

Network Security Essentials

- Chapter_3 공개 키 암호와 메시지 인증(1) -

강 민 채(minchae@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

- 보충

- DES, AES 암호화 과정에서의 대체/치환
- 페이스텔 구조에서의 Sub Key 생성과정
- 페이스텔 구조에서의 F함수
- 스트림 암호와 RC4
- 암호 블록 운용 모드

- 메시지 인증 방법

- 해시 함수

- 메시지 인증 코드

목 차

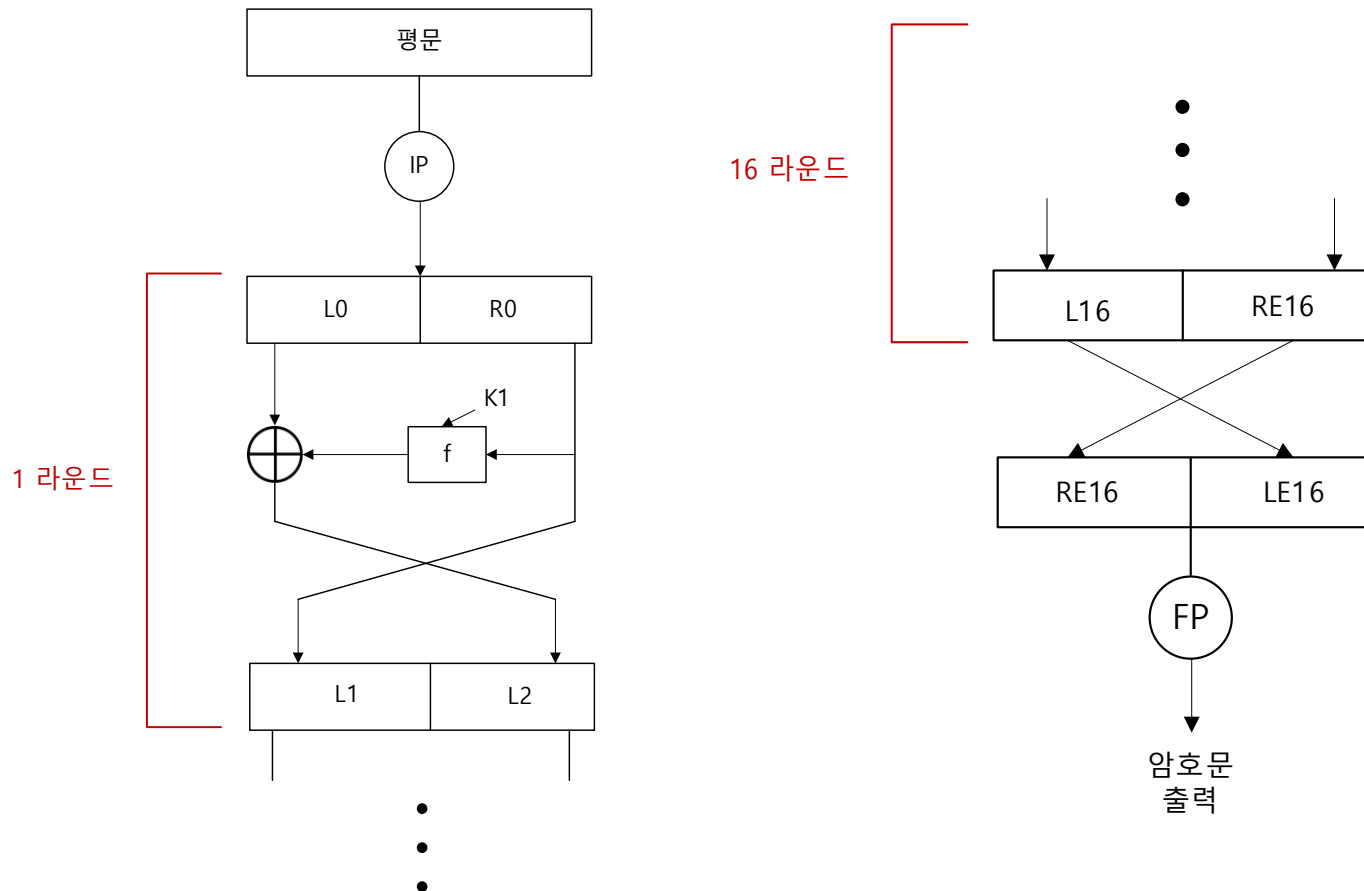
- 보충

- DES, AES 암호화 과정에서의 대체/치환
- 페이스텔 구조에서의 Sub Key 생성과정
- 페이스텔 구조에서의 F함수
- 스트림 암호와 RC4
- 암호 블록 운용 모드

- 메시지 인증 방법
- 해시 함수
- 메시지 인증 코드

보충

- DES(Data Encryption Standard) 암호화 과정
- DES의 전체적인 암호화 과정



보충

- DES(Data Encryption Standard) 암호화 과정
- IP(Initial Permutation)

58번째 비트를 첫번째 자리로 옮기기

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Initial Permutation Table

보충

• DES(Data Encryption Standard) 암호화 과정

• 한 라운드에서의 과정

1. 32비트의 왼쪽 절반 데이터를 오른쪽 데이터에 F함수를 적용한 것과 XOR연산을 한 것으로 대체

$$L_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

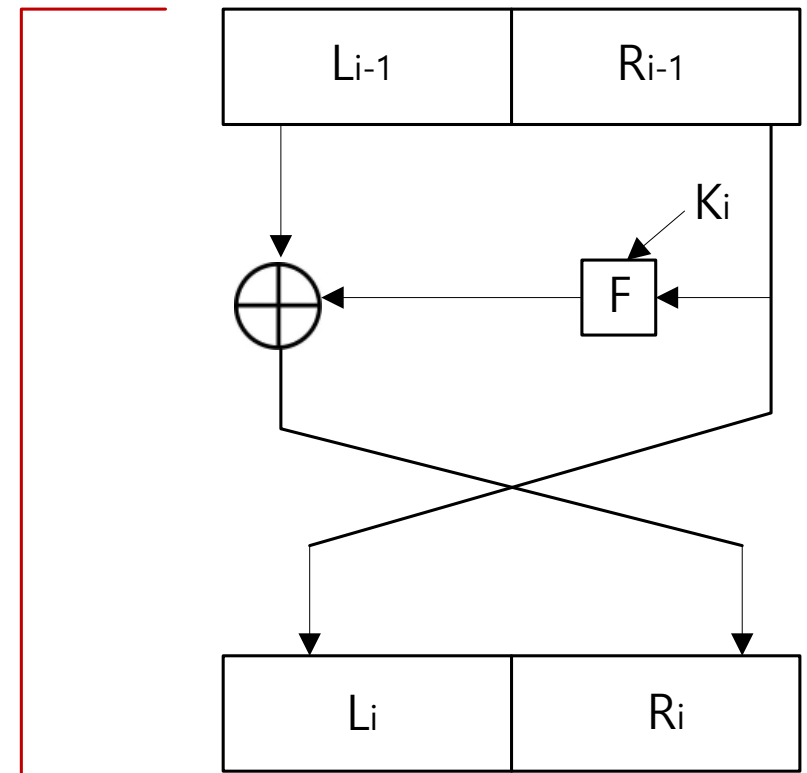
$$R_i = R_{i-1}$$

2. 오른쪽 데이터와 왼쪽 데이터를 치환

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

i 라운드



보충

- DES(Data Encryption Standard) 암호화 과정
- FP(Final Permutation)

40번째 비트를 첫번째 자리로 옮기기

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	09	49	17	57	25

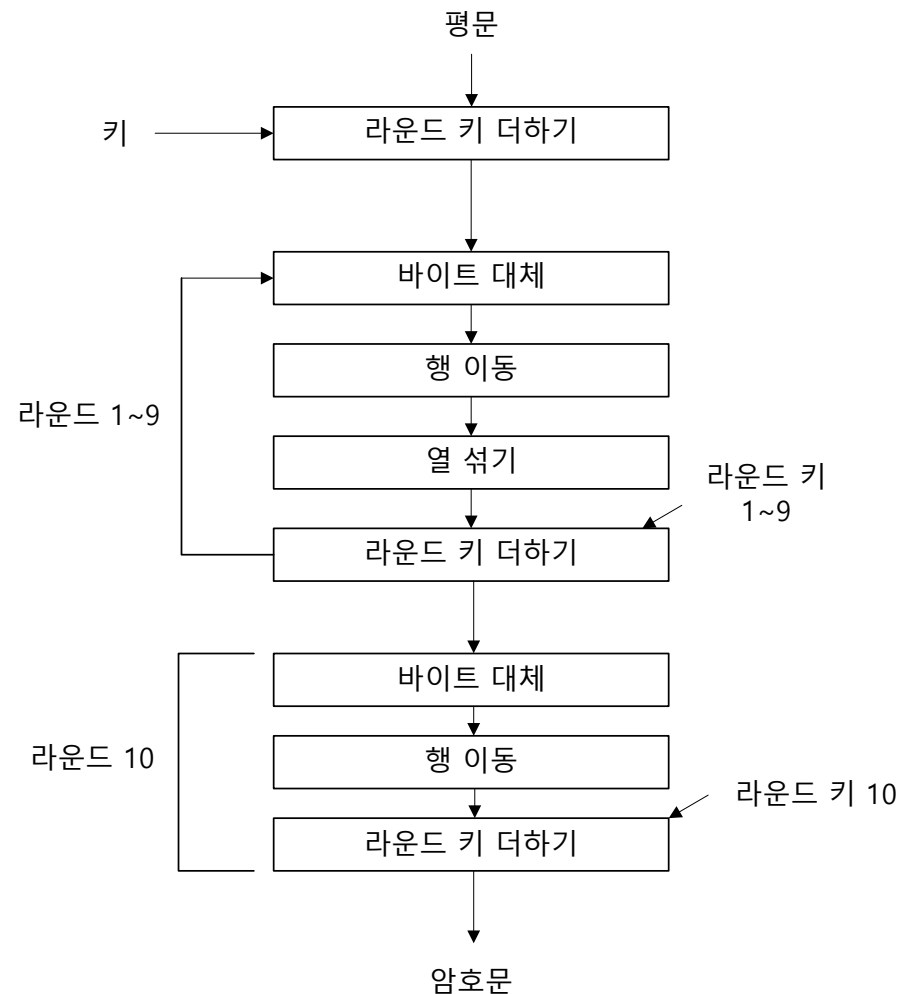
Final Permutation Table

보충

- DES 암호화 과정에서의 대체/치환
 - 대체(Substitution)
 - 한 라운드에서 왼쪽 데이터를 오른쪽 데이터에 라운드 함수를 적용한 것과 XOR 한 것으로 대체
 - 치환(Permutation)
 - 초기 치환(Initial Permutation)
 - 한 라운드에서 양쪽 데이터를 치환
 - 16 라운드까지 끝난 후 양쪽 데이터를 치환
 - 최종 치환(Final Permutation)

보충

- AES(Advanced Encryption Standard) 암호화 과정
- AES의 전체적인 암호화 과정



보충

- AES(Advanced Encryption Standard) 암호화 과정
 - 라운드 키 더하기(Add Round Key)
 - 라운드 키와 현재 블록을 XOR한다

현재 블록

p0	p4	p8	p12
p1	p5	p9	p13
p2	p6	p10	p14
p3	p7	p11	p15

라운드 키

k0	k4	k8	k12
k1	k5	k9	k13
k2	k6	k10	k14
k3	k7	k11	k15



새로운 블록

x0	x4	x8	x12
x1	x5	x9	x13
x2	x6	x10	x14
x3	x7	x11	x15

보충

• AES(Advanced Encryption Standard) 암호화 과정

- 바이트 대체(Substitute Bytes)
 - S-box라는 표를 이용해 바이트 대체

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16


0x19	0xa0	0x9a	0xe9
0x3d	0xf4	0xc6	0xf8
0xe3	0xe2	0x8d	0x48
0xbe	0x2b	0x2a	0x08



0xd4	0xe0	0x88	0xe1
0x27	0xbf	0xb4	0x41
0x11	0x98	0x5d	0x52
0xae	0xf1	0xe5	0x30

보충

- AES(Advanced Encryption Standard) 암호화 과정
 - 행 이동(Shift rows)
 - 첫번째 위치부터 각 행의 위치가 증가되는 수만큼 각 행을 왼쪽으로 이동

