VP-MAC 알고리듬 규격 및 라이브러리 설명서

국민대학교 연구팀
김동찬, 염용진, 전창열, 이제원, 김영효, 김민지

2022.08.16.

목차

제 1 장 정의와 기호, 표기 6

제 1 절 정의 6
제 2 절 기호 6
제 3 절 표기 7
3.1 데이터 색인 7
3.2 바이트열과 워드열 간의 변환 8

제 2 장 WF-LEA 9

제 1 절 개요 9
제 2 절 기본 함수 9
제 3 절 키스케줄 함수 KEYSCHEDULE 9
제 4 절 라운드 함수 ROUND/INVROUND 10
제 5 절 암/복호 함수 ENCRYPT/DECRYPT 11
제 6 절 4 비트 단위 규격 12
6.1 기본 4 비트 함수 12
6.2 32 비트 모듈러덧셈/모듈러뺄셈 13
6.3 32 비트 비트회전 14

제 3 장 WHITEBOX WF-LEA 16

제 1 절 개요 16
1.1 구현 방식 16
1.2 연산기 구성 17
제 2 절 인코딩 19
2.1 외부인코딩 19
2.2 네트워크 랜덤인코딩 19
2.3 절 4 비트 단위 랜덤인코딩 생성 19
제 3 절 기본 함수 21

3.1 ADDWB와 SUBWB 21
3.2 ADDWB와 SUBWB 22
3.3 ROLWB, RORWB, ROLWB, RORWB 23
3.4 ROUNDWB와 INVROUNDWB 24
3.5 ENCRYPTWB 27
3.6 DECRYPTWB 28
제 4 절 네트워크 랜덤인코딩 조건식 31
4.1 암호 연산용 31
4.2 복호 연산용 31
제 5 절 테이블 참조 구현 35
5.1 ADDWB, SUBWB, ROLWB, RORWB 테이블 구현 35
5.2 ROUNDWB, INVROUNDWB, ENCRYPTWB, DECRYPTWB 38
제 6 절 성능 39
6.1 화이트박스 키테이블 크기 39
6.2 속도 39

제 4 장 WHITEBOX WF-LEA기반 기밀성 운영모드 41

제 1 절 평문 확장 41
제 2 절 WB-CBC 41
제 3 절 WB-CTR 43

제 5 장 WHITEBOX WF-LEA 기반 메시지 인증모드 VP-MAC 46

제 1 절 CBC-MAC 46
제 2 절 메시지 확장 46
제 3 절 인증값 생성 및 검증 47
제 4절 VP − MAC − GenTagWB의 구현 50
4.1 A(B(X)⊕Y) 50
4.2 화이트박스키 $\boldsymbol{WKs}, \boldsymbol{WKe}$ 생성 51
4.3 VP-MAC-**GenTagWB** 52

제 6 장 VP-MAC 라이브러리 53

제 1 절 개발환경 **53**
제 2 절 라이브러리 구성 **53**
제 3 절 사용법 **53**
컴파일 53
3.2 난수발생기 설정 54
3.3 키길이 설정 54
3.4 LEA 로 설정하기 55
3.5 WF-LEA 암/복호 연산 55
3.6 WHITEBOX WF-LEA 57

부록 A 테스트벡터 **64**

제 1 절 WHITEBOX WF-LEA **64**
1.1 WHITEBOX WF-LEA128 64
1.2 WHITEBOX WF-LEA192 74
1.3 WHITEBOX WF-LEA256 84
제 2 절 WB-CBC BASED ON LEA **94**
2.1 LENGTH = 79-BYTES 94
2.2 LENGTH = 80-BYTE 95
2.3 LENGTH = 81-BYTE 97
제 3 절 WB-CTR BASED ON LEA **98**
3.1 LENGTH = 79-BYTE 98
3.2 LENGTH = 80-BYTE 99
3.3 LENGTH = 81-BYTE 101
제 4 절 VP-MAC BASED ON WF − LEA **102**
4.1 LENGTH = 79-BYTE 102
4.2 LENGTH = 80-BYTE 103
4.3 LENGTH = 81-BYTE 104
4.4 LENGTH = 161-BYTE 105
4.5 32 바이트 시드로 4 비트 단위 16 바이트 치환 생성하기 105

부록 B 블록암호 LEA 키스케줄 **108**

제 1 절 LEA128 ..... **108**
테스트벡터 109
제 2 절 LEA192 ..... **109**
테스트벡터 110

제 3 절 LEA256 111
테스트벡터 112

제 1장 정의와 기호, 표기

본 장에서는 본 규격서에서 사용하는 기호와 표기를 소개한다.

제 1절 정의

니블(nibble)은 4비트열, 바이트(byte)는 8비트열, 워드(word)는 32비트열로 정의한다.

제 2절 기호

다음의 기호를 사용한다.

0b	비트열 또는 정수의 2진법 표기를 위한 접두사
0x	비트열 또는 정수의 16진법 표기를 위한 접두사
$\mathbb{Z}$	모든 정수의 집합
$a \bmod n$	정수 a와 양의 정수 n에 대하여 a를 n으로 나눈 나머지
$div(a, b)$	정수 a를 음이 아닌 정수 b로 나누었을 때의 몫 q과 나머지 r를 반환, 즉, $q, r \leftarrow div(a, b)$
$[a, b)$	a보다 크거나 같고, b보다 작은 모든 정수의 집합, $\{x \in \mathbb{Z}: a \leq x < b\}$
$[n]$	0부터 n보다 작은 모든 정수의 집합, 즉, $[n] = \{0,1, \dots, n-1\}$
$\lceil a \rceil$	실수 a보다 작지 않은 최소 정수, $\min\{x \in \mathbb{Z}: x \geq a\}$
$\lfloor a \rfloor$	실수 a보다 크지 않은 최대 정수, $\max\{x \in \mathbb{Z}: x \leq a\}$
$\{0,1\}^n$	모든 n비트열의 집합
$\mathcal{N}$	모든 니블의 집합, 즉, $\mathcal{B} = \{0,1\}^4$
$\mathcal{B}$	모든 바이트의 집합, 즉, $\mathcal{B} = \{0,1\}^8$
$\mathcal{W}$	모든 워드의 집합, 즉, $\mathcal{B} = \{0,1\}^{32}$
$\mathcal{B}^n$	n개 바이트의 모든 연접 집합, 즉, $\mathcal{B}^n = \{X_0 \parallel \cdots \parallel X_{n-1}: X_n \in \mathcal{B}\}$
$\mathcal{N}^n$	n개 니블의 모든 연접 집합, 즉, $\mathcal{N}^n = \{X_0 \parallel \cdots \parallel X_{n-1}: X_n \in \mathcal{N}\}$
$\mathcal{W}^n$	n개 워드의 모든 연접 집합, 즉, $\mathcal{W}^n = \{X_0 \parallel \cdots \parallel X_{n-1}: X_n \in \mathcal{W}\}$
$X \parallel Y$	두 비트열 X와 Y의 연접
X^n	비트열 X을 n회 반복 연접한 비트열, 즉, $X^n = \underbrace{X \parallel X \parallel \cdots \parallel X}_{n \text{ times}}$
$X \mid Y$	두 n비트열 X와 Y의 논리합 (OR)
$X \oplus Y$	두 n비트열 $X, Y \in \{0,1\}^n$의 논리적 배타합 (XOR; eXclusive OR)
$X \ll l$	n비트열 $X \in \{0,1\}^n$를 l비트 왼쪽으로 이동

$X \gg l$ n비트열 $X \in \{0,1\}^n$를 l비트 오른쪽으로 이동

$X \lll l$ n비트열 $X \in \{0,1\}^n$를 l비트 왼쪽으로 회전

$X \ggg l$ n비트열 $X \in \{0,1\}^n$를 l비트 오른쪽으로 회전

$Y \leftarrow X$ X의 비트열을 Y에 대입하는 연산

$\mathrm{B2I}(X)$ n비트열 $X = x_{n-1} \| \cdots \| x_0 \in \{0,1\}^n$를 음이 아닌 n비트 정수 $\sum_{j=0}^{n-1} x_j\, 2^j \in [0, 2^n)$로 변환

$\mathrm{I2B}_n(X)$ 음이 아닌 n비트 정수 $\sum_{j=0}^{n-1} x_j\, 2^j \in [0, 2^n)$를 n비트열 $X = x_{n-1} \| \cdots \| x_0 \in \{0,1\}^n$를 변환

$X \boxplus Y$ 두 n비트열 X, Y를 음이 아닌 정수로 간주한 2^n-모듈러덧셈

즉, $\mathrm{I2B}_n(\mathrm{B2I}(X) + \mathrm{B2I}(Y) \bmod 2^n)$

$X \boxminus Y$ 두 n비트열 X, Y를 음이 아닌 정수로 간주한 2^n-모듈러뺄셈

즉, $\mathrm{I2B}_n(\mathrm{B2I}(X) - \mathrm{B2I}(Y) \bmod 2^n)$

$X + i$ n비트열 X와 음이 아닌 정수 i에 대해 $X \boxplus \mathrm{I2B}_n(i)$를 반환

$\mathcal{P}_n$ n비트 모든 치환함수의 집합

$f \leftarrow \mathcal{P}_n$ $\mathcal{P}_n$에서 임의 선택한 n비트 치환 f

$\mathrm{diag}(F_{m-1}, \ldots, F_0) \in (\mathcal{P}_n)^m$ 다음과 같이 m개의 n비트 치환함수 F_j로 정의하는 mn비트 치환 함수

$$\mathrm{diag}(F_{m-1}, \ldots, F_0)(X) := F_{m-1}(X_{m-1}) \| \cdots \| F_0(X_0), \quad \text{where } X = X_{m-1} \| \cdots \| X_0,\ X_j \in \{0,1\}^n.$$

Memo.

(1) $F = \mathrm{diag}(F_{m-1}, \ldots, F_0)$의 역함수 F^{-1}는 다음과 같다.

$$F^{-1} = \mathrm{diag}(F_{m-1}^{-1}, \ldots, F_0^{-1}).$$

(2) $\mathcal{P}_1$에 속하는 치환함수는 다음 2가지뿐이다.

$$f(x) = x, \quad f(x) = x \oplus 1.$$

제 3절 표기

3.1 데이터 색인

비트열 내 비트색인 n비트열 X의 비트색인은 다음과 같이 내림차순으로 표기한다.

$$X = x_{n-1} \| x_{n-2} \| \cdots \| x_0, \quad \big(x_j \in \{0,1\}\big).$$

워드 내 니블색인 워드 X의 니블색인은 다음과 같이 내림차순으로 표기한다.

$$X = X_7 \| X_6 \| \cdots \| X_0 \in \mathcal{N}^8.$$

워드 내 바이트색인 워드 X의 바이트색인은 다음과 같이 내림차순으로 표기한다.

$$X = X_3 \| X_2 \| X_1 \| X_0 \in \mathcal{B}^4.$$

데이터열 색인 데이터열 X(니블열 또는 바이트열 또는 워드열)의 색인은 다음과 같이 오름차순으로 표기한다.

$$X = (X_j)_{j=0}^{n-1} = X_0 \| X_1 \| X_2 \| \cdots \| X_{n-1}.$$

3.2 바이트열과 워드열 간의 변환

바이트열 $(B_j)_{j=0,1,\ldots,4n-1} \in \mathcal{B}^{4n}$과 워드열 $(W_j)_{j=0,1,\ldots,n-1} \in \mathcal{W}^n$ 간 변환은 다음의 규약을 따른다.

$$W_j = B_{4j+3} \| B_{4j+2} \| B_{4j+1} \| B_{4j} \in \mathcal{W}, \quad j = 0,1,\ldots,n-1.$$

제 2장 WF-LEA

제 1절 개요

블록암호 WF-LEA 일방향 키스케줄을 사용하는 블록암호 LEA이다 . WF-LEA는 키길이를 기준으로 WF-LEA128, WF-LEA192, WF-LEA256 으로 분류하며, 각 알고리듬의 파라미터는 [표 2.1: 블록암호 WF-LEA 파라미터] 과 같다.

