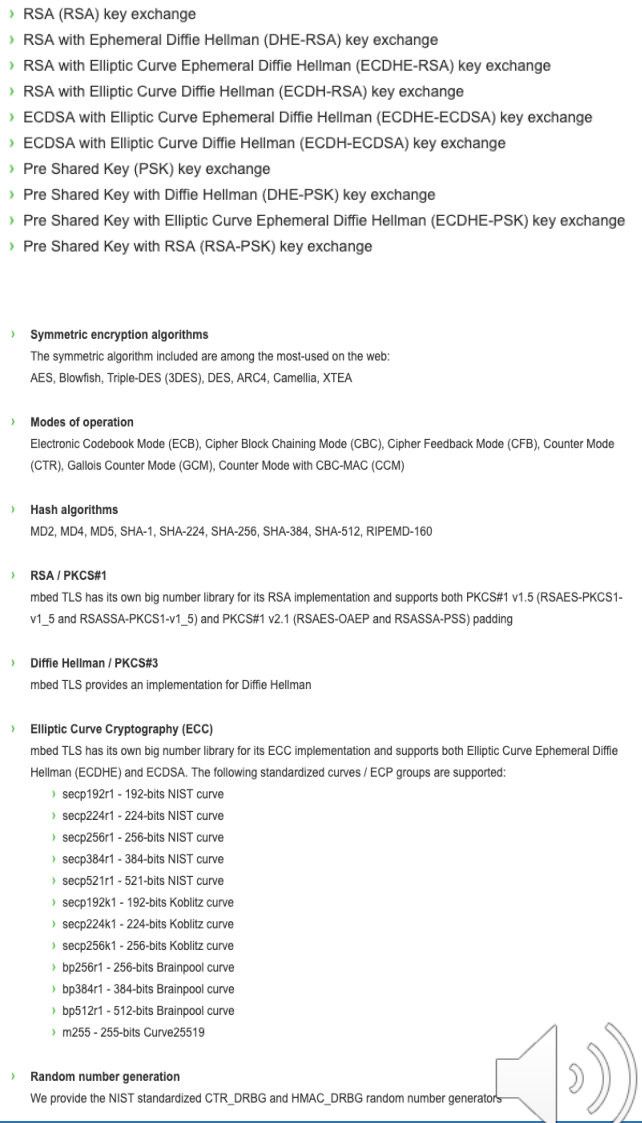
GCM(Galois/Counter)





mbed TLS란?

- PolarSSL에서 mbed TLS로 이름 변경
- OpenSSL 보다 사용이 쉬움
- SSL(Secure Sockets Layer)
  - 안전한 네트워크 통신을 위한 암호화 및 인증 표준화 기술
- AES 뿐만 아니라 다양한 암호화 라이브러리 제공



- RSA (RSA) key exchange
- RSA with Ephemeral Diffie Hellman (DHE-RSA) key exchange
- RSA with Elliptic Curve Ephemeral Diffie Hellman (ECDHE-RSA) key exchange
- RSA with Elliptic Curve Diffie Hellman (ECDH-RSA) key exchange
- ECDSA with Elliptic Curve Ephemeral Diffie Hellman (ECDHE-ECDSA) key exchange
- ECDSA with Elliptic Curve Diffie Hellman (ECDH-ECDSA) key exchange
- Pre Shared Key (PSK) key exchange
- Pre Shared Key with Diffie Hellman (DHE-PSK) key exchange
- Pre Shared Key with Elliptic Curve Ephemeral Diffie Hellman (ECDHE-PSK) key exchange
- Pre Shared Key with RSA (RSA-PSK) key exchange

- **Symmetric encryption algorithms**
  The symmetric algorithm included are among the most-used on the web:
  AES, Blowfish, Triple-DES (3DES), DES, ARC4, Camellia, XTEA
- **Modes of operation**
  Electronic Codebook Mode (ECB), Cipher Block Chaining Mode (CBC), Cipher Feedback Mode (CFB), Counter Mode (CTR), Gallois Counter Mode (GCM), Counter Mode with CBC-MAC (CCM)
- **Hash algorithms**
  MD2, MD4, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, RIPEMD-160
- **RSA / PKCS#1**
  mbed TLS has its own big number library for its RSA implementation and supports both PKCS#1 v1.5 (RSAES-PKCS1-v1_5 and RSASSA-PKCS1-v1_5) and PKCS#1 v2.1 (RSAES-OAEP and RSASSA-PSS) padding
- **Diffie Hellman / PKCS#3**
  mbed TLS provides an implementation for Diffie Hellman
- **Elliptic Curve Cryptography (ECC)**
  mbed TLS has its own big number library for its ECC implementation and supports both Elliptic Curve Ephemeral Diffie Hellman (ECDHE) and ECDSA. The following standardized curves / ECP groups are supported:
  - secp192r1 - 192-bits NIST curve
  - secp224r1 - 224-bits NIST curve
  - secp256r1 - 256-bits NIST curve
  - secp384r1 - 384-bits NIST curve
  - secp521r1 - 521-bits NIST curve
  - secp192k1 - 192-bits Koblitz curve
  - secp224k1 - 224-bits Koblitz curve
  - secp256k1 - 256-bits Koblitz curve
  - bp256r1 - 256-bits Brainpool curve
  - bp384r1 - 384-bits Brainpool curve
  - bp512r1 - 512-bits Brainpool curve
  - m255 - 255-bits Curve25519
- **Random number generation**
  We provide the NIST standardized CTR_DRBG and HMAC_DRBG random number generators