0xd4	0xe0	0x88	0xe1		0xd4	0xe0	0x88	0xe1
0x27 ←	0xbf	0xb4	0x41		0xbf	0xb4	0x41	0x27
0x11 ←	0x98	0x5d	0x52		0x5d	0x52	0x11	0x98
0xae ←	0xf1	0xe5	0x30		0x30	0xae	0xf1	0xe5

보충

- AES(Advanced Encryption Standard) 암호화 과정
 - 열 섞기(Mix Columns)
 - 열을 고정 행렬과 섞음

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 0xd4 \\ 0xbf \\ 0x5d \\ 0x30 \end{bmatrix} = \begin{bmatrix} 0xd4 \\ 0x66 \\ 0x81 \\ 0xe5 \end{bmatrix}$$

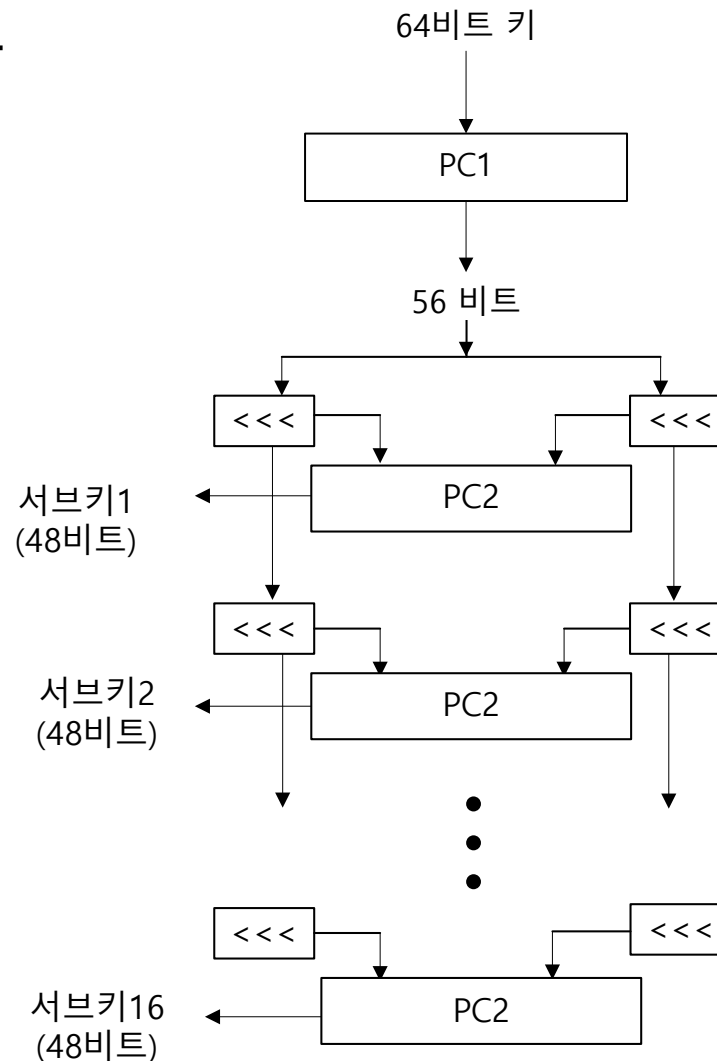
0xd4	0xe0	0x88	0xe1
0xbf	0xb4	0x41	0x27
0x5d	0x52	0x11	0x98
0x30	0xae	0xf1	0xe5

보충

- AES 암호화 과정에서의 대체/치환
 - 대체(Substitution)
 - 바이트 대체
 - S-box라는 표를 이용하여 바이트 단위로 블록을 대체
 - 열 섞기
 - 열에 있는 각 바이트를 고정행렬을 사용하여 대체
 - 라운드 키 더하기
 - 확장된 키와 현재 블록을 비트별로 XOR하여 대체
 - 치환(Permutation)
 - 행 이동
 - 행과 행을 이동(shift)하여 치환

보충

- 페이스텔 암호 구조에서의 서브키 생성 과정
- 전체적인 구조



대칭 암호 원리

- 페이스텔 암호 구조에서의 서브키 생성 과정
- PC1 전치(Permuted Choice One)

1	2	3	4	5	6	7
9	10	11	12	13	14	15
17	18	19	20	21	22	23
25	26	27	28	29	30	31
33	34	35	36	37	38	39
41	42	43	44	45	46	47
49	50	51	52	53	54	55
57	58	59	60	61	62	63



57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	21	4

7비트마다 삽입된 패리티 비트가
제거되어 56비트의 새로운 키 생성

대칭 암호 원리

- 페이스텔 암호 구조에서의 서브키 생성 과정
 - 좌측 순환 이동(Schedule of Left Shifts)
 - 반으로 나눈 각 28비트의 키를 라운드의 수만큼 누적해서 좌측 순환 이동
 - 1, 2, 9, 16 라운드는 1번, 나머지는 2번
 - e.g., 1라운드일때는
before : 1110110 0100101 0100000 0100001
after : 1101100 1001010 1000000 1000011

대칭 암호 원리

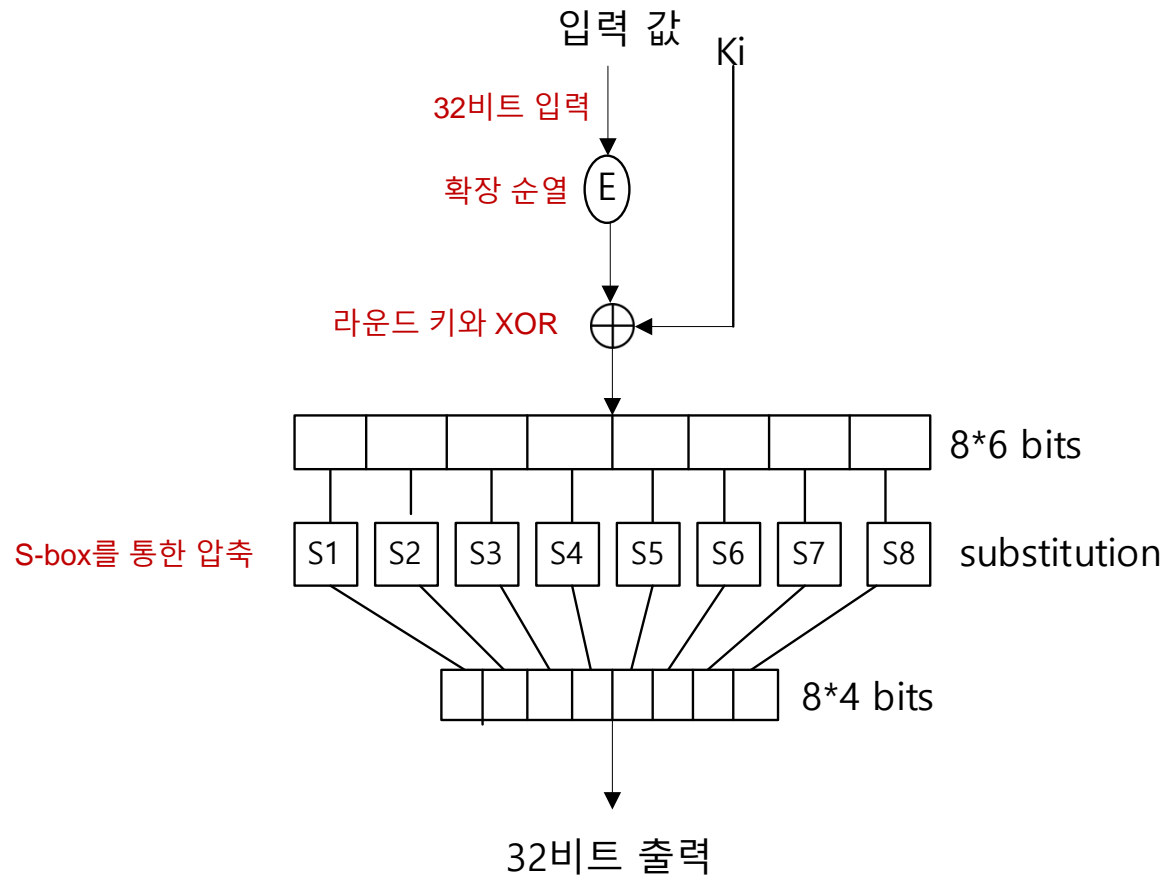
- 페이스텔 암호 구조에서의 서브키 생성 과정
- PC2 전치(Permuted Choice Two)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

PC2 테이블을 통해 56비트 중 48비트가 선택되어 새로운 서브키 생성

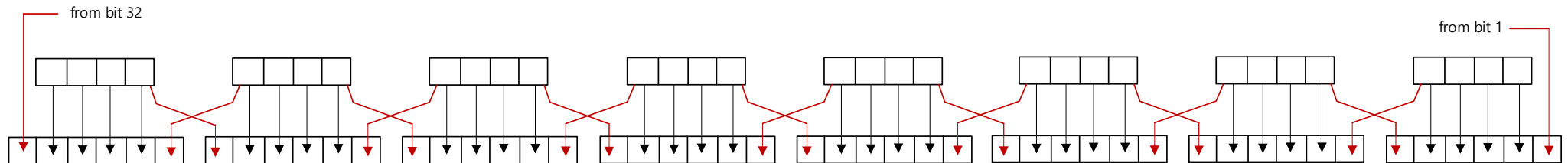
보충

- 페이스텔 구조에서의 라운드 함수
- 라운드 함수의 전체적인 동작 과정



보충

- 페이스텔 구조에서의 라운드 함수
- Expansion Box를 통한 확장
 - E-Box를 통해 8개의 4비트가 각 6비트로 확장됨



보충

- 페이스텔 구조에서의 라운드 함수
 - Substitution Box를 통한 대체와 압축
 - 각 8개의 S-Box를 통해 8개의 6비트에서 새로운 8개의 4비트가 생성됨
 - e.g., S1에서, 입력이 011001일 때
 - 행 = 01 = (10진수) 1
 - 열 = 1100 = (10진수) 12
 - 출력 = 9 (1001)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

보충

- RC4(Rivest Cipher 4)

- 정의

- Ron Rivest가 설계한 바이트 단위로 작동되도록 만들어진 대칭 키 스트림 암호

- 특징

- 랜덤 치환 사용
- 이전에는 여러 프로토콜에서 사용되었으나, 이후 취약점이 발견되어 권장되지 않음
- 비트 단위의 알고리즘보다 소프트웨어적인 실행 속도가 빠름

보충

- RC4(Rivest Cipher 4)

- 과정(1/4)

- $S[0], S[1], \dots, S[255]$ 를 원소로 갖는 256바이트의 배열 S 를 0부터 255까지 오름차순으로 정렬

S

0	1	2	3	4
---	---	---	---	---

...