[표 2.1: 블록암호 WF-LEA 파라미터]

알고리듬	블록길이 (**b**-byte)	키길이(**k**-byte)	라운드키길이 (byte)	라운드수(**r**)	라운드키 전체크기(byte)
WF-LEA128	16	16	24	24	576(= 24 × 24)
WF-LEA192	16	24	24	28	672(= 24 × 28)
WF-LEA256	16	32	24	32	768(= 24 × 32)

Memo. WF-LEA 소스코드에서 키스케줄함수를 LEA 키스케줄함수로 변경하면 LEA와 동일하다.

제 2절 기본 함수

블록암호 WF-LEA 다음 32비트 단위 기본 함수를 사용한다.

$\mathrm{XOR}(X, Y)$	$X, Y \in \mathcal{W}$의 논리적 배타합을 반환 (XOR; eXclusive OR)
$\mathrm{ROL}_l(X)$	$X \in \mathcal{W}$의 l비트 왼쪽 회전 비트열을 반환
$\mathrm{ROR}_l(X)$	$X \in \mathcal{W}$의 l비트 오른쪽 회전 비트열을 반환
$\mathrm{ADD}(X, Y)$	32비트 모듈러덧셈 즉, $X, Y \in \mathcal{W}$에 대해 $X \boxplus Y = \mathrm{I2B}_{32}(\mathrm{B2I}(X) + \mathrm{B2I}(Y) \bmod 2^{32})$를 반환
$\mathrm{SUB}(X, Y)$	32비트 모듈러뺄셈 즉, $X, Y \in \mathcal{W}$에 대해 $X \boxminus Y = \mathrm{I2B}_{32}(\mathrm{B2I}(X) - \mathrm{B2I}(Y) \bmod 2^{32})$를 반환

제 3절 키스케줄 함수 KeySchedule

블록암호 WF-LEA 는 해시함수 SHA256를 사용하여 **r**개의 192비트 라운드키를 생성한다. 알고리듬 1은 **k**바이트 키 K를 입력받아 **r**개의 24바이트 라운드키 $RK^{(0)}, \ldots, RK^{(\mathbf{r}-1)}$를 생성하는 키스케줄함수 KeySchedule를 보여준다. ((**k**, **r**) = (16,24) or (24,28) or (32,32))

알고리듬 1 WF-LEA: 키스케줄함수 KeySchedule

Input: 키 $K \in \mathcal{B}^{\mathbf{k}}$
Output: 라운드키 $(RK^{(l)})_{l\in[\mathbf{r}]}$ $(RK^{(l)} \in \mathcal{W}^6)$

procedure KeySchedule(K)
  for $l = 0$ to $\mathbf{r}-1$ **do**
    $T \leftarrow \mathsf{SHA256}(K \parallel \mathsf{I2B}_8(l+1))$
    $RK^{(l)} \leftarrow$ First 6-word string of T
  end for
  return $(RK^{(l)})_{l\in[\mathbf{r}]}$
end procedure

Input: 키 $K \in \mathcal{B}^{\mathbf{k}}$, 색인집합 I
Output: 라운드키 $(RK^{(l)})_{l\in I}$ $(RK^{(l)} \in \mathcal{W}^6)$

procedure KeySchedule(K, I)
  for $l \in I$ **do**
    $T \leftarrow \mathsf{SHA256}(K \parallel \mathsf{I2B}_8(l+1))$
    $RK^{(l)} \leftarrow$ First 6-word string of T
  end for
  return $(RK^{(l)})_{l\in I}$
end procedure

Memo. 블록암호 키스케줄은 별첨 B를 참고한다.

제 4절 라운드 함수 Round/InvRound

라운드 함수 Round/InvRound는 128비트 $X \in \mathcal{W}^4$와 192비트 라운드키 $RK \in \mathcal{W}^6$를 입력받아 128비트 $Y \in \mathcal{W}^4$를 출력하는 함수이다.

$$\text{Round/InvRound}: \mathcal{W}^4 \times \mathcal{W}^6 \to \mathcal{W}^4.$$

X, Y, K의 32비트 워드열 색인은 다음과 같다.

$$X := X_0 \parallel X_1 \parallel X_2 \parallel X_3, \quad Y := Y_0 \parallel Y_1 \parallel Y_2 \parallel Y_3, \quad RK := RK_0 \parallel RK_1 \parallel RK_2 \parallel RK_3 \parallel RK_4 \parallel RK_5.$$

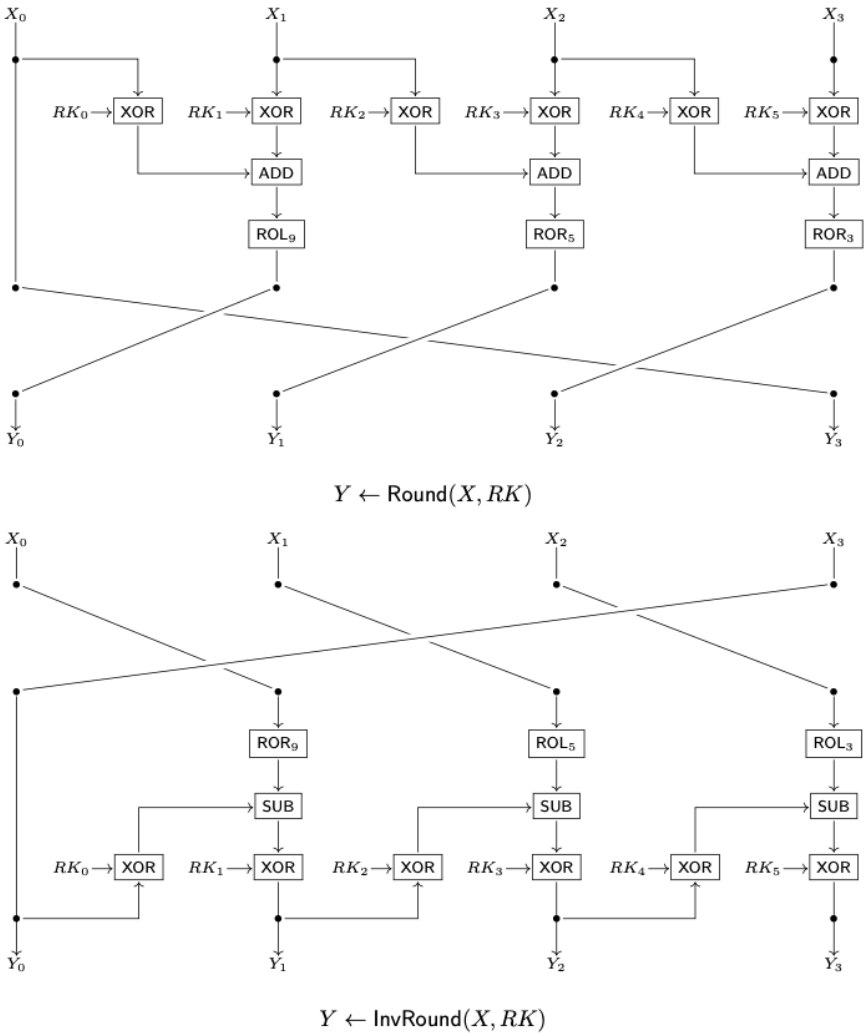
알고리듬 2는 Round와 InvRound의 유사부호이며, [그림 2.1은 이를 도식화한 것이다.

알고리듬 2 WF-LEA: 라운드 함수 Round와 InvRound

Input: $X = X_0\|X_1\|X_2\|X_3 \in \mathcal{W}^4$, $RK = RK_0\|RK_1\|RK_2\|RK_3\|RK_4\|RK_5 \in \mathcal{W}^6$
Output: $Y = Y_0\|Y_1\|Y_2\|Y_3 \in \mathcal{W}^4$

procedure Round(X, RK)
  $A, B \leftarrow \mathsf{XOR}(X_0, RK_0), \mathsf{XOR}(X_1, RK_1)$
  $T \leftarrow \mathsf{ADD}(A, B)$
  $Y_0 \leftarrow \mathsf{ROL}_9(T)$
  $A, B \leftarrow \mathsf{XOR}(X_1, RK_2), \mathsf{XOR}(X_2, RK_3)$
  $T \leftarrow \mathsf{ADD}(A, B)$
  $Y_1 \leftarrow \mathsf{ROR}_5(T)$
  $A, B \leftarrow \mathsf{XOR}(X_2, RK_4), \mathsf{XOR}(X_3, RK_5)$
  $T \leftarrow \mathsf{ADD}(A, B)$
  $Y_2 \leftarrow \mathsf{ROR}_3(T)$
  $Y_3 \leftarrow X_0$
  return $Y = Y_0\|Y_1\|Y_2\|Y_3$
end procedure

procedure InvRound(X, RK)
  $Y_0 \leftarrow X_3$
  $A, B \leftarrow \mathsf{ROR}_9(X_0), \mathsf{XOR}(Y_0, RK_0)$
  $T \leftarrow \mathsf{SUB}(A, B)$
  $Y_1 \leftarrow \mathsf{XOR}(T, RK_1)$
  $A, B \leftarrow \mathsf{ROL}_5(X_1), \mathsf{XOR}(Y_1, RK_2)$
  $T \leftarrow \mathsf{SUB}(A, B)$
  $Y_2 \leftarrow \mathsf{XOR}(T, RK_3)$
  $A, B \leftarrow \mathsf{ROL}_3(X_2), \mathsf{XOR}(Y_2, RK_4)$
  $T \leftarrow \mathsf{SUB}(A, B)$
  $Y_3 \leftarrow \mathsf{XOR}(T, RK_5)$
  return $Y = Y_0\|Y_1\|Y_2\|Y_3$
end procedure



[그림 2.1: WF-LEA 라운드 함수: Round, InvRound]

제 5절 암/복호 함수 Encrypt/Decrypt

알고리듬 3은 WF-LEA의 암호함수 Encrypt와 복호함수 Decrypt의 유사부호이다.

알고리듬 3 WF-LEA: 암호함수 Encrypt와 복호함수 Decrypt

Input: $X \in \mathcal{B}^{16}$, $K \in \mathcal{B}^{\mathrm{k}}$
Output: $Y \in \mathcal{B}^{16}$

```
1: procedure Encrypt(X, K)
2:     (RK^(l))_{l∈[r]} ← KeySchedule(K)
3:     X_0 ← X
4:     for l = 0 to r − 1 do
5:         X_{l+1} ← Round(X_l, RK^(l))
6:     end for
7:     Y ← X_r
8:     return Y
9: end procedure
```

```
1: procedure Decrypt(X, K)
2:     (RK^(l))_{l∈[r]} ← KeySchedule(K)
3:     X_0 ← X
4:     for l = 0 to r − 1 do
5:         X_{l+1} ← InvRound(X_l, RK^(r−1−l))
6:     end for
7:     Y ← X_r
8:     return Y
9: end procedure
```

고정키 K에 대해 $\mathrm{Encrypt}(X,K)$와 $\mathrm{Decrypt}(X,K)$를 각각 $\mathrm{Encrypt}_K(X)$, $\mathrm{Decrypt}_K(X)$로 표기한다.

제 6절 4비트 단위 규격

6.1 기본 4비트 함수

다음 함수를 정의한다.

1. $c', Z \leftarrow \mathrm{add}(c, X, Y)$

   $c \in \{0,1\}, X, Y \in \mathcal{N}$에 대해 $\mathrm{B2I}(c) + \mathrm{B2I}(X) + \mathrm{B2I}(Y)$를 16으로 나누었을 때 몫에 해당하는 비트 c'과 나머지에 해당하는 4비트열 Z를 반환

$$c' = \mathrm{I2B}_1\left(\left\lfloor \frac{\mathrm{B2I}(c) + \mathrm{B2I}(X) + \mathrm{B2I}(Y)}{2^4} \right\rfloor\right), \quad Z = \mathrm{I2B}_4(\mathrm{B2I}(c) + \mathrm{B2I}(X) + \mathrm{B2I}(Y) \bmod 2^4).$$

2. $b', Z \leftarrow \mathrm{sub}(b, X, Y)$

   $b \in \{0,1\}, X, Y \in \mathcal{N}$에 대해 $\mathrm{B2I}(X) - \mathrm{B2I}(Y) - \mathrm{B2I}(b)$를 16으로 나누었을 때 몫의 절댓값에 해당하는 비트 b'과 나머지에 해당하는 4비트열 Z을 반환

$$b' = \mathrm{I2B}_1\left(\left\lfloor \left| \frac{\mathrm{B2I}(X) - \mathrm{B2I}(Y) - \mathrm{B2I}(b)}{2^4} \right| \right\rfloor\right), Z = \mathrm{I2B}_4(\mathrm{B2I}(X) - \mathrm{B2I}(Y) - \mathrm{B2I}(b) \bmod 2^4).$$

3. $Z \leftarrow \mathrm{rol}_l(X, Y)$

   $X, Y \in \mathcal{N}$에 대해 $Z = (X \ll l) | (Y \gg (4-l)) \in \mathcal{N}$를 반환

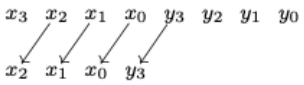
4. $Z \leftarrow \mathrm{ror}_l(X, Y)$

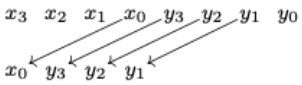
$X, Y \in \mathcal{N}$에 대해 $Z = \big(X \ll (4-l)\big) | (Y \gg l) \in \mathcal{N}$를 반환

Memo. $\mathrm{rol}_l(X,Y) = \big((X \| Y) \gg (4-l)\big) \wedge 1^l, \mathrm{ror}_l(X,Y) = \big((X \| Y) \gg l\big) \wedge 1^l$.

Example. $X = x_3 \| x_2 \| x_1 \| x_0, Y = y_3 \| y_2 \| y_1 \| y_0 \in \mathcal{N}$에 대해 다음이 성립한다.

$\mathrm{rol}_1(X,Y) = \mathrm{ror}_3(X,Y) = x_2 \| x_1 \| x_0 \| y_3$, $\mathrm{ror}_1(X,Y) = \mathrm{rol}_3(X,Y) = x_0 \| y_3 \| y_2 \| y_1$.





6.2 32비트 모듈러덧셈/모듈러뺄셈

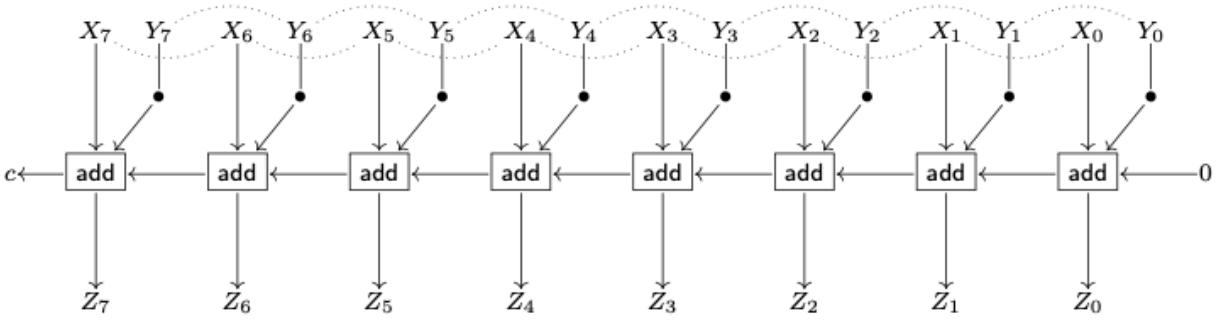
32비트 모듈러덧셈 ADD와 32비트 모듈러뺄셈 SUB는 각각 4비트 함수 add와 sub를 사용하여 알고리듬 4와 같이 계산한다.

알고리듬 4 $\mathrm{ADD}(X,Y)$와 $\mathrm{SUB}(X,Y)$

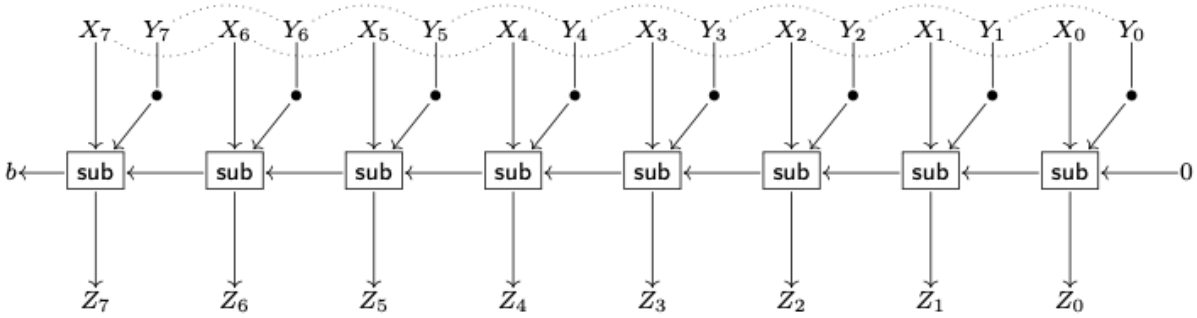
Input: $X = X_7 \| X_6 \| \cdots \| X_0, Y = Y_7 \| Y_6 \| \cdots \| Y_0 \in \mathcal{N}^8$
Output: $Z = Z_7 \| Z_6 \| \cdots \| Z_0 \in \mathcal{N}^8$

```
1: procedure ADD(X, Y)
2:     c ← 0
3:     for j = 0 to 7 do
4:         c, Z_j ← add(c, X_j, Y_j)
5:     end for
6:     return Z = Z_7 || Z_6 || ··· || Z_0
7: end procedure
```

```
1: procedure SUB(X, Y)
2:     b ← 0
3:     for j = 0 to 7 do
4:         b, Z_j ← sub(b, X_j, Y_j)
5:     end for
6:     return Z = Z_7 || Z_6 || ··· || Z_0
7: end procedure
```



$Z \leftarrow \mathrm{ADD}(X, Y)$



$Z \leftarrow \mathrm{SUB}(X, Y)$

[그림2.2: 32비트 모듈러덧셈 $\mathrm{ADD}(X, Y)$과 모듈러뺄셈 $\mathrm{SUB}(X, Y)$]

6.3 32비트 비트회전

32비트 비트회전 ROL과 ROR은 각각 rol와 ror을 사용해서 알고리듬 5와 같이 계산한다.

알고리듬 5 $\mathrm{ROL}_l(X)$와 $\mathrm{ROR}_l(X)$ $(0 < l < 32)$

Input: $X = X_7 \| X_6 \| \cdots \| X_0 \in \mathcal{N}^8$
Output: $Z = Z_7 \| Z_6 \| \cdots \| Z_0 \in \mathcal{N}^8$

procedure $\mathrm{ROL}_l(X)$
  $q, r \leftarrow div(l, 4)$
  for $j = 0$ to 7 **do**
    $Z_j \leftarrow \mathrm{rol}_r(X_{j-q \bmod 8}, X_{j-q-1 \bmod 8})$
  end for
  return $Z = Z_7 \| Z_6 \| \cdots \| Z_0$
end procedure

procedure $\mathrm{ROR}_l(X)$
  $q, r \leftarrow div(l, 4)$
  for $j = 0$ to 7 **do**
    $Z_j \leftarrow \mathrm{ror}_r(X_{j+q+1 \bmod 8}, X_{j+q \bmod 8})$
  end for
  return $Z = Z_7 \| Z_6 \| \cdots \| Z_0$
end procedure

다음은 WF-LEA에서 사용하는 비트회전 함수의 규격이다.

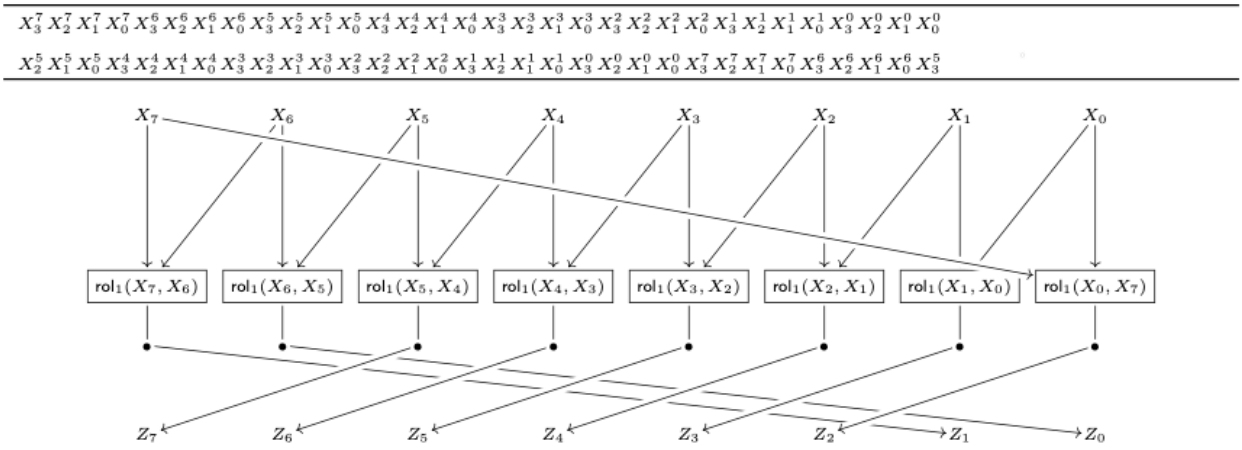
1. $\mathrm{ROL}_9(X)$: $Z_j \leftarrow \mathrm{rol}_1(X_{j-2 \bmod 8}, X_{j-3 \bmod 8})$
2. $\mathrm{ROR}_5(X)$: $Z_j \leftarrow \mathrm{ror}_1(X_{j+2 \bmod 8}, X_{j+1 \bmod 8})$

3. $\mathrm{ROR}_3(X)$: $Z_j \leftarrow \mathrm{ror}_3\left(X_{j+1 \bmod 8}, X_{j \bmod 8}\right)$

4. $\mathrm{ROR}_9(X)$: $Z_j \leftarrow \mathrm{ror}_1\left(X_{j+3 \bmod 8}, X_{j+2 \bmod 8}\right)$

5. $\mathrm{ROL}_5(X)$: $Z_j \leftarrow \mathrm{rol}_1\left(X_{j-1 \bmod 8}, X_{j-2 \bmod 8}\right)$

6. $\mathrm{ROL}_3(X)$: $Z_j \leftarrow \mathrm{rol}_3\left(X_{j \bmod 8}, X_{j-1 \bmod 8}\right)$



[그림2.3: $Z \leftarrow \mathrm{ROL}_9(X)$]

Memo. C언어 구현 시 $j-2 \bmod 8$을 (j-2)%8이 아닌 (j+6)%8으로 구현해야 함을 주의한다. $j-3 \bmod 8, j-1 \bmod 8$도 각각 (j+5)%8와 (j+7)%8으로 구현한다.