253	254	255
-----	-----	-----

```
/*Initialization*/  
for i=0 to 255 do  
  S[i] = i;  
  T[i] = K[i mod keylen];
```

보충

- RC4(Rivest Cipher 4)

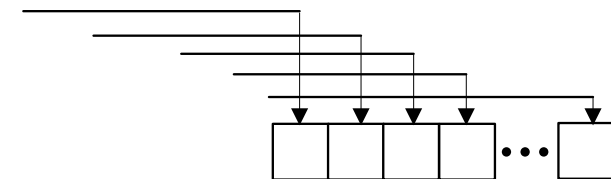
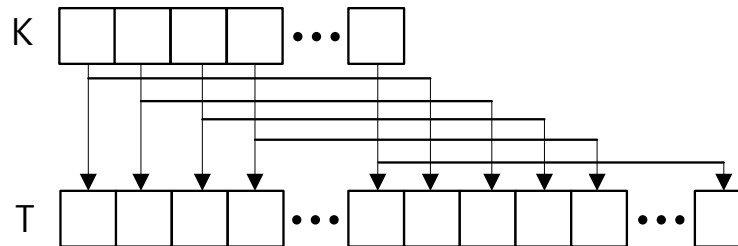
- 과정(2/4)

- 키 K의 길이(keylen)만큼 키 배열 K를 임시 배열 T에 이동하고, $\text{keylen} < 256$ 인 경우, T를 채울 때까지 반복하여 K를 T에 복사

```
/*Initialization*/  
for i=0 to 255 do
```

```
  S[i] = i;
```

```
  T[i] = K[i mod keylen];
```



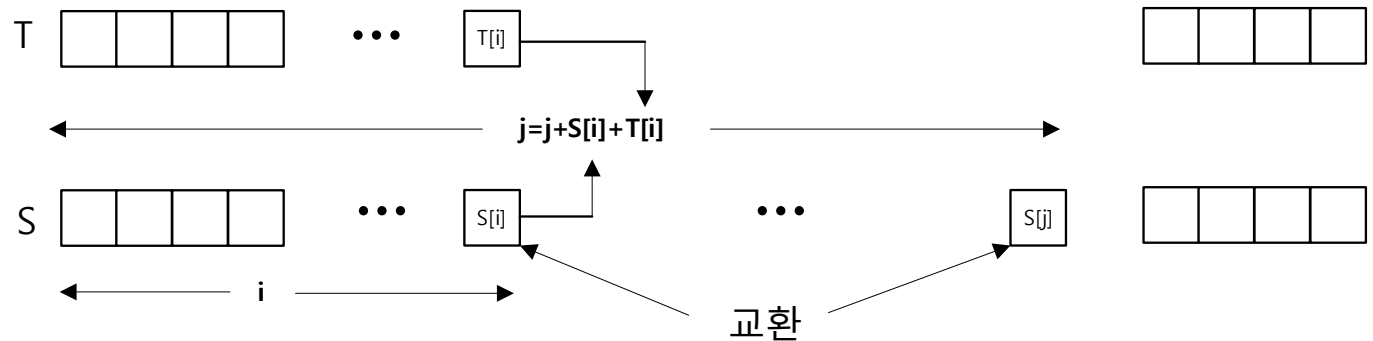
보충

- RC4(Rivest Cipher 4)

- 과정(3/4)

- 각 $S[i]$ 를 $T[i]$ 의 값에 따라서 S 의 다른 바이트 $S[j]$ 와 교환

```
/*Initial Permutation of S*/  
j=0;  
for i = 0 to 255 do  
  j=(j+S[i]+T[i]) mod 256;  
  Swap (S[i], S[j]);
```



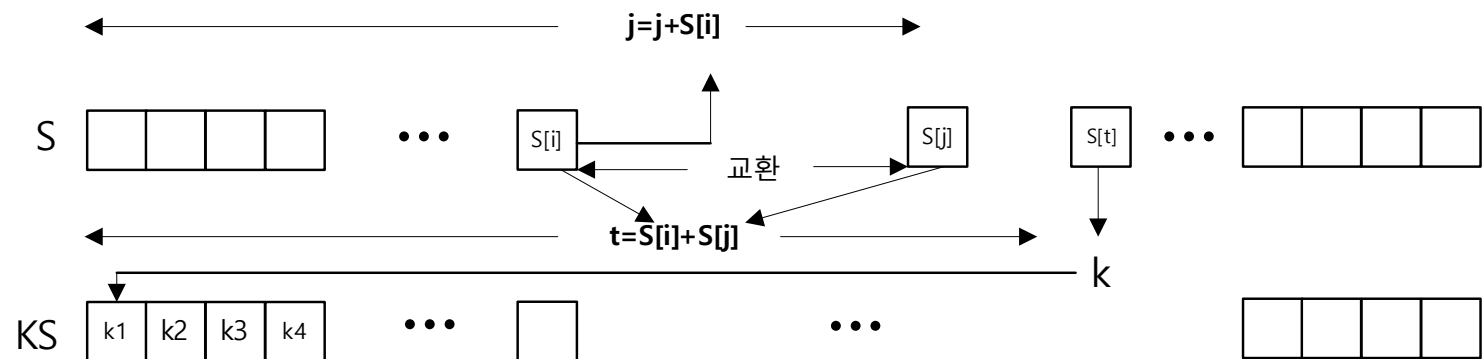
보충

- RC4(Rivest Cipher 4)

- 과정(4/4)

- $S[0]$ 부터 $S[i]$ 에 의해 결정된 $S[j]$ 를 $S[i]$ 와 Swap하고
 $S[i]$ 와 $S[j]$ 의 값에 따라 생성된 키 k 를 평문의 다음 바이트와 XOR

```
/*Stream Generation*/  
i, j=0;  
while (true)  
  i = (i+1) mod 256;  
  j = (j+S[i]) mod 256;  
  Swap (S[i], S[j]);  
  t = (S[i]+S[j]) mod 256;  
  k = S[t];
```

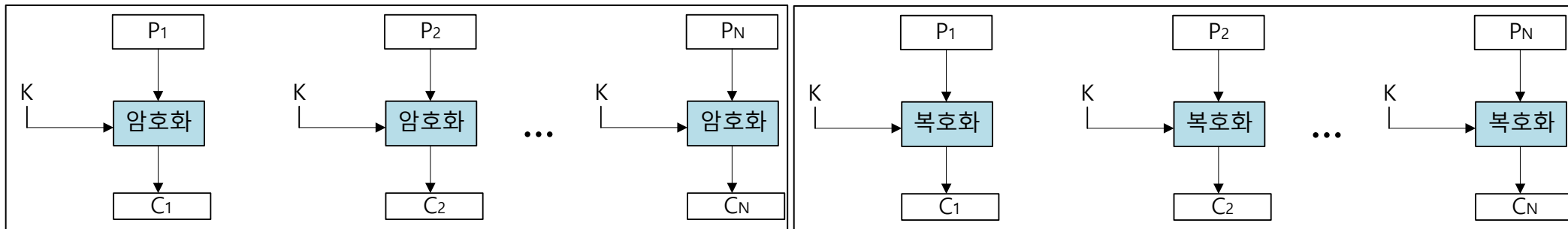


보충

- 암호 블록 운용 모드(Operation Modes)
 - 정의
 - 블록 암호를 다양하게 응용하기 위해 NIST에서 정의한 5가지 운용모드
 - 종류
 - 전자 코드북 모드(ECB, Electronic Codebook Mode)
 - 암호 블록 체인 모드(CBC, Cipher Block Chaining Mode)
 - 암호 피드백 모드 (CFB, Cipher Feedback Mode)
 - 출력 피드백 모드 (OFB, Output Feedback Mode)
 - 카운터 모드(CTR, Counter Mode)

보충

- 암호 블록 운용 모드(Operation Modes)
- ECB(Electronic Codebook) 모드
 - 평문을 같은 크기의 블록으로 나누고 각 블록을 동일한 키로 암호화하는 방식

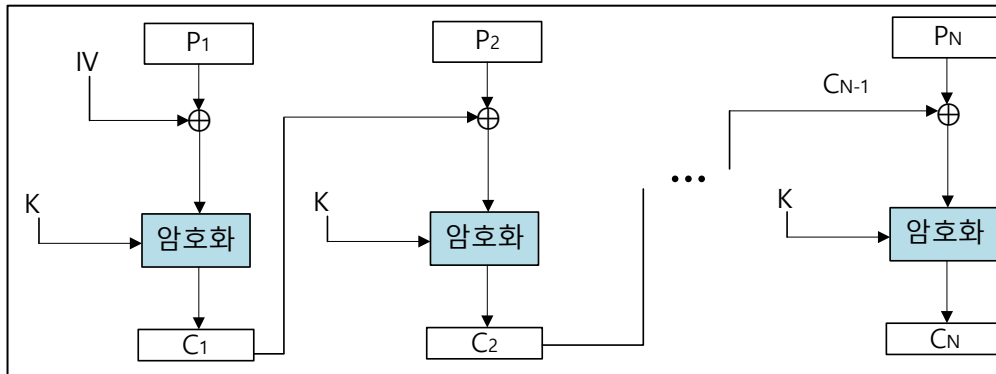


암호화

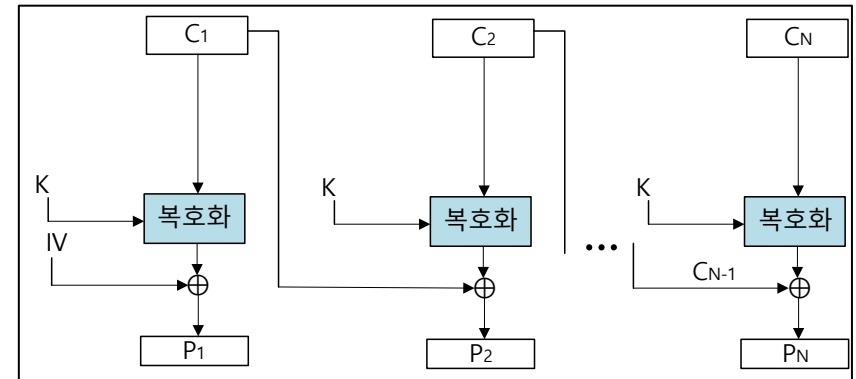
복호화

보충

- 암호 블록 운용 모드(Operation Modes)
- CBC(Cipher Block Chaining) 모드
 - 현재의 평문 블록과 바로 직전의 암호 블록을 XOR한 결과를 알고리즘의 입력으로 사용하는 방식



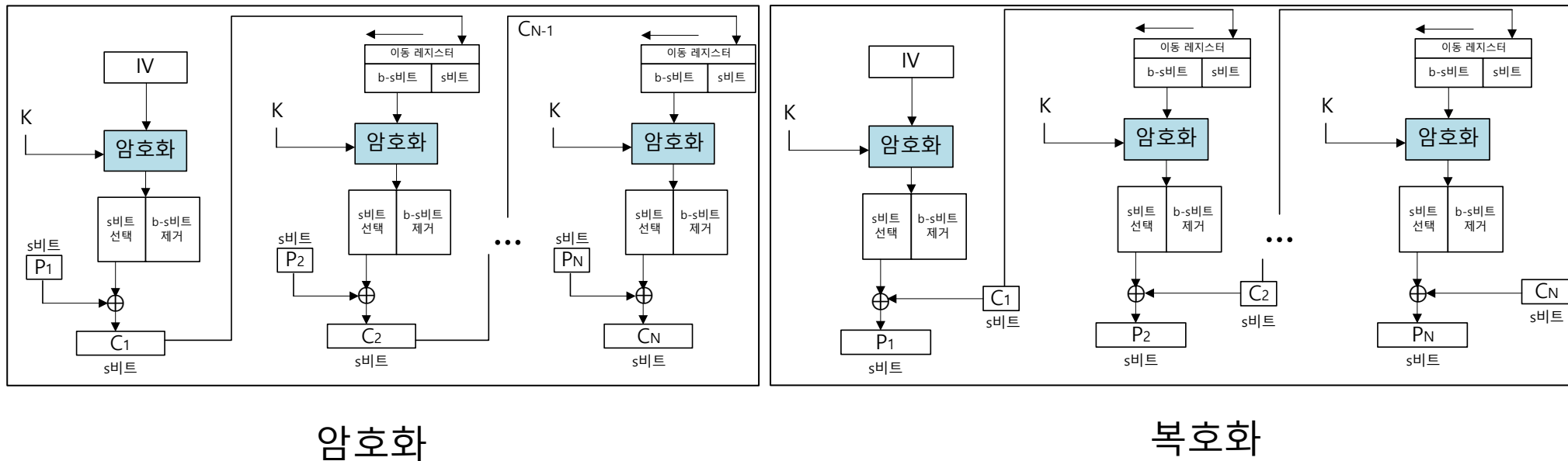
암호화



복호화

보충

- 암호 블록 운용 모드(Operation Modes)
- CFB(Cipher Feedback) 모드
 - 다음 바이트를 암호화 할 때 이전의 암호문을 이용하는 방식