제 3 장 Whitebox WF-LEA

제 1절 개요

1.1 구현 방식

1.1.1 외부인코딩

Whitebox WF-LEA는 WF-LEA를 외부인코딩(external encoding) 기법을 사용하여 구현한다. 즉, 키 K와 무관한 두 가역 외부인코딩 A, B에 대해 암호 연산 Encrypt_K는 다음과 같이 표현할 수 있다.

$$\text{Encrypt}_K = \text{B} \circ \text{B}^{-1} \circ \text{Encrypt}_K \circ \text{A}^{-1} \circ \text{A}.$$

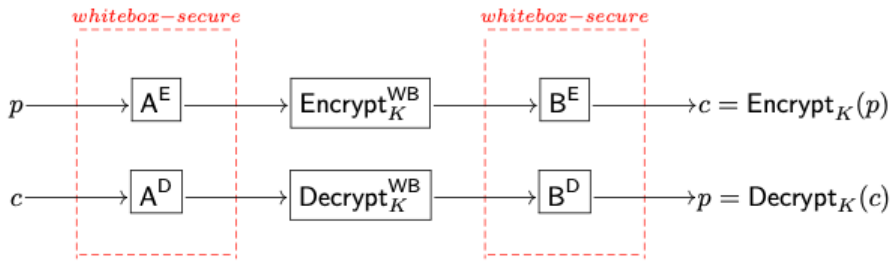
여기서 화이트박스 공격자에게 비밀인 연산 A와 B는 화이트박스 공격자가 접근할 수 없는 영역(whitebox-secure zone)에서 수행하고, 화이트박스 공격 가능 영역(whitebox zone)에서는 $\text{B}^{-1} \circ \text{Encrypt}_K \circ \text{A}^{-1}$를 수행한다. 이때 $\text{B}^{-1} \circ \text{Encrypt}_K \circ \text{A}^{-1}$를 WF-LEA의 화이트박스 암호 연산이라고 하며, $\text{Encrypt}_K^{\text{WB}}$라고 표기한다. 즉,

$$\text{Encrypt}_K^{\text{WB}} := \text{B}^{-1} \circ \text{Encrypt}_K \circ \text{A}^{-1}.$$

동일한 방식으로 WF-LEA의 복호 연산 Decrypt_K에 대해 화이트박스 복호 연산 $\text{Decrypt}_K^{\text{WB}}$를 다음과 같이 정의한다.

$$\text{Decrypt}_K^{\text{WB}} := \text{B}^{-1} \circ \text{Decrypt}_K \circ \text{A}^{-1}.$$

이때, 암호 연산에서 사용하는 외부인코딩 A와 B는 키와 관련이 없으며, 복호 연산에서 사용하는 A와 B와는 다름을 주의한다. 앞으로 암호 연산에서 사용하는 외부인코딩을 $\text{A}^{\text{E}}, \text{B}^{\text{E}}$, 복호 연산에서 사용하는 외부인코딩을 $\text{A}^{\text{D}}, \text{B}^{\text{D}}$라고 표기한다. 그림 3.1은 외부인코딩을 적용한 화이트박스 연산을 도식화한 것이다.



[그림3.1: 외부인코딩을 적용한 화이트박스 연산]

1.1.2 네트워크 랜덤인코딩

l비트 치환함수 $F_1, F_2, \ldots, F_n$의 결합함수인 $F = F_n \circ F_{n-1} \circ \cdots \circ F_1$는 임의의 l비트 치환함수 $R_1, R_2, \ldots, R_{n-1}$에 대해 다음 함수와 동치이다.

$$\begin{aligned} F &= F_n \circ (R_{n-1})^{-1} \circ R_{n-1} \circ F_{n-1} \circ (R_{n-2})^{-1} \circ R_{n-2} \circ \cdots \circ (R_1)^{-1} \circ R_1 \circ F_1 \\ &= {F'}_n \circ {F'}_{n-1} \circ \cdots \circ {F'}_1, \end{aligned}$$

where

$${F'}_l := \begin{cases} R_1 \circ F_1, & l = 1, \\ R_l \circ F_l \circ (R_{l-1})^{-1}, & l = 2, \ldots, n-1, \\ F_n \circ (R_{n-1})^{-1}, & l = n. \end{cases}$$

F_l가 비밀함수일 때, $R_l \circ F_l \circ (R_{l-1})^{-1}$와 같이 랜덤치환함수를 F_l의 전후로 결합시켜 비밀함수 F_l의 입출력을 숨기는 방식을 네트워크 랜덤인코딩(networked random encoding)이라고 한다. WF-LEA의 화이트박스 연산 $\text{Encrypt}_K^{\text{WB}}$와 $\text{Decrypt}_K^{\text{WB}}$는 이 방식을 사용한다.

1.2 연산기 구성

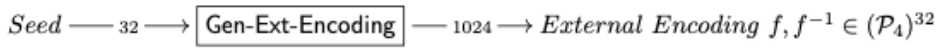
Whitebox WF-LEA는 다음 연산기를 가진다.

1. WF-LEA 암/복호 연산기
   - 입력: 데이터(16바이트열), 키(16/24/32바이트열)
   - 출력: 데이터(16바이트열)

2. 외부변환 바이너리 생성기
   - 입력: 시드(32바이트열)

— 출력: 외부인코딩 연산 정보 바이너리

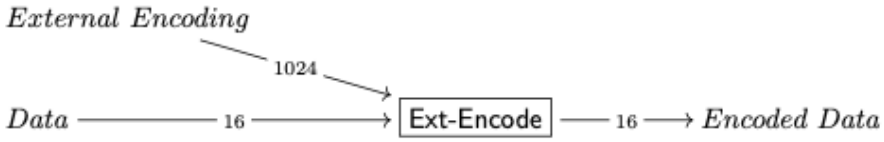
(1,024바이트열 = 16 × 8 × 4 × 2; 4비트 랜덤치환 32개와 그 역변환 32개 생성)



3. 데이터 외부변환 연산기 (whitebox-secure zone에서 수행)

— 입력: 데이터(16바이트열), 외부변환 바이너리(1,024바이트열)

— 출력: 데이터(16바이트열)

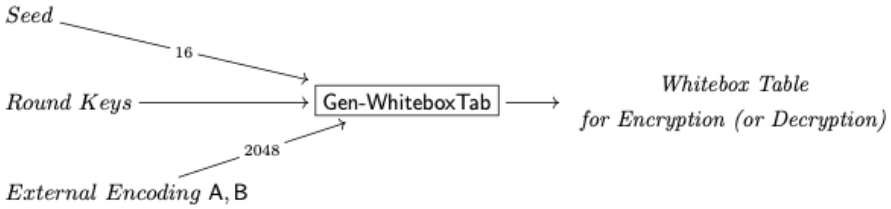


※ 외부변환 바이너리는 변환 f와 그 역변환 f^{-1}를 함께 저장하고 있기 때문에 Ext-Encode는 두 변환 중 하나를 선택하여 입력 데이터를 인코딩한다.

4. 암/복호 연산용 화이트박스 키테이블 생성기

— 입력: 시드(32바이트열), 라운드키, 외부변환 A와 B 바이너리(= 2,048바이트열= 2 × 1,024)

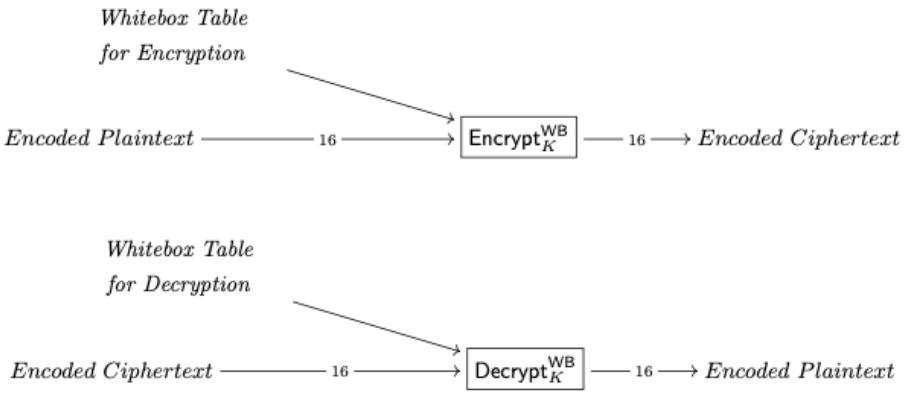
— 출력: 화이트박스 키테이블 바이너리 (442,368 or 516,096 or 589,824바이트열)



5. 화이트박스 암/복호 연산기: $\text{Encrypt}_K^{\text{WB}}$, $\text{Decrypt}_K^{\text{WB}}$

— 입력: 데이터(16바이트열), 암/복호 연산용 화이트박스 키테이블 바이너리 (442,368 or 516,096 or 589,824바이트열)

— 출력: 데이터(16바이트열)



제 2절 인코딩

2.1 외부인코딩

외부인코딩 A는 4비트 단위 32비트 치환함수 4개 A_0, A_1, A_2, A_3으로 정의한다. 즉,

$$\mathrm{A} := \mathrm{diag}(A_0, A_1, A_2, A_3) \in (\mathcal{P}_4^8)^4, \quad \text{where } A_j := \mathrm{diag}\left(A_{j,7}, \dots, A_{j,0}\right) \in \mathcal{P}_4^8.$$

외부인코딩 B 역시 동일하다.

$$\mathrm{B} := \mathrm{diag}(B_0, B_1, B_2, B_3) \in (\mathcal{P}_4^8)^4, \quad \text{where } B_j := \mathrm{diag}\left(B_{j,7}, \dots, B_{j,0}\right) \in \mathcal{P}_4^8.$$

2.2 네트워크 랜덤인코딩

Whitebox WF-LEA는 라운드 함수에 다음 세 가지 네트워크 랜덤인코딩을 사용한다.

1. 덧셈(뺄셈) 입/출력에 적용하는 4비트 단위 랜덤인코딩
2. 덧셈(뺄셈) 내 받아올림(받아내림)에 적용하는 1비트 단위 랜덤인코딩
3. 비트회전 입/출력에 적용하는 4비트 단위 랜덤인코딩

2.3절 4비트 단위 랜덤인코딩 생성

Fisher-Yates Shuffle은 1938년 Ronald Fisher와 Frank Yates이 제안한 유한 집합에 대한 랜덤 전단사 함수를 생성하는 알고리듬이다. 알고리듬 6은 이 알고리듬의 동작 과정을 보여준다.

알고리듬 6 Fisher-Yates Shuffle

Input: 배열 $A = [A_0, A_1, \ldots, A_{n-1}]$
Output: 섞인 배열 A

```
1: procedure Shuffle(A)
2:     for i ← n − 1 downto 1 do
3:         j ←$ [0..i]
4:         swap(A_i, A_j)
5:     end for
6:     return A
7: end procedure
```

4비트 랜덤인코딩은 16바이트 난수열과 Fisher-Yates Shuffle을 응용하여 모든 입/출력 정보를 가진 테이블 형태로 생성한다. 알고리듬 7은 이 과정을 보여준다.

알고리듬 7 4비트 랜덤인코딩 생성

Input: 16바이트 난수열 $R = R_0 \| \cdots \| R_{15} \in \mathcal{B}^{16}$
Output: 4비트 랜덤인코딩 테이블 $A[16]$

```
1: procedure GenRand4BitPerm(R)
2:     A ← [0, 1, ..., 15]
3:     for i ← 15 downto 1 do
4:         t ← ⌊log_2(i)⌋ + 1
5:         cnt ← 0
6:         while cnt ≤ 8 − t do
7:             j ← B2I(Last t-bit of (R_i ≫ cnt))
8:             if j ≤ i then
9:                 BREAK
10:            end if
11:            cnt ← cnt + 1
12:        end while
13:        j ← j mod (i + 1)
14:        swap(A[i], A[j])
15:    end for
16:    return A
17: end procedure
```

Memo.