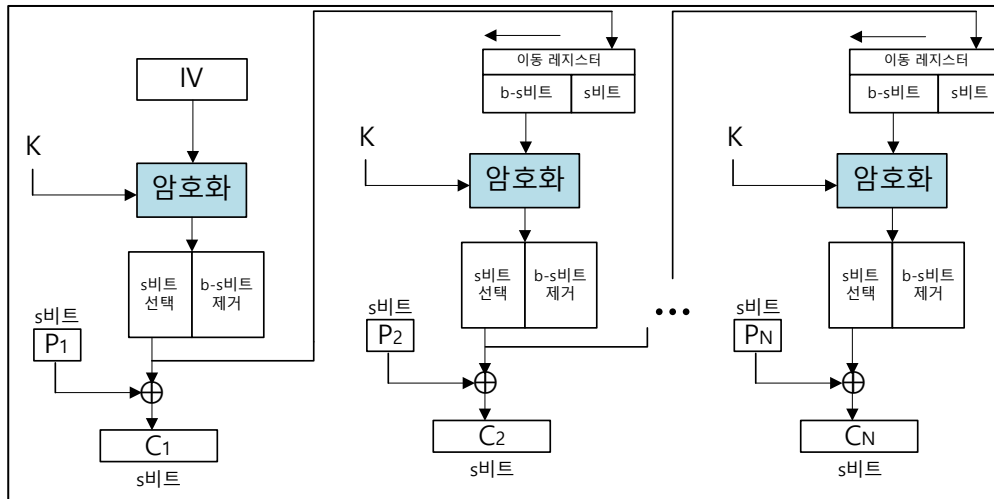


보충

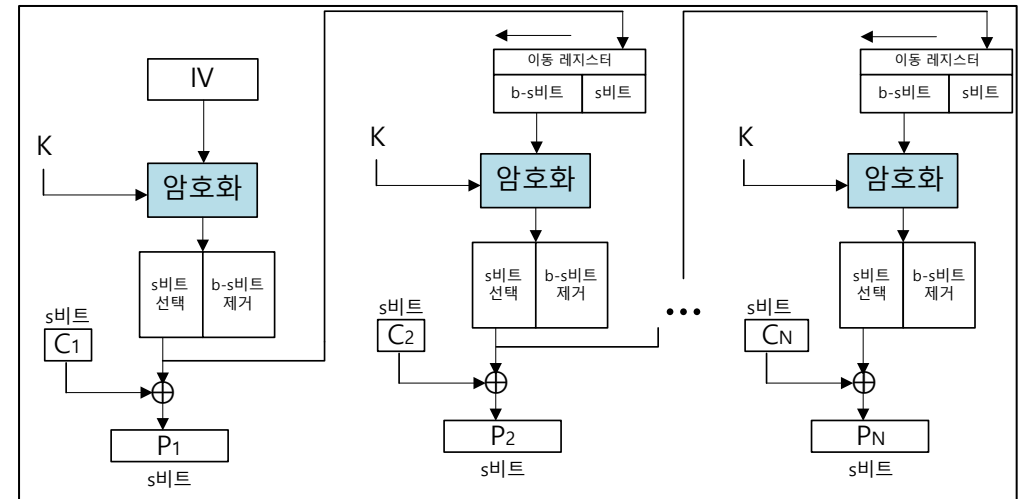
- 암호 블록 운용 모드(Operation Modes)

- OFB(Output Feedback) 모드

- CFB와 비슷하지만, 다음 바이트를 암호화할 때 이전의 암호문을 사용하지 않고, 단순히 이전의 암호화된 결과만을 사용하는 방식



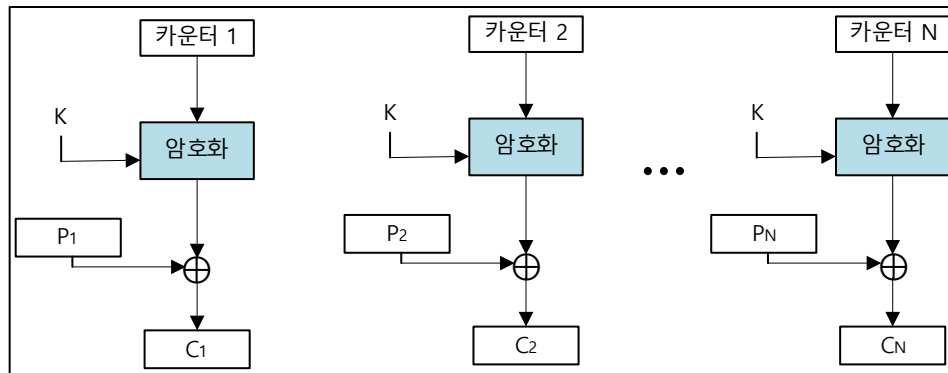
암호화



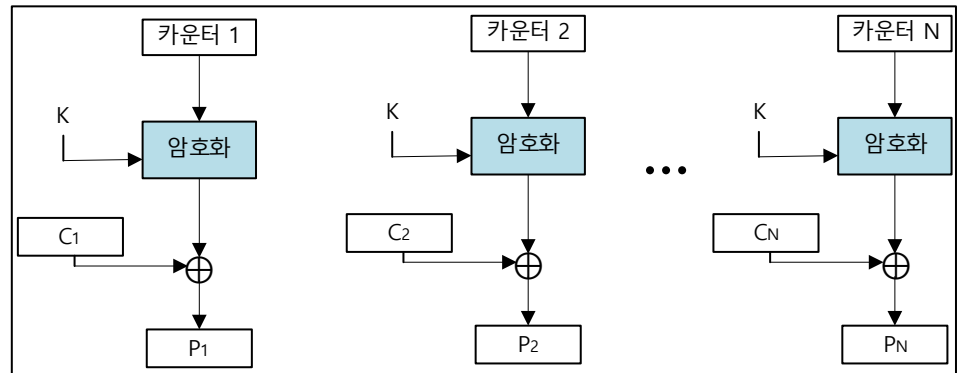
복호화

보충

- 암호 블록 운용 모드(Operation Modes)
- CTR(CounTeR) 모드
 - 각 평문 블록마다 값이 다른 카운터를 암호화하여 평문 블록과 XOR하는 방식



암호화



복호화

보충

- 암호 블록 운용 모드(Operation Modes)
- 운용 모드 비교 표

암호 운용 모드	병렬 처리	초기화 벡터	패딩
ECB	O	X	O
CBC	복호화만	O	O
CFB	복호화만	O	X
OFB	X	O	X
CTR	O	X	X

목 차

- 보충

- DES, AES 암호화 과정에서의 대체/치환
- 페이스텔 구조에서의 Sub Key 생성과정
- 페이스텔 구조에서의 F함수
- 스트림 암호와 RC4
- 암호 블록 운용 모드

- 메시지 인증 방법

- 해시 함수
- 메시지 인증 코드

메시지 인증 방법

- 메시지 인증(Message Authentication)

- 정의

- 통신 양측으로 하여금 메시지의 내용이 전송 도중 불법적으로 변경되지 않고 안전하게 전송됨을 보증하는 것

- 특징

- 메시지의 무결성 보호
 - 메시지 송신자의 검증
 - 부인 방지

메시지 인증 방법

- 메시지 인증(Message Authentication)

- 인증 방법

- 대칭 키 암호를 사용한 인증

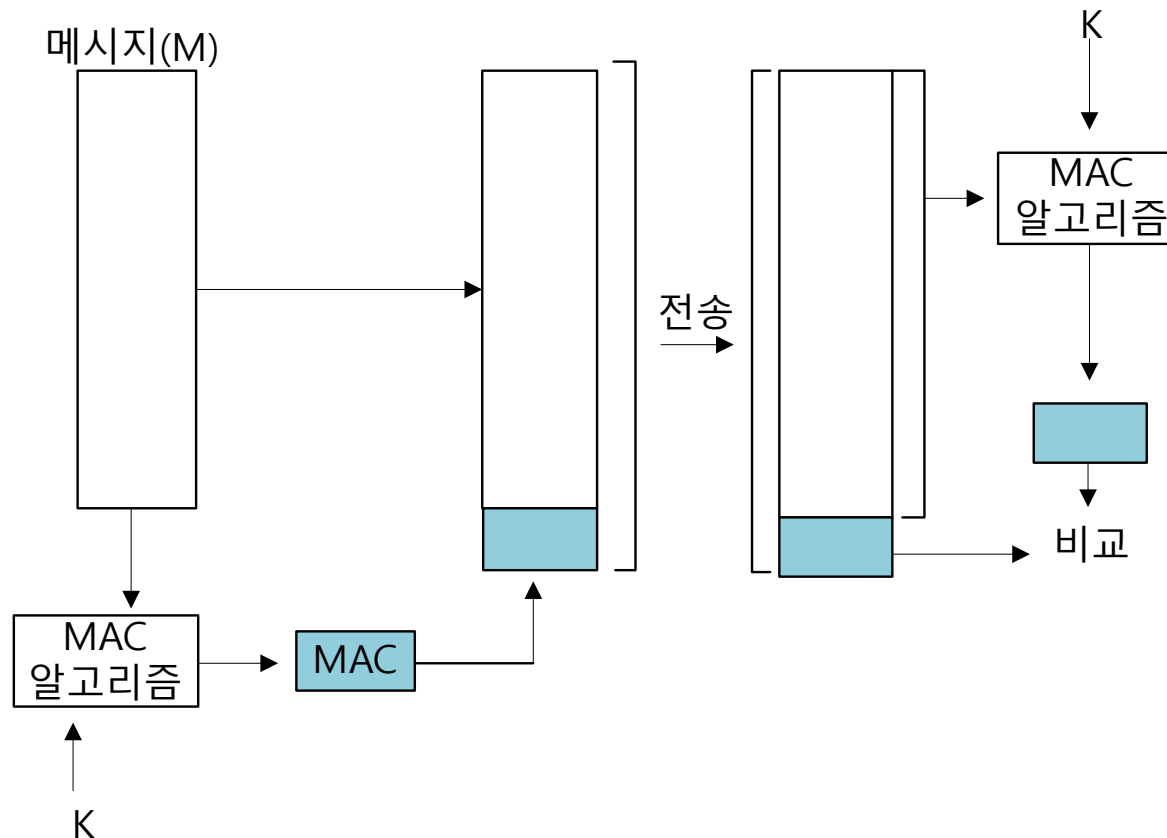
- 송수신자가 공유하는 비밀 키를 이용해 메시지를 암호/복호화할 수 있음
 - 오류 감지 코드를 통해 수신자는 메시지의 변경 여부 확인 가능
 - 순서 번호를 통해 수신자는 메시지의 순서가 맞는지 확인 가능
 - 타임스탬프를 통해 수신자는 메시지의 고의적 시간 지연이 있었는지 확인 가능

- 메시지 암호화 없는 메시지 인증

- 인증 꼬리표를 생성하여 메시지에 붙여서 보냄
 - e.g.,
 - 동일한 메시지를 다수의 수신자에게 브로드캐스팅하는 경우
 - 메시지 교환 시 송수신자 중 한 쪽에 부하가 걸려 메시지 복호화가 힘든 경우
 - 컴퓨터 프로그램을 평문인 채로 인증하는 경우

메시지 인증 방법

- 메시지 암호화 없는 메시지 인증
 - 메시지 인증 코드(MAC, Message Authentication Code)를 이용한 메시지 인증
 $[MAC_M = F(K, M)]$



목 차

- 보충

- DES, AES 암호화 과정에서의 대체/치환
- 페이스텔 구조에서의 Sub Key 생성과정
- 페이스텔 구조에서의 F함수
- 스트림 암호와 RC4
- 암호 블록 운용 모드

- 메시지 인증 방법

- 해시 함수

- 메시지 인증 코드

해시 함수

- 일방향 해시 함수

- 정의

- 특정 입력 데이터에서 고정 길이 값 또는 해시 값을 생성하며, 역으로 수행될 수 없는 알고리즘

- 특징

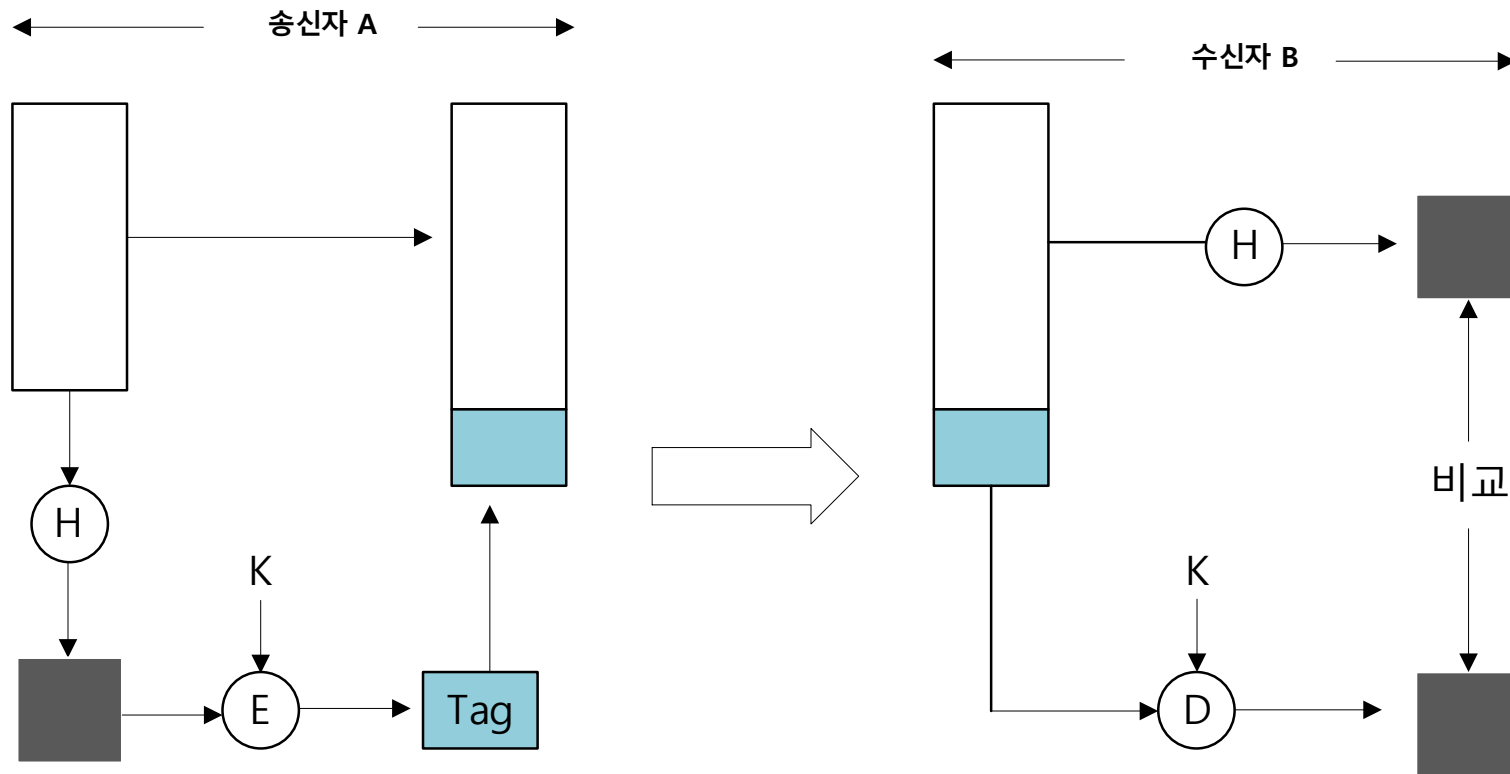
- 비밀 키를 입력으로 사용하지 않음
 - 인증된 상태의 메시지 다이제스트를 메시지와 함께 전송함
 - 데이터의 무결성을 검증하는 데에 사용됨

- 세 가지 인증 방법

- 대칭 키 암호를 사용한 인증
 - 공개 키 암호를 사용한 인증
 - 비밀 값을 사용한 인증

해시 함수

- 일방향 해시 함수
 - 대칭 키 암호를 사용한 인증
 - 송수신자는 공유하는 비밀 키를 사용

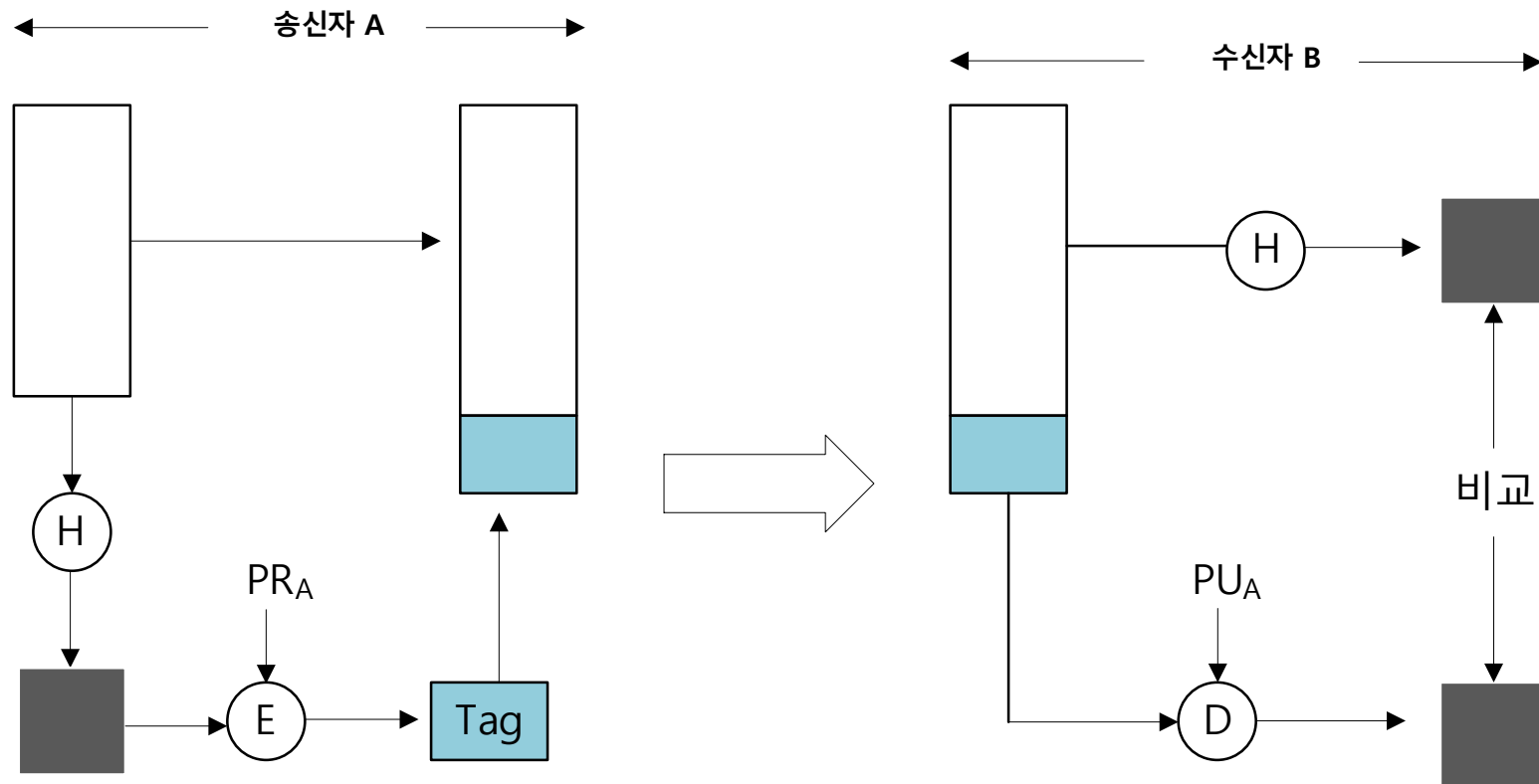


해시 함수

- 일방향 해시 함수

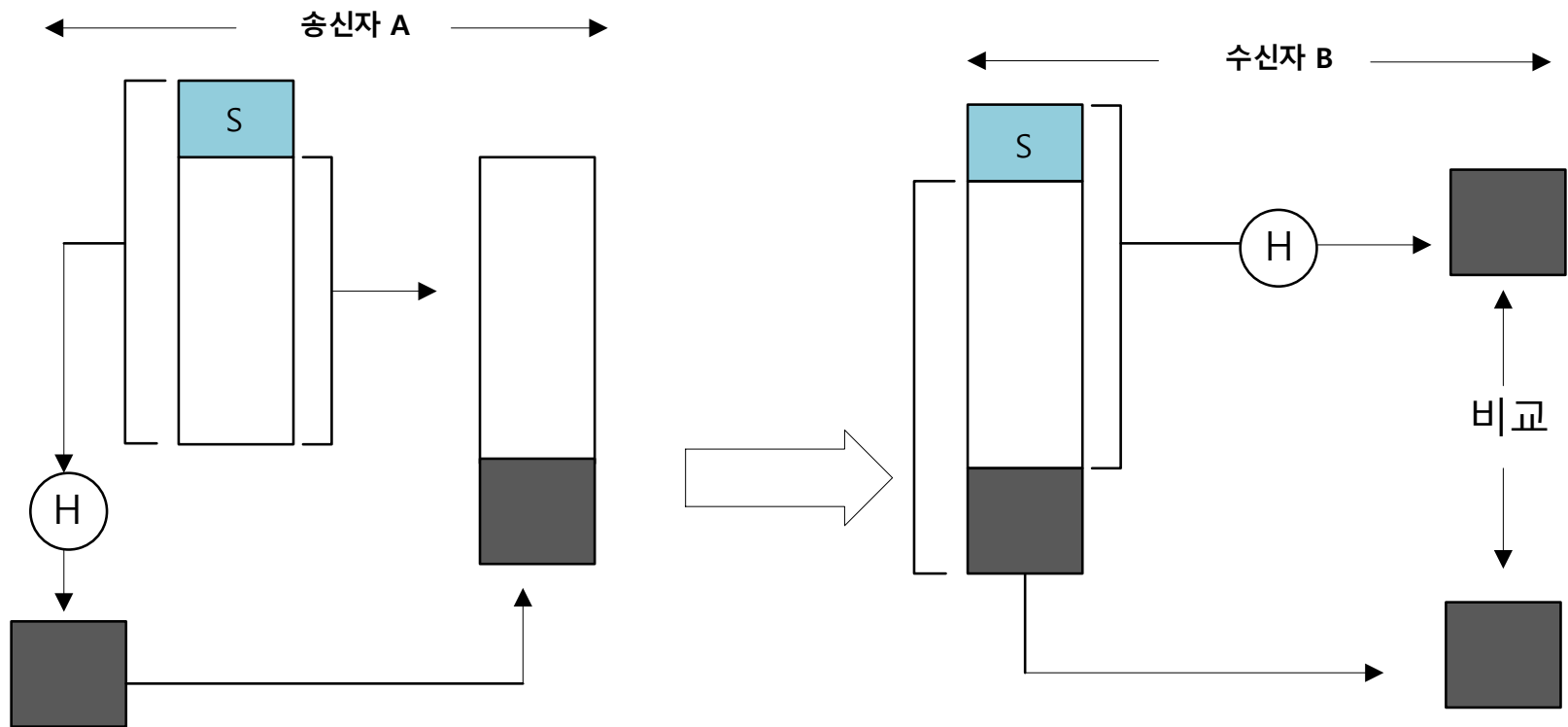
- 공개 키 암호를 사용한 인증

- 송신자의 개인 키를 통해 암호화, 송신자의 공개 키를 통해 복호화



해시 함수

- 일방향 해시 함수
 - 비밀 값을 사용한 인증
 - 송수신자는 공유하는 비밀 값 S 를 사용



해시 함수

- 해시 함수

- 해시 함수 필수 조건

- 어떠한 크기의 데이터 블록에도 적용될 수 있어야 함
 - 일정한 길이의 출력 값을 생성해야 함
 - 계산이 쉬워야 하고, 구현을 실제로 할 수 있어야 함

- 해시 함수 성질

- 일방향 성질

- $H(x) = h$ 가 성립되는 x 를 찾는 것이 어려워야 함

- 약한 충돌 저항성

- $H(x) = H(y)$ 를 만족하는 $x \neq y$ 를 찾는 것이 어려워야 함

- 강한 충돌 저항성

- $H(x) = H(y)$ 를 만족하는 쌍 (x, y) 를 찾는 것이 어려워야 함

해시 함수

- 해시 함수 보안

- 길이가 n 비트인 해시 코드에 대한 공격 난이도

해시 함수 성질	공격을 위해 찾아야하는 메시지의 개수
일방향 성질	2^n
약한 충돌 저항성	2^n
강한 충돌 저항성	$2^{n/2}$

- 현 시점에서 256, 384, 512비트 길이의 해시 함수가 가장 많이 쓰임
 - 128비트 해시 함수의 경우, 24시간만에 충돌이 발견됨

해시 함수

- 단순 해시 함수

- 입력은 연속된 n비트 블록으로 간주
- 각 비트의 자리별로 패리티를 계산하는 세로 덧셈임
검사를 함
- $C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$