1. 알고리듬 6에서 $j \xleftarrow{\$} [0..i]$는 난수비트열이 아닌 정수형태의 난수를 생성해야 한다. 이를 위해 단순 소거법, 복합 소거법, 단순 모듈러법, 복합 모듈러법 등의 방식을 사용한다. (TTAK.KO-12.0189/R1) 하지만 이 방식은 상수 시간 알고리듬이 아니거나, 64비트 이상의 난수열을 필요로 하기에 구현 효율성을 고려하여 Whitebox WF-LEA는 알고리듬 7와 같이 최대 8비트 내에서 i보다 작거나 같은 랜덤 정수 j를 결정하는 방식을 사용한다.

2. 첫번째 바이트 R_0는 사용하지 않으나, 구현의 용이함을 위해 난수열의 길이를 16바이트로 설정한다.

4비트 랜덤인코딩을 생성하기 위해서 16바이트가 필요하다. 따라서 4비트 단위 치환의 연접인 128비트 치환 $A \in (\mathcal{P}_4^8)^4$를 임의 생성하기 위해서는 $512 (= 16 \times 32)$바이트의 난수열이 필요하다. 알고리듬 8은 32바이트 시드로부터 512바이트를 생성하는

동작과정으로, LEA128-CTR-DRBG를 현 상황에 맞게 간소화한 것이다. A의 저장이 필요한 경우 A를 테이블을 저장시 512(= 16 × 32)바이트가 필요하지만 32바이트의 시드만을 저장하고, 이 시드로부터 A를 생성하는 방식으로 구현하면 메모리를 절약할 수 있다.

알고리듬 8 512바이트 난수열 생성

Input: 32바이트 시드 $S = S_0 \| S_1 \in (\mathcal{B}^{\mathbf{b}})^2$
Output: 512바이트 $R = R_0 \| \cdots \| R_{31} \in (\mathcal{B}^{\mathbf{b}})^{32}$

procedure GenRand512Bytes(S)
  $K, V \leftarrow 0\text{x}00^{\mathbf{b}}, 0\text{x}00^{\mathbf{b}}$
  $K, V \leftarrow \mathsf{Encrypt}_K(V+1) \oplus S_0, \mathsf{Encrypt}_K(V+2) \oplus S_1$
  for $i = 0$ to 31 **do**
    $R_i \leftarrow \mathsf{Encrypt}_K(V+1+i)$
  end for
  return $R = R_0 \| \cdots \| R_{31}$
end procedure

제 3절 기본 함수

3.1 $\mathsf{add}^{\mathsf{WB}}$와 $\mathsf{sub}^{\mathsf{WB}}$

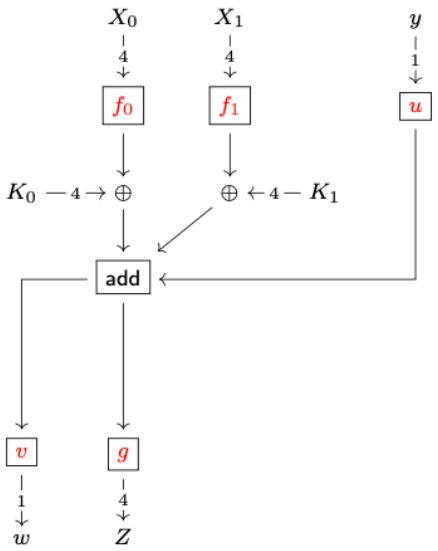
1비트 단위 치환 $u, v \in \mathcal{P}_1$와 4비트 단위 치환 $f_0, f_1, g \in \mathcal{P}_4$, 4비트 키 $K_0, K_1 \in \mathcal{N}$에 대해 함수 $\mathsf{add}^{\mathsf{WB}}$와 $\mathsf{sub}^{\mathsf{WB}}$를 알고리듬 9과 같이 정의한다. 그림 3.2는 이 두 함수를 도식화한 것이다.

알고리듬 9 $\mathsf{add}^{\mathsf{WB}}$와 $\mathsf{sub}^{\mathsf{WB}}$

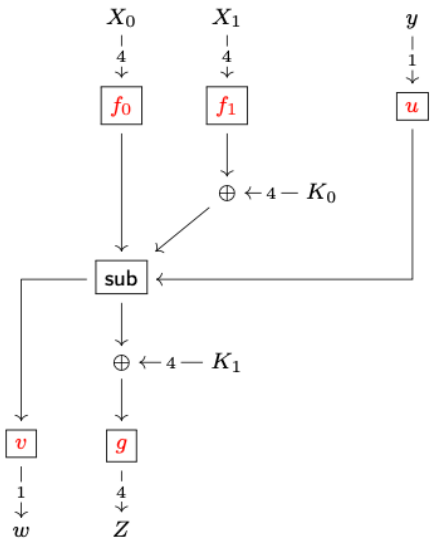
Input: $u, v \in \mathcal{P}_1,\ f_0, f_1, g \in \mathcal{P}_4,\ y \in \{0,1\},\ K_0, K_1, X_0, X_1 \in \mathcal{N}$
Output: $w \in \{0,1\}, Z \in \mathcal{N}$

procedure $\mathsf{add}^{\mathsf{WB}}(u, v, f_0, f_1, g, K_0, K_1; y, X_0, X_1)$
  $w, Z \leftarrow \mathsf{add}(u(y), f_0(X_0) \oplus K_0, f_1(X_1) \oplus K_1)$
  $w, Z \leftarrow v(w), g(Z)$
  return w, Z
end procedure

procedure $\mathsf{sub}^{\mathsf{WB}}(u, v, f_0, f_1, g, K_0, K_1; y, X_0, X_1)$
  $w, Z \leftarrow \mathsf{sub}(u(y), f_0(X_0), f_1(X_1) \oplus K_0)$
  $w, Z \leftarrow v(w), g(Z \oplus K_1)$
  return w, Z
end procedure



$\mathsf{add}^{\mathsf{WB}}(u, v, f_0, f_1, g, K_0, K_1; y, X_0, X_1)$



$\mathsf{sub}^{\mathsf{WB}}(u, v, f_0, f_1, g, K_0, K_1; y, X_0, X_1)$

[그림3.2: $\mathsf{add}^{\mathsf{WB}}$와 $\mathsf{sub}^{\mathsf{WB}}$]

3.2 $\mathsf{ADD}^{\mathsf{WB}}$와 $\mathsf{SUB}^{\mathsf{WB}}$

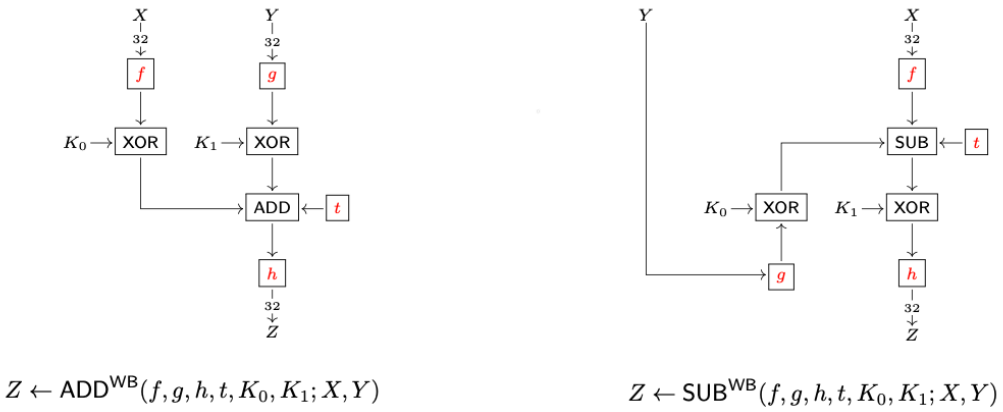
32비트 함수 $\mathsf{ADD}^{\mathsf{WB}}$와 $\mathsf{SUB}^{\mathsf{WB}}$를 알고리듬 10과 같이 4비트 단위 함수로 정의한다. 그림 3.3은 이 두 함수를 도식화한 것이다.

알고리듬 10 $\mathsf{ADD}^{\mathsf{WB}}$와 $\mathsf{SUB}^{\mathsf{WB}}$

Input: $f = \mathsf{diag}(f_7, \ldots, f_0)$, $g = \mathsf{diag}(g_7, \ldots, g_0)$, $h = \mathsf{diag}(h_7, \ldots, h_0) \in \mathcal{P}_4^8$, $t = (t_7, \ldots, t_0) \in \mathcal{P}_1^8$, $K_0 = K_{0,7} \| \cdots \| K_{0,0}, K_1 = K_{1,7} \| \cdots \| K_{1,0}$, $X = X_7 \| \cdots \| X_0$, $Y = Y_7 \| \cdots \| Y_0 \in \mathcal{N}^8$
Output: $Z = Z_7 \| \cdots \| Z_0 \in \mathcal{N}^8$

procedure $\mathsf{ADD}^{\mathsf{WB}}(f, g, h, t, K_0, K_1; X, Y)$
  $t_{-1} \leftarrow$ Identity
  $c \leftarrow 0$
  for $j = 0$ to 7 **do**
    $c, Z_j \leftarrow \mathsf{add}^{\mathsf{WB}}(t_{j-1}, t_j, f_j, g_j, h_j, K_{0,j}, K_{1,j}; c, X_j, Y_j)$
  end for
  return $Z = Z_7 \| \cdots \| Z_0$
end procedure

procedure $\mathsf{SUB}^{\mathsf{WB}}(f, g, h, t, K_0, K_1; X, Y)$
  $t_{-1} \leftarrow$ Identity
  $b \leftarrow 0$
  for $j = 0$ to 7 **do**
    $b, Z_j \leftarrow \mathsf{sub}^{\mathsf{WB}}(t_{j-1}, t_j, f_j, g_j, h_j, K_{0,j}, K_{1,j}; b, X_j, Y_j)$
  end for
  return $Z = Z_7 \| \cdots \| Z_0$
end procedure



$Z \leftarrow \mathsf{ADD}^{\mathsf{WB}}(f, g, h, t, K_0, K_1; X, Y)$

$Z \leftarrow \mathsf{SUB}^{\mathsf{WB}}(f, g, h, t, K_0, K_1; X, Y)$

[그림3.3: $\mathsf{ADD}^{\mathsf{WB}}$와 $\mathsf{SUB}^{\mathsf{WB}}$]

3.3 $\mathsf{rol}^{\mathsf{WB}}$, $\mathsf{ror}^{\mathsf{WB}}$, $\mathsf{ROL}^{\mathsf{WB}}$, $\mathsf{ROR}^{\mathsf{WB}}$

4비트 단위 함수 $\mathsf{rol}_l^{\mathsf{WB}}$와 $\mathsf{ror}_l^{\mathsf{WB}}$를 알고리듬 11과 같이 정의한다. 그림 3.4는 이 두 함수를 도식화한 것이다.