	비트 1	비트 2	...	비트 n
블록 1	b_{11}	b_{21}	...	b_{n1}
블록 2	b_{12}	b_{22}	...	b_{n2}
...
블록 m	b_{1m}	b_{2m}	...	b_{nm}
해시 코드	C_1	C_2	...	C_n

C_i = 해시 코드의 i번째 비트
 m = 입력의 n 비트 블록의 수
 b_{ij} = j번째 블록의 i번째 비트

해시 함수

- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)
 - 정의
 - NIST가 개발한 암호학적 해시 함수들의 모음
 - 종류
 - SHA-0
 - SHA-1
 - SHA-2
 - SHA-224
 - SHA-256
 - SHA-384
 - SHA-512
 - SHA-3

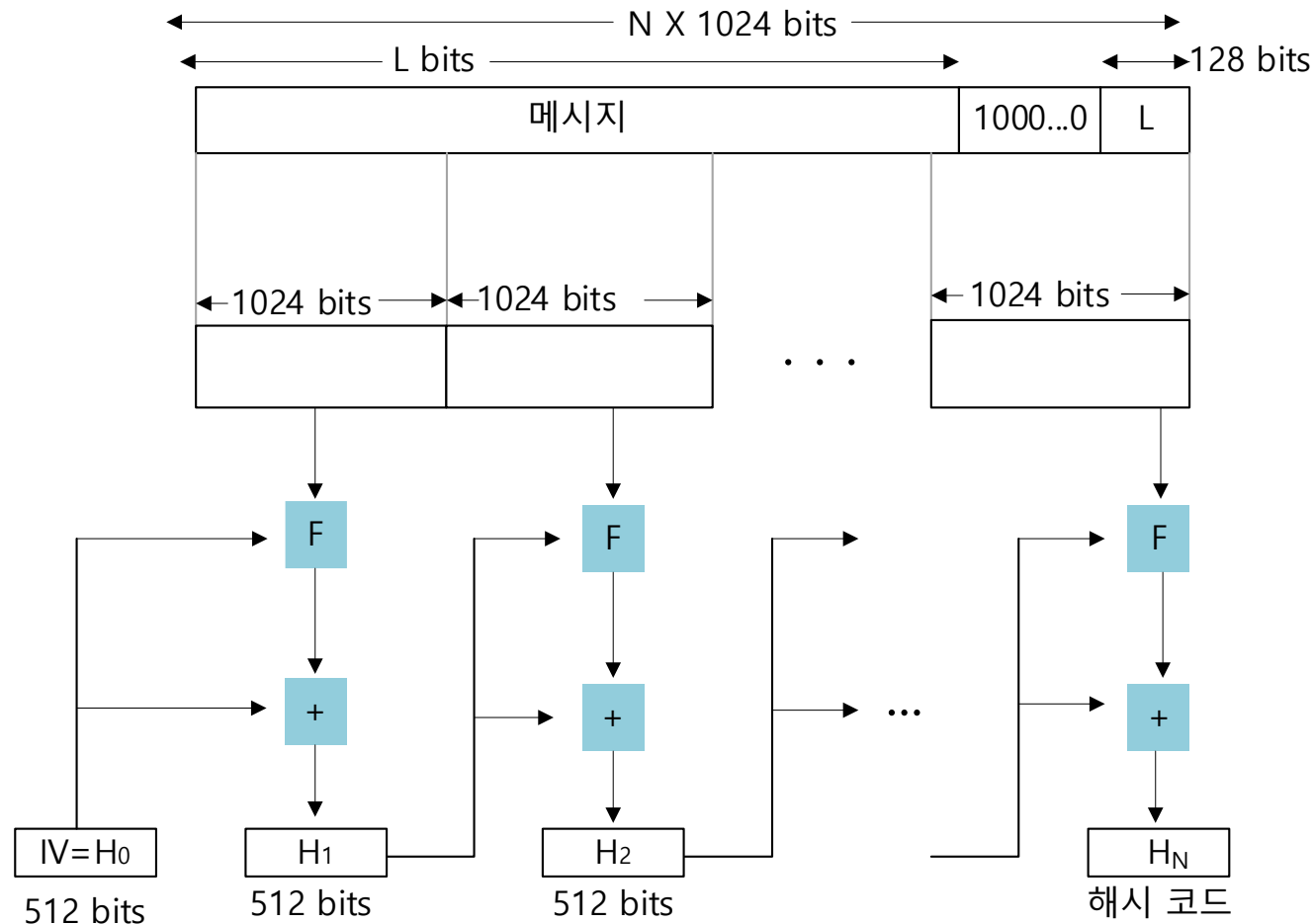
해시 함수

- 안전 해시 알고리즘(SHA: Secure Hash Algorithm)
- SHA 매개 변수 비교

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
메시지 다이제스트 비트 길이	160	224	256	384	512
메시지 비트 길이	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
블록 비트 길이	512	512	512	1024	1024
단어 비트 길이	32	32	32	64	64
단계 수	80	64	64	80	80

해시 함수

- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)
- SHA-512 알고리즘의 동작 과정



해시 함수

- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)

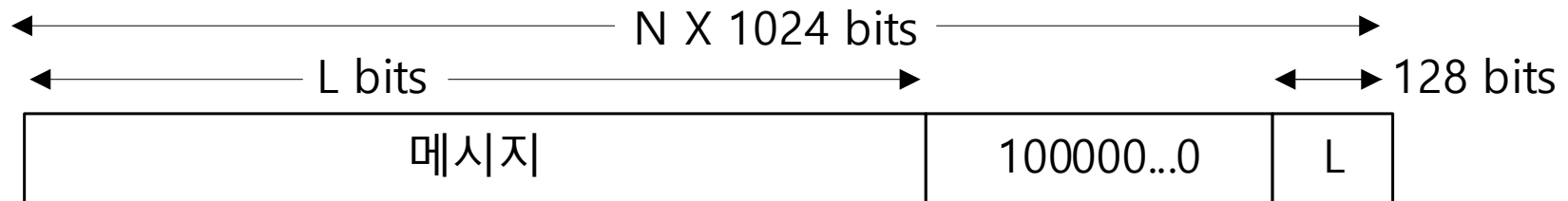
- SHA-512 알고리즘의 동작 과정

1. 패딩 비트 붙이기(Appending Padding Bits)

- 총 길이를 896 (mod 1024)가 되도록 메시지 패딩을 추가
- 패딩을 구성하는 비트는 첫번째 비트가 1, 나머지는 0

2. 길이 붙이기(Append Length)

- 128비트 블록을 메시지에 추가하여 전체 메시지의 길이를 1024의 배수가 되게 만듦
 - 원래 메시지의 길이를 포함하는 블록



해시 함수

- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)

- SHA-512 알고리즘의 동작 과정

- 3. MD 버퍼 초기화(Initialize MD Buffer)

- 64비트 버퍼 8개를 각각의 고정된 64비트 정수로 초기화
 - 이 값은 처음 8개의 소수의 제곱근의 소수점 이하 처음 64비트
 - 512비트 해시 값 출력을 위해 사용됨

a = 6A09E667F3BCC908

b = BB67AE8584CAA73B

c = 3C6EF372FE94F82B

d = A54FF53A5F1D36F1

e = 510E527FADE682D1

f = 9B05688CEB3E6C1F

g = 1F83D9ABFB41BD6B

h = 5BE0CDI9137E2179

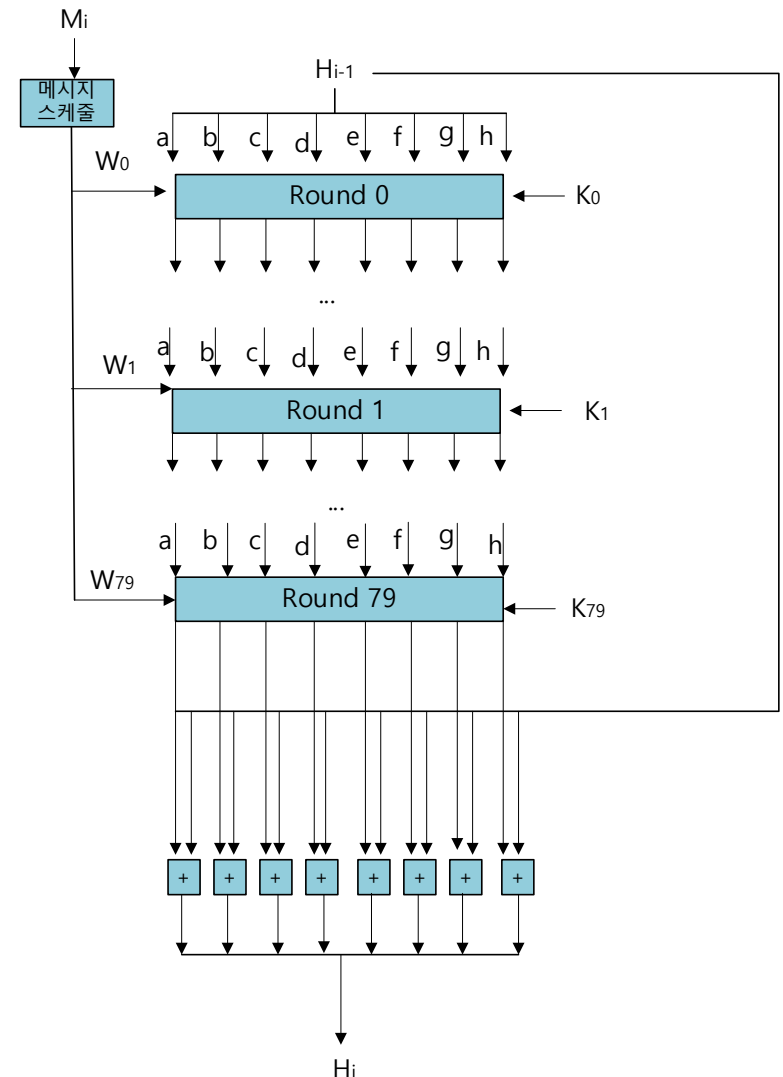
해시 함수

- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)

- SHA-512 알고리즘의 동작 과정

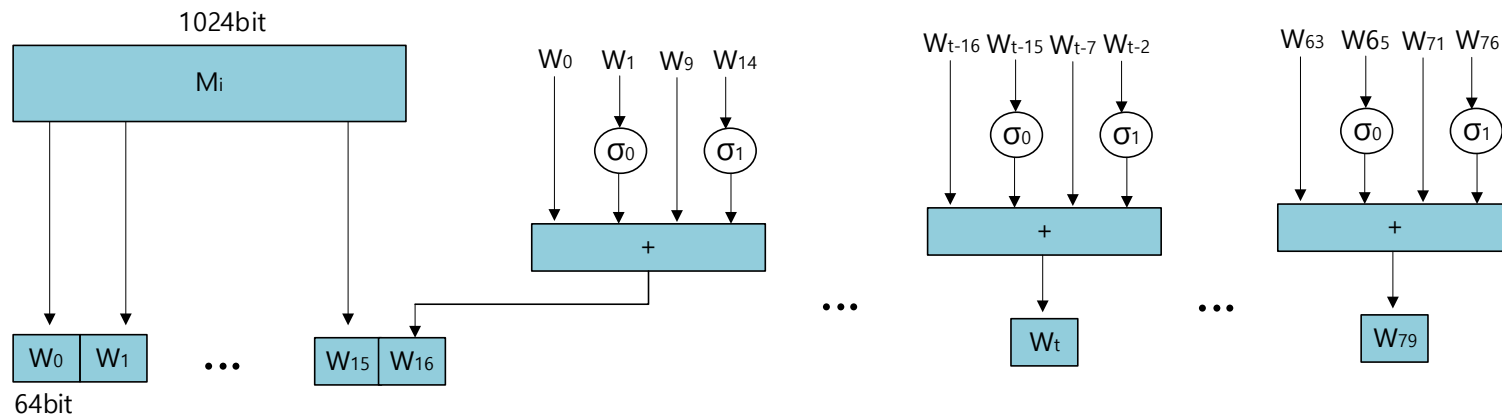
- 4. 1024 비트 블록 메시지 처리
(Process Message in 1024 Bit)

- 총 80라운드로 구성
 - 첫 라운드에서 버퍼는 H_{i-1} 값을 가짐
 - 라운드마다 8개의 64비트 버퍼 값을 갱신
 - 마지막 라운드가 끝난 후 출력 블록을 라운드 시작 전 입력 블록과 각각 XOR한 후 512비트를 출력



해시 함수

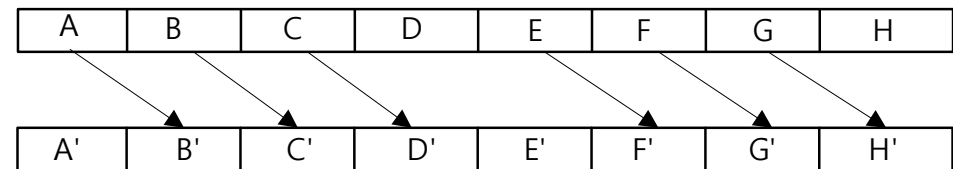
- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)
- SHA-512 알고리즘의 동작 과정
 - 메시지 스케줄
 - 16개의 워드를 80개로 확장



$\sigma_0(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$
 $\sigma_1(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$
 $ROTR^n(x)$ = x 를 n 비트만큼 오른쪽으로
순환 이동한 것
 $SHR^n(x)$ = x 를 n 비트만큼 왼쪽으로 이동하고
오른쪽을 0으로 채움

해시 함수

- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)
- SHA-512 알고리즘의 동작 과정
 - 라운드 함수
 - 상수 K_0, K_1, \dots, K_{79}
 - 처음 80개 소수 세제곱근의 소수점 이하 64비트



Majority(x,y,z):

$(x \text{ AND } y) \oplus (y \text{ AND } z) \oplus (z \text{ AND } x)$

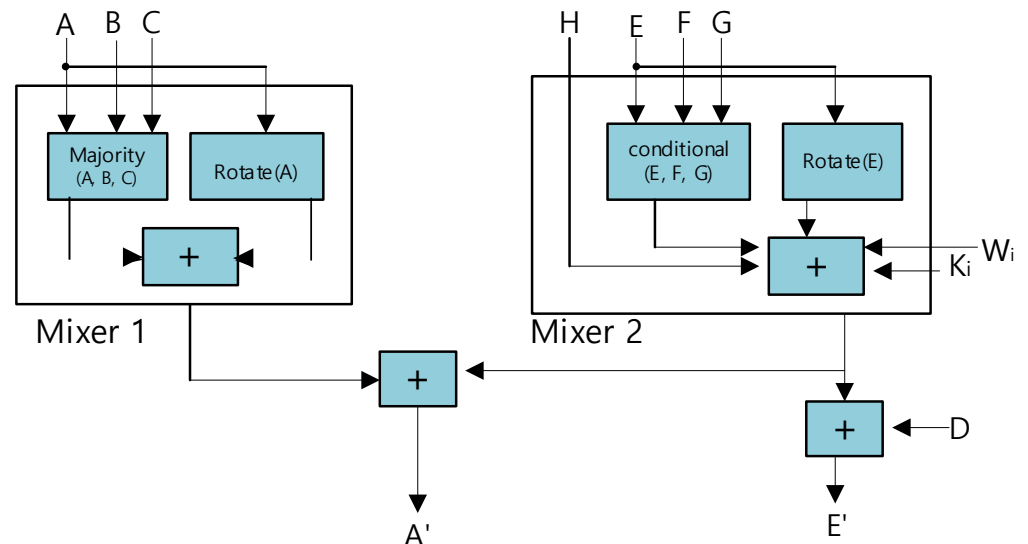
Conditional (x, y, z):

$(x \text{ AND } y) \oplus (\text{NOT } x \text{ AND } z)$

Rotate (A):

$\text{ROTR}_{28}(x) \oplus \text{ROTR}_{34}(x) \oplus \text{ROTR}_{39}(x)$

$\text{ROTR}_i(x)$: x를 i비트 만큼 오른쪽으로 shift



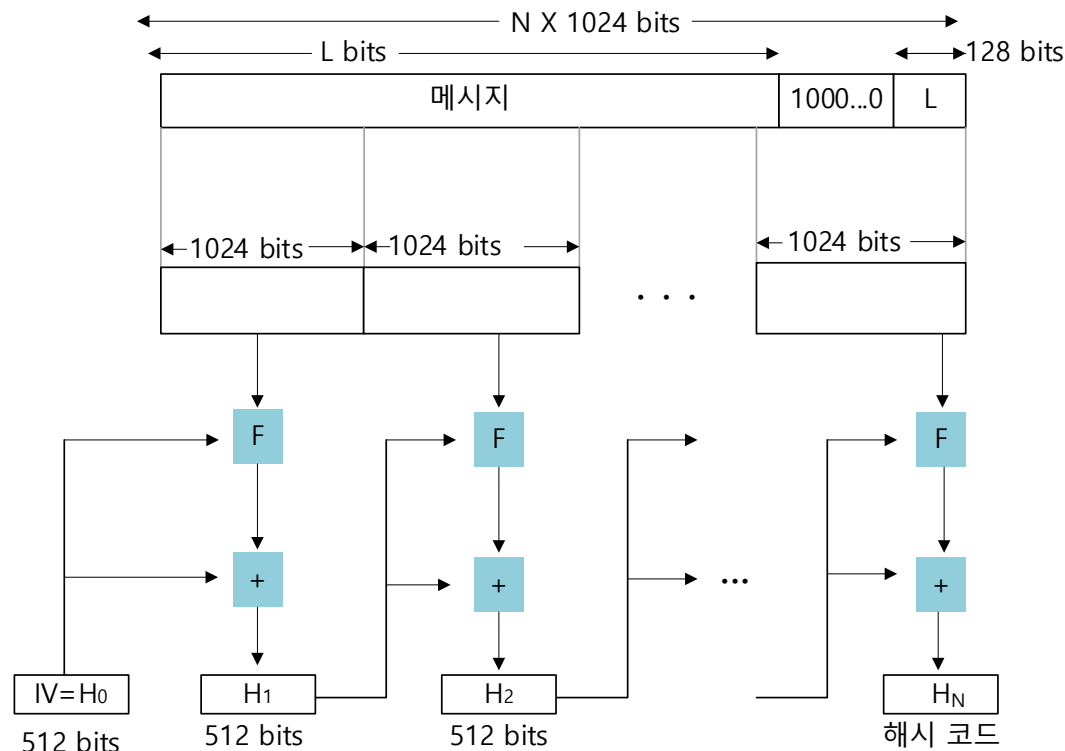
해시 함수

- 안전 해시 알고리즘(SHA, Secure Hash Algorithm)

- SHA-512 알고리즘의 동작 과정

5. 출력(Output)

- N개의 1024 비트 블록 모두가 처리된 뒤에 N번째 단계에서 512 비트 메시지 다이제스트를 얻음



목 차

- 보충

- DES, AES 암호화 과정에서의 대체/치환
- 페이스텔 구조에서의 Sub Key 생성과정
- 페이스텔 구조에서의 F함수
- 스트림 암호와 RC4
- 암호 블록 운용 모드

- 메시지 인증 방법

- 해시 함수

- 메시지 인증 코드

메시지 인증 코드

- HMAC(Hashed MAC)

- 정의

- 해시 함수와 비밀 키를 수반하는 메시지 인증 코드

- 특징

- 해싱 기법을 사용하여 메시지의 위변조 방지
 - 기존 해시 함수 변경 가능
 - 기능 저하가 유발되지 않도록 설계됨

메시지 인증 코드

- HMAC(Hashed MAC)
- 용어

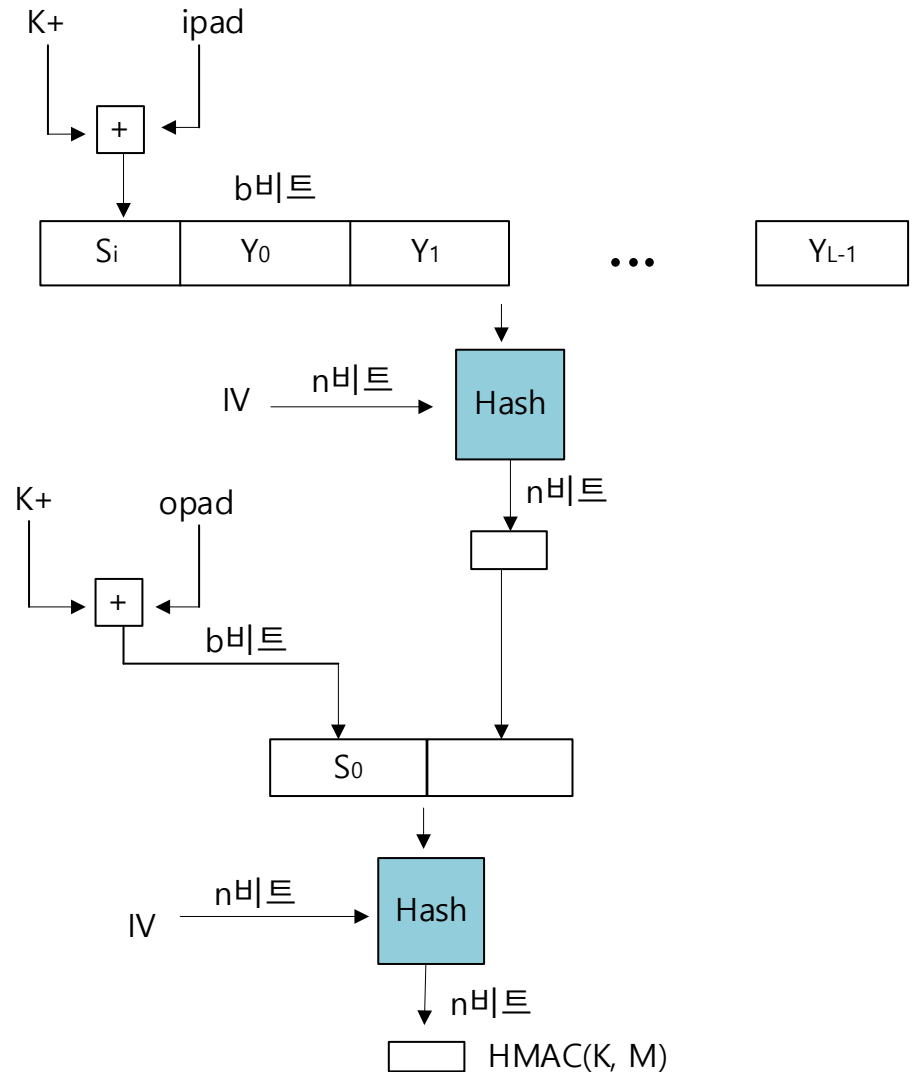
용어	뜻
H	해시 함수
M	HMAC의 입력 메시지
Y_i	M 의 i 번째 블록
L	M 의 블록 수
b	블록의 비트 수
n	내장된 해시 함수에 의해 생성된 해시 코드의 길이
K	비밀 키
K^+	K 의 왼쪽에 0을 붙여서 길이가 b 비트가 되도록 한 것
ipad	00110110를 $b/8$ 번 반복한 2진 수열
opad	01011100를 $b/8$ 번 반복한 2진 수열

메시지 인증 코드

- HMAC(Hashed MAC)

- 동작 과정

- 1) b 비트 스트림 K^+ 생성
- 2) K^+ 와 $ipad$ 를 XOR연산하여 b 비트 블록 S_i 생성
- 3) S_i 에 M 을 붙임
- 4) 3)에서 생성된 스트림에 해시함수를 적용
- 5) K^+ 와 $opad$ 를 XOR연산하여 b 비트 블록 S_0 생성
- 6) 4)에서 얻은 해시 결과를 S_0 에 붙임
- 7) 6)에서 생성된 스트림에 해시함수를 적용해서 출력



메시지 인증 코드

- CMAC(Cipher based MAC)
 - 정의
 - 블록 암호 기반 메시지 인증 코드
 - 특징
 - AES, 3DES가 사용됨
 - 메시지는 일정 길이의 블록으로 나뉨
 - 2개의 키가 사용됨

메시지 인증 코드

- CMAC(Cipher based MAC)
- 계산식과 용어

$$\begin{aligned}C_1 &= E(K, M_1) \\C_2 &= E(K, [M_2 \oplus C_1]) \\C_3 &= E(K, [M_3 \oplus C_2]) \\&\vdots \\&\vdots \\C_n &= E(K, [M_N \oplus C_{n-1} \oplus K_1]) \\T &= MSB_{Tlen}(C_n)\end{aligned}$$