알고리듬 11 $\mathsf{rol}_l^{\mathsf{WB}}$와 $\mathsf{ror}_l^{\mathsf{WB}}$ $(0 < l < 4)$

Input: $f, g, h \in \mathcal{P}_4$, $X, Y \in \mathcal{N}$
Output: $Z \in \mathcal{N}$

1: **procedure** $\mathsf{rol}_l^{\mathsf{WB}}(f, g, h; X, Y)$
2: $\quad Z \leftarrow \mathsf{rol}_l(f(X), g(Y))$
3: $\quad Z \leftarrow h(Z)$
4: $\quad$ **return** Z
5: **end procedure**

1: **procedure** $\mathsf{ror}_l^{\mathsf{WB}}(f, g, h; X, Y)$
2: $\quad Z \leftarrow \mathsf{ror}_l(f(X), g(Y))$
3: $\quad Z \leftarrow h(Z)$
4: $\quad$ **return** Z
5: **end procedure**

$Z \leftarrow \mathsf{rol}_l^{\mathsf{WB}}(f, g, h; X, Y)$

$Z \leftarrow \mathsf{ror}_l^{\mathsf{WB}}(f, g, h; X, Y)$

[그림3.4: $\mathsf{rol}_l^{\mathsf{WB}}$와$\mathsf{ror}_l^{\mathsf{WB}}$]

32비트 함수 $\mathsf{ROL}_l^{\mathsf{WB}}$와 $\mathsf{ROR}_l^{\mathsf{WB}}$를 알고리듬 12와 같이 4비트 단위 함수로 정의한다. 그림3.5 는 이 두 함수를 도식화한 것이다.

알고리듬 12 $\mathsf{ROL}_l^{\mathsf{WB}}$와 $\mathsf{ROR}_l^{\mathsf{WB}}$ $(0 < l < 32)$

Input: $f = \mathsf{diag}(f_7, \ldots, f_0)$, $g = \mathsf{diag}(g_7, \ldots, g_0) \in \mathcal{P}_4^8$, $X = X_7 \| \cdots \| X_0 \in \mathcal{N}^8$,
Output: $Y = Y_7 \| \cdots \| Y_0 \in \mathcal{N}^8$

```
1: procedure ROL_l^WB(f, g; X)
2:     q, r ← div(l, 4)
3:     for j = 0 to 7 do
4:         u ← j − q mod 8
5:         v ← j − q − 1 mod 8
6:         Y_j ← rol_r^WB(f_u, f_v, g_j; X_u, X_v)
7:     end for
8:     return Y = Y_7‖Y_6‖···‖Y_0
9: end procedure
```

```
1: procedure ROR_l^WB(f, g; X)
2:     q, r ← div(l, 4)
3:     for j = 0 to 7 do
4:         u ← j + q + 1 mod 8
5:         v ← j + q mod 8
6:         Y_j ← ror_r^WB(f_u, f_v, g_j; X_u, X_v)
7:     end for
8:     return Y = Y_7‖Y_6‖···‖Y_0
9: end procedure
```

$Y \leftarrow \mathsf{ROL}_l^{\mathsf{WB}}(f, g; X)$

$Z \leftarrow \mathsf{ROR}_l^{\mathsf{WB}}(f, g; X)$

[그림3.5: ROL_l^{WB}와 $\mathrm{ROR}_l^{\mathsf{WB}}$]

3.4 Round$^{\mathsf{WB}}$와 InvRound$^{\mathsf{WB}}$

32비트 함수 Round$^{\mathsf{WB}}$와 InvRound$^{\mathsf{WB}}$를 알고리듬 13과 같이 정의한다. 그림 3.6과 그림 3.7은 이 두 함수를 도식화한 것이다.

알고리듬 13 Whitebox WF-LEA: 라운드 함수 $\mathsf{Round}^{\mathsf{WB}}$, $\mathsf{InvRound}^{\mathsf{WB}}$

Input: $X = X_0 \| X_1 \| X_2 \| X_3 \in \mathcal{W}^4$, $RK = RK_0 \| \cdots \| RK_5 \in \mathcal{W}^6$, $f = (f_0, f_1, f_2, f_3) \in (\mathcal{P}_4^8)^4$, $g = (g_0, g_1, g_2), h = (h_0, h_1, h_2) \in (\mathcal{P}_4^8)^3$, $t = (t_0, t_1, t_2) \in (\mathcal{P}_1^8)^3$

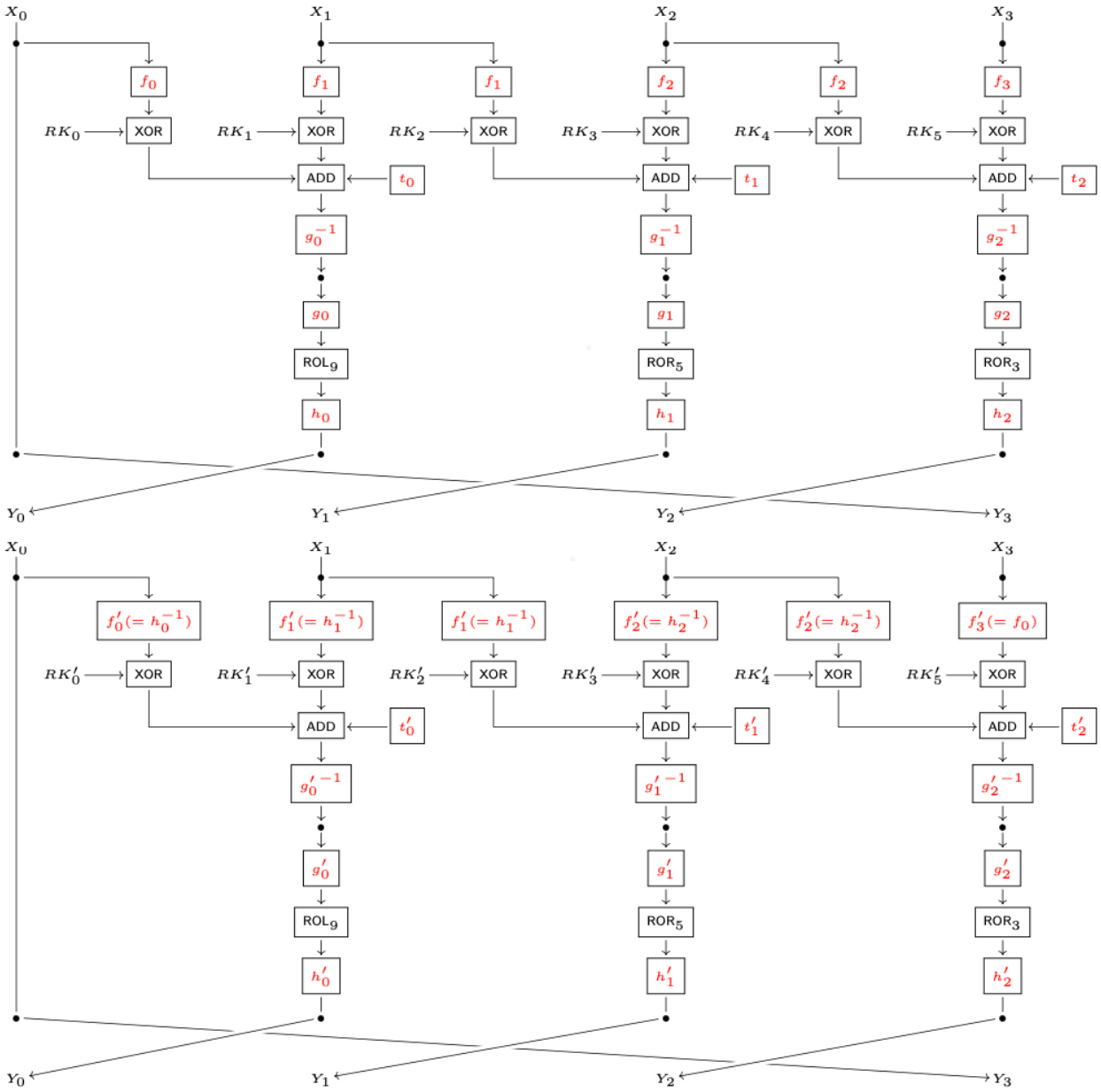
Output: $Y = Y_0 \| Y_1 \| Y_2 \| Y_3 \in \mathcal{W}^4$

procedure $\mathsf{Round}^{\mathsf{WB}}(f, g, h, t, RK; X)$
  $T \leftarrow \mathsf{ADD}^{\mathsf{WB}}(f_0, f_1, g_0^{-1}, t_0, RK_0, RK_1; X_0, X_1)$
  $Y_0 \leftarrow \mathsf{ROL}_9^{\mathsf{WB}}(g_0, h_0; T)$
  $T \leftarrow \mathsf{ADD}^{\mathsf{WB}}(f_1, f_2, g_1^{-1}, t_1, RK_2, RK_3; X_1, X_2)$
  $Y_1 \leftarrow \mathsf{ROR}_5^{\mathsf{WB}}(g_1, h_1; T)$
  $T \leftarrow \mathsf{ADD}^{\mathsf{WB}}(f_2, f_3, g_2^{-1}, t_2, RK_4, RK_5; X_2, X_3)$
  $Y_2 \leftarrow \mathsf{ROR}_3^{\mathsf{WB}}(g_2, h_2; T)$
  $Y_3 \leftarrow X_0$
  return $Y = Y_0 \| Y_1 \| Y_2 \| Y_3$
end procedure

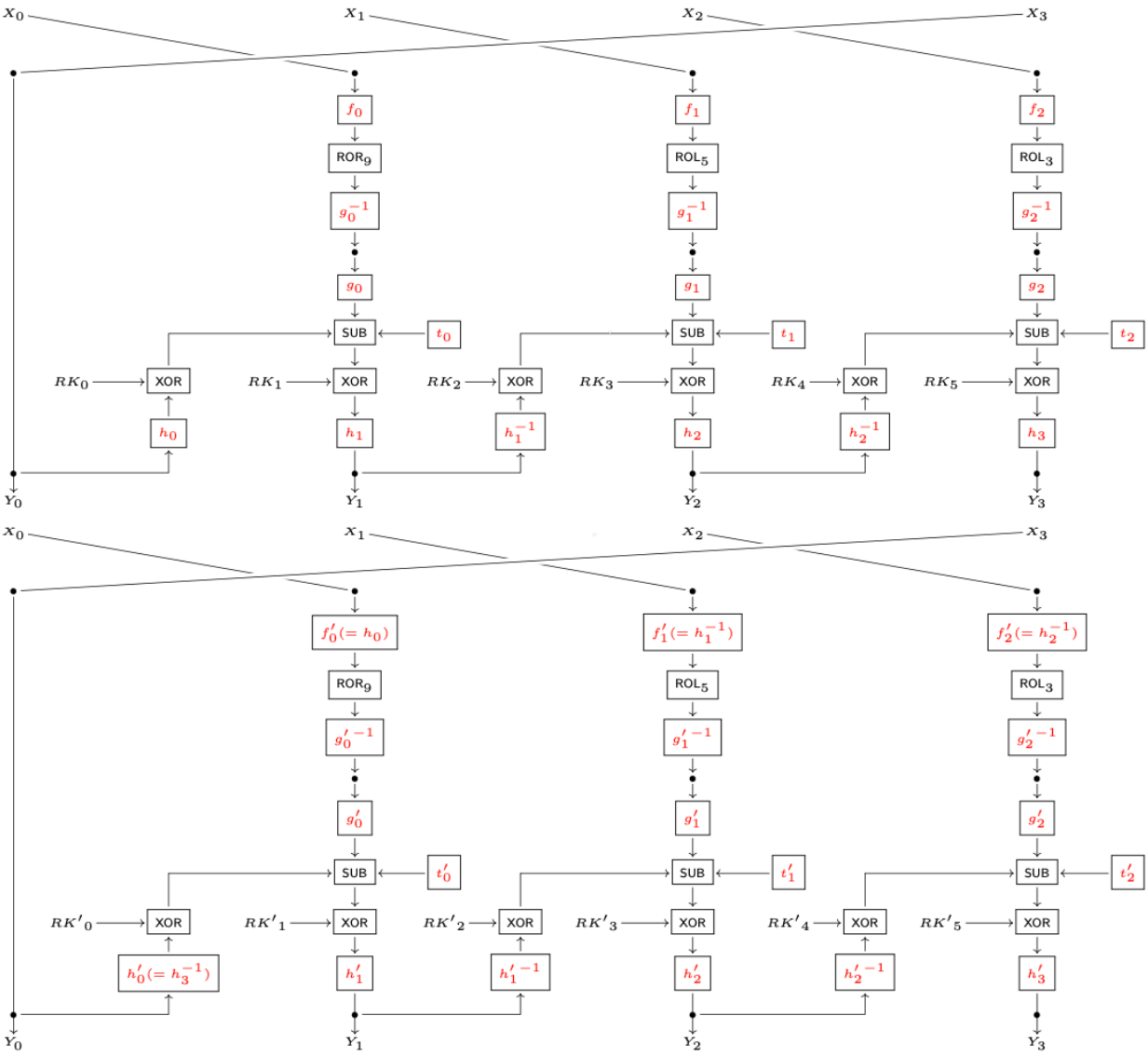
Input: $X = X_0 \| X_1 \| X_2 \| X_3 \in \mathcal{W}^4$, $RK = RK_0 \| \cdots \| RK_5 \in \mathcal{W}^6$, $f = (f_0, f_1, f_2), g = (g_0, g_1, g_2) \in (\mathcal{P}_4^8)^3$, $h = (h_0, h_1, h_2, h_3) \in (\mathcal{P}_4^8)^4$, $t = (t_0, t_1, t_2) \in (\mathcal{P}_1^8)^3$

Output: $Y = Y_0 \| Y_1 \| Y_2 \| Y_3 \in \mathcal{W}^4$

procedure $\mathsf{InvRound}^{\mathsf{WB}}(f, g, h, t, RK; X)$
  $Y_0 \leftarrow X_3$
  $T \leftarrow \mathsf{ROR}_9^{\mathsf{WB}}(f_0, g_0^{-1}; X_0)$
  $Y_1 \leftarrow \mathsf{SUB}^{\mathsf{WB}}(g_0, h_0, h_1, t_0, RK_0, RK_1; T, Y_0)$
  $T \leftarrow \mathsf{ROL}_5^{\mathsf{WB}}(f_1, g_1^{-1}; X_1)$
  $Y_2 \leftarrow \mathsf{SUB}^{\mathsf{WB}}(g_1, h_1^{-1}, h_2, t_1, RK_2, RK_3; T, Y_1)$
  $T \leftarrow \mathsf{ROL}_3^{\mathsf{WB}}(f_2, g_2^{-1}; X_2)$
  $Y_3 \leftarrow \mathsf{SUB}^{\mathsf{WB}}(g_2, h_2^{-1}, h_3, t_2, RK_4, RK_5; T, Y_2)$
  return $Y = Y_0 \| Y_1 \| Y_2 \| Y_3$
end procedure



[그림3.6: $Y \leftarrow \text{Round}^{\text{WB}}(f, g, h, t, RK; X)$]



[그림3.7: $Y \leftarrow Inv\mathrm{Round}^{\mathrm{WB}}(f, g, h, t, RK; X)$]

3.5 Encrypt$^{\mathrm{WB}}$

l라운드에서 사용하는 랜덤인코딩은 다음의 표기를 따르며, 그림 3.8은 각 랜덤인코딩의 적용 위치를 보여준다. 받아올림비트 랜덤인코딩 $t_j^{(l)}$는 ADD의 입력으로 표현했음을 주의한다.

$$
\begin{aligned}
f^{(l)} &= \left(f_0^{(l)}, f_1^{(l)}, f_2^{(l)}, f_3^{(l)}\right) \in (\mathcal{P}_4^8)^4, \\
g^{(l)} &= \left(g_0^{(l)}, g_1^{(l)}, g_2^{(l)}\right) \in (\mathcal{P}_4^8)^3, \\
h^{(l)} &= \left(h_0^{(l)}, h_1^{(l)}, h_2^{(l)}\right) \in (\mathcal{P}_4^8)^3, \\
t^{(l)} &= \left(t_0^{(l)}, t_1^{(l)}, t_2^{(l)}\right) \in (\mathcal{P}_1^8)^3.
\end{aligned}
$$

$f^{(l)}$과 $h^{(l)}$, 그리고 외부인코딩 $\mathrm{A}^{\mathrm{E}} = \mathrm{diag}(A_0, A_1, A_2, A_3)$와 $\mathrm{B}^{\mathrm{E}} = \mathrm{diag}(B_0, B_1, B_2, B_3) \in (\mathcal{P}_4^8)^4$는 다음 관계를 가진다.

$$\begin{aligned} f_3^{(l)} &= f_0^{(l-1)}, \\ \left(h_0^{(l)}, h_1^{(l)}, h_2^{(l)}\right) &= \left(\left(f_0^{(l+1)}\right)^{-1}, \left(f_1^{(l+1)}\right)^{-1}, \left(f_2^{(l+1)}\right)^{-1}\right), \\ \left(f_0^{(0)}, f_1^{(0)}, f_2^{(0)}, f_3^{(0)}\right) &= (A_0^{-1}, A_1^{-1}, A_2^{-1}, A_3^{-1}), \\ \left(h_0^{(\mathbf{r}-1)}, h_1^{(\mathbf{r}-1)}, h_2^{(\mathbf{r}-1)}, f_0^{(\mathbf{r}-1)}\right) &= (B_0^{-1}, B_1^{-1}, B_2^{-1}, B_3^{-1}). \end{aligned}$$

3.6 Decrypt$^{\mathrm{WB}}$

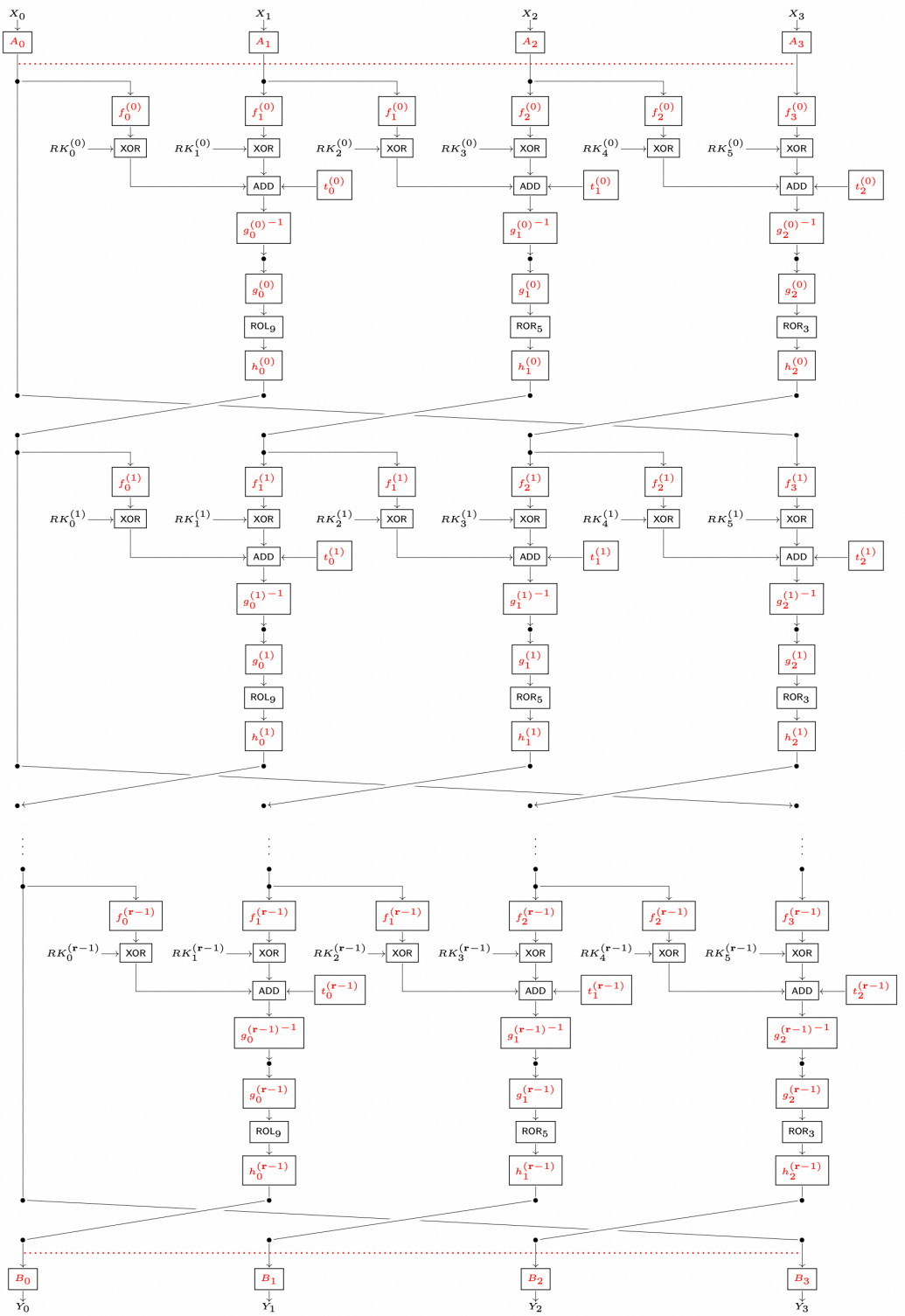
Decrypt$^{\mathrm{WB}}$의 l라운드에서 사용하는 랜덤인코딩은 다음의 표기를 따르며, 그림 3.9각 랜덤인코딩의 적용 위치를 보여준다. 받아내림비트 랜덤인코딩 $t_j^{(l)}$는 SUB의 입력으로 표현했음을 주의한다.

$$\begin{aligned} f^{(l)} &= \left(f_0^{(l)}, f_1^{(l)}, f_2^{(l)}\right) \in (\mathcal{P}_4^8)^3, \\ g^{(l)} &= \left(g_0^{(l)}, g_1^{(l)}, g_2^{(l)}\right) \in (\mathcal{P}_4^8)^3, \\ h^{(l)} &= \left(h_0^{(l)}, h_1^{(l)}, h_2^{(l)}, h_3^{(l)}\right) \in (\mathcal{P}_4^8)^4, \\ t^{(l)} &= \left(t_0^{(l)}, t_1^{(l)}, t_2^{(l)}\right) \in (\mathcal{P}_1^8)^3. \end{aligned}$$