용어	뜻
T	메시지 인증 코드, 또는 태그(tag)라고 함
T_{len}	T 의 비트 길이
$MSB_s(X)$	비트열 X 의 왼쪽부터 s 개 비트
K	k 비트 암호키
K_1	$E(K, 0) \ll 1$
K_2	$E(K_1, 0) \ll 1$

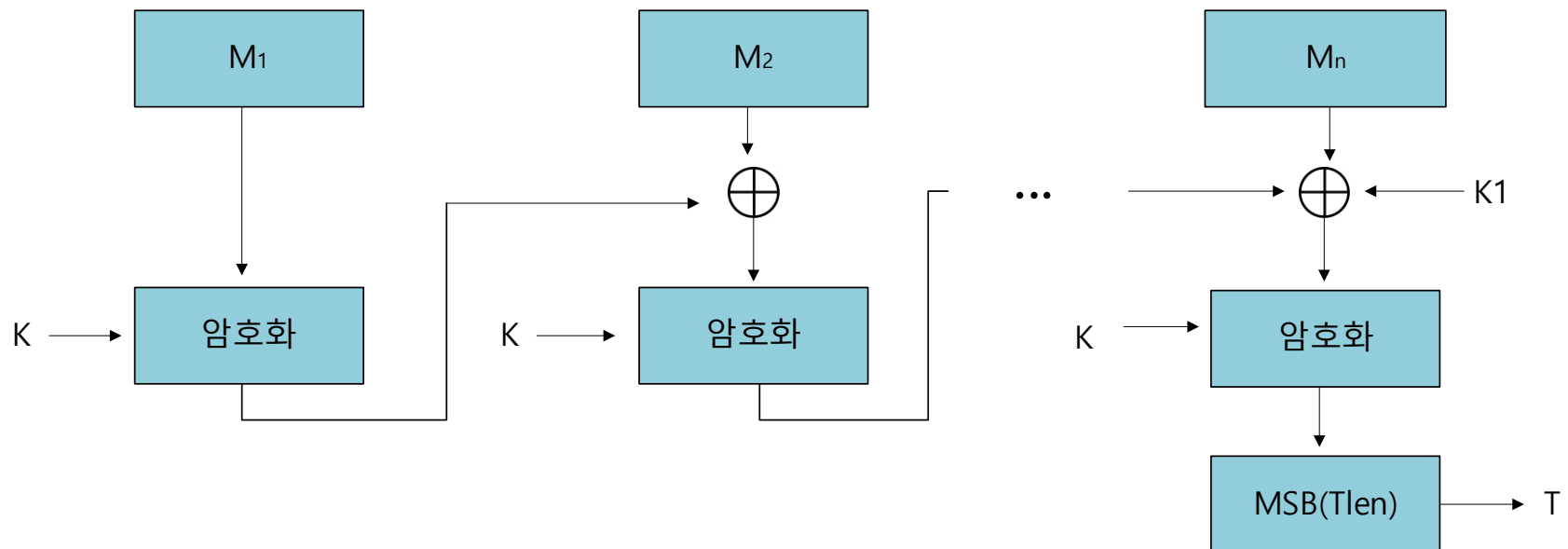
메시지 길이가 블록의 정수배인 경우의 계산식

메시지 인증 코드

- CMAC(Cipher based MAC)

- 동작 과정

- 메시지 길이가 블록 길이의 정수배인 경우
 - k 비트 암호 키 K 와 b 비트 키 K_1 를 사용

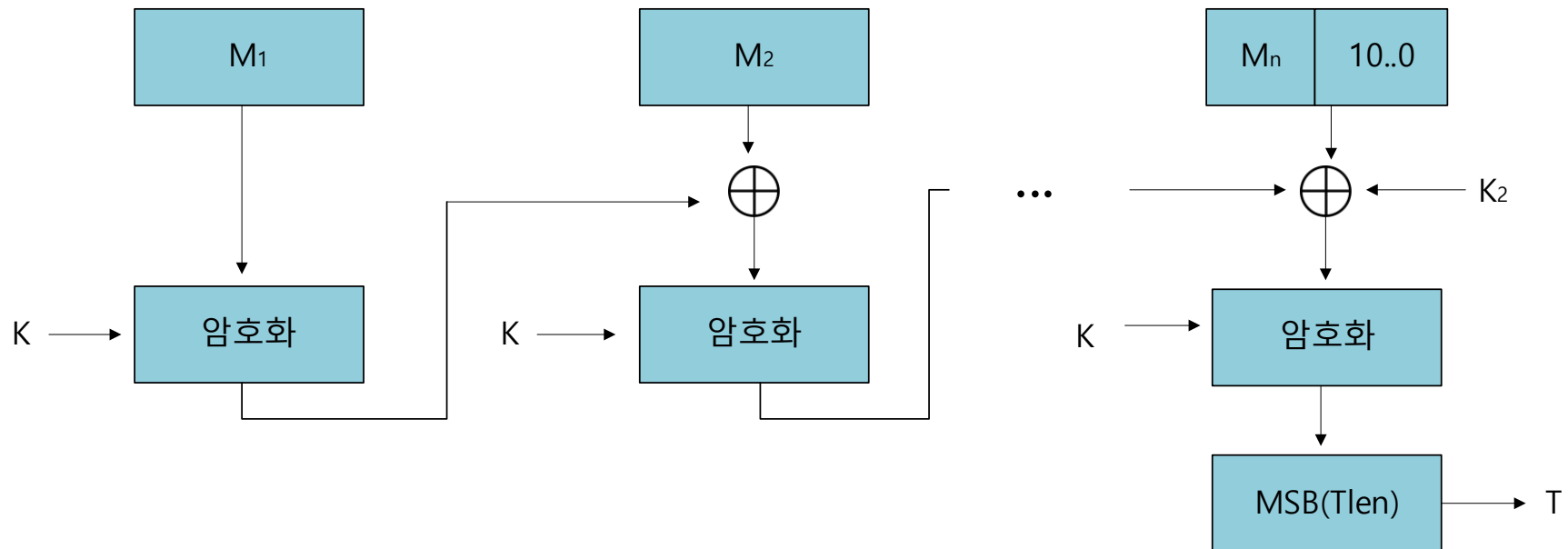


메시지 인증 코드

- CMAC(Cipher based MAC)

- 동작 과정

- 메시지 길이가 블록 길이의 정수배가 아닌 경우
 - 마지막 평문 블록을 패딩 처리하여 메시지의 길이를 블록 길이의 정수배가 되도록 함
 - k비트 암호 키 K 와 b비트 키 K_2 를 사용

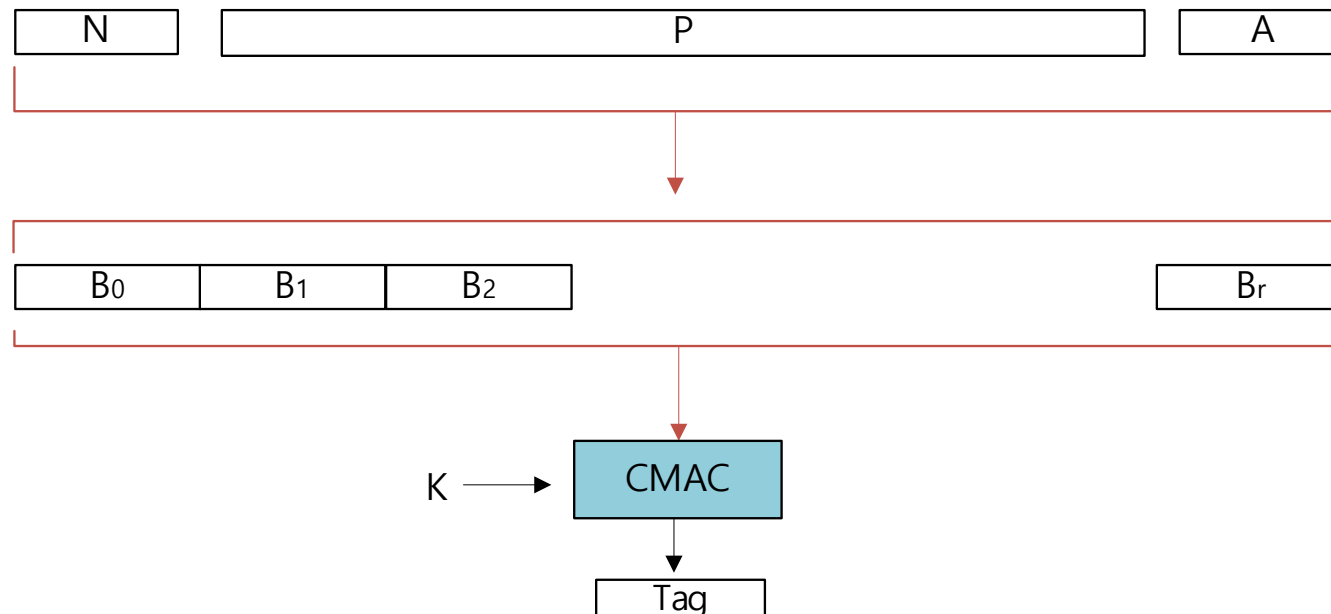


메시지 인증 코드

- CCM(Counter With Cipher Block Chaining MAC)
 - 정의
 - 하나의 키 아래에서 블록 암호를 반복적으로 안전하게 이용하게 하는 메시지 인증 코드
 - 특징
 - AES 알고리즘, CTR 운용 모드, CMAC 알고리즘 사용
 - 암호화와 인증에 동일한 키를 사용
 - 인증과 기밀성을 모두 제공하도록 설계됨

메시지 인증 코드

- CCM(Counter With Cipher Block Chaining MAC)
- 인증 과정
 - 비표(N), 유관 데이터(A), 평문(P)를 입력
 - 입력 값을 B_0 부터 B_r 까지 블록 열 형식으로 나타냄
 - 첫 번째 블록에는 N, A와 P의 길이를 나타내는 형식 비트가 추가됨
 - 블록 열을 CMAC 알고리즘의 입력 값으로 사용



메시지 인증 코드

- CCM(Counter With Cipher Block Chaining MAC)

- 암호화 과정

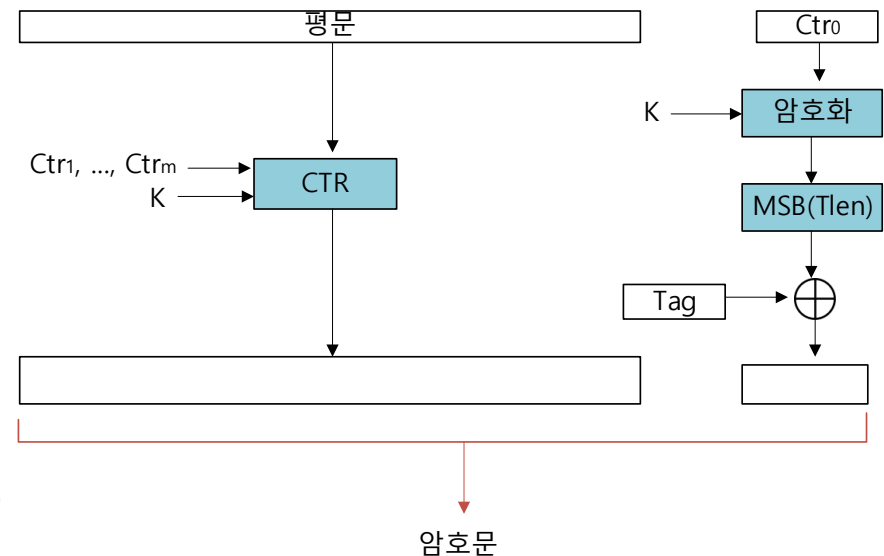
- 인증 태그를 Ctr_0 를 이용하여 CTR모드로 암호화

- 카운터는 비표를 기초로 해서 작성됨

- 출력된 비트 중 $Tlen$ 개의 유효 비트를 태그와 XOR해서 암호화된 태그 생성

- 나머지 카운터는 평문을 CTR 모드로 암호화 할 때 사용

- 암호화된 평문을 암호화된 태그와 이어 붙여서 암호문으로 출력



Thanks!

강민채 (minchae@pel.sejong.ac.kr)