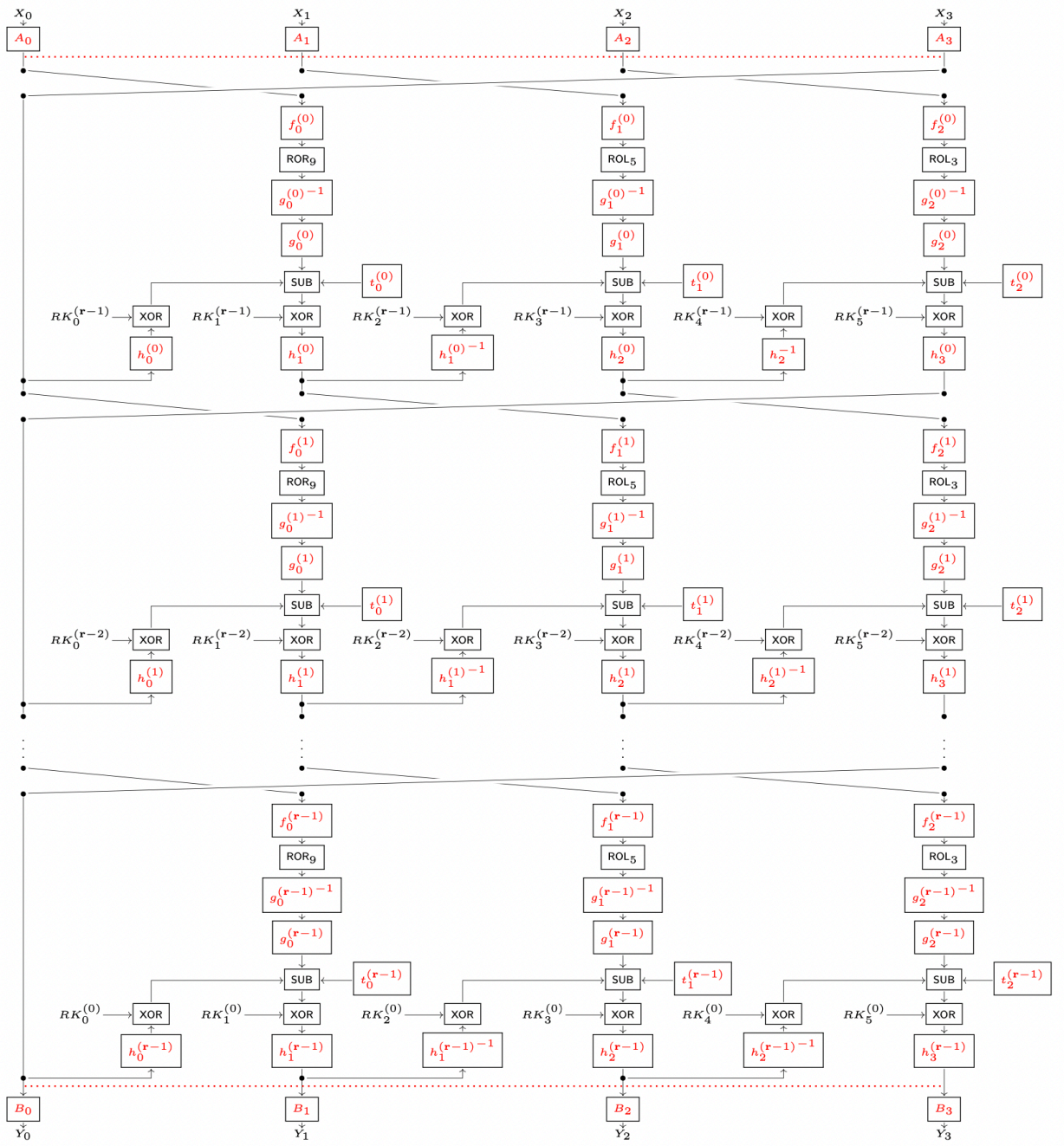
$f^{(l)}$과 $h^{(l)}$, 그리고 외부인코딩 $\mathrm{A}^{\mathrm{D}} = \mathrm{diag}(A_0, A_1, A_2, A_3)$와 $\mathrm{B}^{\mathrm{D}} = \mathrm{diag}(B_0, B_1, B_2, B_3) \in (\mathcal{P}_4^8)^4$는 다음의 관계를 가진다.

$$\begin{aligned} h_0^{(l)} &= \left(h_3^{(l-1)}\right)^{-1}, \\ \left(f_0^{(l)}, f_1^{(l)}, f_2^{(l)}\right) &= \left(h_0^{(l-1)}, \left(h_1^{(l-1)}\right)^{-1}, \left(h_2^{(l-1)}\right)^{-1}\right), \\ \left(f_0^{(0)}, f_1^{(0)}, f_2^{(0)}, h_0^{(0)}\right) &= (A_0^{-1}, A_1^{-1}, A_2^{-1}, A_3^{-1}), \\ \left(h_0^{(\mathbf{r}-1)}, h_1^{(\mathbf{r}-1)}, h_2^{(\mathbf{r}-1)}, h_3^{(\mathbf{r}-1)}\right) &= (B_0, B_1^{-1}, B_2^{-1}, B_3^{-1}), \end{aligned}$$

알고리듬 14는 랜덤인코딩을 사용하는 Decrypt$^{\mathrm{WB}}$의 동작원리를 보여준다.



[그림 3.8: $Whitebox\ WF-LEA:\ Encrypt_k^{WB}(X)$]



[그림 3.9: $Whitebox\ WF-LEA: Decrypt_K^{WB}(X)$]

알고리듬 14 Whitebox WF-LEA: $\text{Decrypt}^{\text{WB}}$

Input: 128비트 평문 X, f, g, h, t, RK

Output: 128비트 암호문 Y

procedure $\text{Decrypt}^{\text{WB}}(f, g, h, t, RK; X)$
  $X_0 \leftarrow X$
  for $l = 0$ to $\mathbf{r} - 1$ **do**
    $X_{l+1} \leftarrow \text{InvRound}^{\text{WB}}(f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)}, RK^{(\mathbf{r}-1-l)}; X_l)$
  end for
  $Y \leftarrow X_{\mathbf{r}}$
  return Y
end procedure

제 4절 네트워크 랜덤인코딩 조건식

4.1 암호 연산용

화이트박스 암호 연산 $\text{Encrypt}_K^{\text{WB}}\left(= \text{B}^{-1} \circ \text{Encrypt}_K \circ \text{A}^{-1}\right)$에 사용하는 네트워크 랜덤인코딩은 외부인코딩 A와 B를 이용하여 다음과 같이 생성한다. 이때 $f \leftarrow \mathcal{P}_n^m$은 집합 $\mathcal{P}_n^m$에서 임의로 선택함을 의미한다.

$$f^{(l)} = (f_0^{(l)}, f_1^{(l)}, f_2^{(l)}, f_3^{(l)}) \leftarrow \begin{cases} (A_0^{-1}, A_1^{-1}, A_2^{-1}, A_3^{-1}), & l = 0, \\ (\mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8, f_0^{(l-1)}), & l = 1, \ldots, \mathbf{r}-2, \\ (B_3, \mathcal{P}_4^8, \mathcal{P}_4^8, f_0^{(\mathbf{r}-2)}), & l = \mathbf{r}-1. \end{cases}$$

$$h^{(l)} = (h_0^{(l)}, h_1^{(l)}, h_2^{(l)}) \leftarrow \begin{cases} ((f_0^{(l+1)})^{-1}, (f_1^{(l+1)})^{-1}, (f_2^{(l+1)})^{-1}), & l = 0, \ldots, \mathbf{r}-2, \\ (B_0^{-1}, B_1^{-1}, B_2^{-1}), & l = \mathbf{r}-1, \end{cases}$$

$$g^{(l)} = (g_0^{(l)}, g_1^{(l)}, g_2^{(l)}) \leftarrow (\mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8), \quad l = 0, \ldots, \mathbf{r}-1,$$

$$t^{(l)} = (t_0^{(l)}, t_1^{(l)}, t_2^{(l)}) \leftarrow (\mathcal{P}_1^8, \mathcal{P}_1^8, \mathcal{P}_1^8), \quad l = 0, \ldots, \mathbf{r}-1.$$

알고리듬 15 Whitebox WF-LEA: GenNetRandEncodings4Enc

Input: 외부인코딩 $\text{A} = \text{diag}(A_0, A_1, A_2, A_3)$, $\text{B} = \text{diag}(B_0, B_1, B_2, B_3)$, 라운드 색인집합 $I = \{ind_s, \ldots, ind_e\}$
Output: $(f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)})_{l \in I}$

procedure GenNetRandEncodings4Enc(A, B, I)
  $f^{(ind_s)} = (f_0^{(ind_s)}, f_1^{(ind_s)}, f_2^{(ind_s)}, f_3^{(ind_s)}) \leftarrow (A_0^{-1}, A_1^{-1}, A_2^{-1}, A_3^{-1})$
  for $l = ind_s + 1$ to $ind_e - 1$ **do**
    $f^{(l)} = (f_0^{(l)}, f_1^{(l)}, f_2^{(l)}, f_3^{(l)}) \leftarrow (\mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8, f_0^{(l-1)})$
  end for
  $f^{(ind_e)} = (f_0^{(ind_e)}, f_1^{(ind_e)}, f_2^{(ind_e)}, f_3^{(ind_e)}) \leftarrow (B_3, \mathcal{P}_4^8, \mathcal{P}_4^8, f_0^{(ind_e-1)})$
  for $l = ind_s$ to $ind_e - 1$ **do**
    $h^{(l)} = (h_0^{(l)}, h_1^{(l)}, h_2^{(l)}) \leftarrow ((f_0^{(l+1)})^{-1}, (f_1^{(l+1)})^{-1}, (f_2^{(l+1)})^{-1})$
  end for
  $h^{(ind_e)} = (h_0^{(ind_e)}, h_1^{(ind_e)}, h_2^{(ind_e)}) \leftarrow (B_0^{-1}, B_1^{-1}, B_2^{-1})$
  for $l = ind_s$ to ind_e **do**
    $g^{(l)} = (g_0^{(l)}, g_1^{(l)}, g_2^{(l)}) \leftarrow (\mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8)$
    $t^{(l)} = (t_0^{(l)}, t_1^{(l)}, t_2^{(l)}) \leftarrow (\mathcal{P}_1^8, \mathcal{P}_1^8, \mathcal{P}_1^8)$
  end for
end procedure

4.2 복호 연산용

화이트박스 복호 연산 $\text{Decrypt}_K^{\text{WB}}\left(= \text{B}^{-1} \circ \text{Decrypt}_K \circ \text{A}^{-1}\right)$에 사용하는 네트워크 랜덤인코딩은 외부인코딩 A와 B를 이용하여 다음과 같이 생성한다.

$$h^{(l)} = (h_0^{(l)}, h_1^{(l)}, h_2^{(l)}, h_3^{(l)}) \leftarrow \begin{cases} (A_3^{-1}, \mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8), & l = 0, \\ ((h_3^{(l-1)})^{-1}, \mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8), & l = 1, \ldots, \mathbf{r}-3, \\ ((h_3^{(l-1)})^{-1}, \mathcal{P}_4^8, \mathcal{P}_4^8, B_0^{-1}), & l = \mathbf{r}-2, \\ (B_0, B_1^{-1}, B_2^{-1}, B_3^{-1}), & l = \mathbf{r}-1 \end{cases}$$

$$f^{(l)} = (f_0^{(l)}, f_1^{(l)}, f_2^{(l)}) \leftarrow \begin{cases} (A_0^{-1}, A_1^{-1}, A_2^{-1}), & l = 0, \\ (h_0^{(l-1)}, (h_1^{(l-1)})^{-1}, (h_2^{(l-1)})^{-1}), & l = 1, \ldots, \mathbf{r}-1, \end{cases}$$

$$g^{(l)} = (g_0^{(l)}, g_1^{(l)}, g_2^{(l)}) \leftarrow (\mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8), \quad l = 0, \ldots, \mathbf{r}-1,$$

$$t^{(l)} = (t_0^{(l)}, t_1^{(l)}, t_2^{(l)}) \leftarrow (\mathcal{P}_1^8, \mathcal{P}_1^8, \mathcal{P}_1^8), \quad l = 0, \ldots, \mathbf{r}-1.$$

알고리듬 16 Whitebox WF-LEA: GenNetRandEncodings4Dec

Input: 외부인코딩 $\mathsf{A} = \mathsf{diag}(A_0, A_1, A_2, A_3), \mathsf{B} = \mathsf{diag}(B_0, B_1, B_2, B_3)$, 라운드 색인집합 $I = \{ind_s, \ldots, ind_e\}$
Output: $(f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)})_{l \in I}$

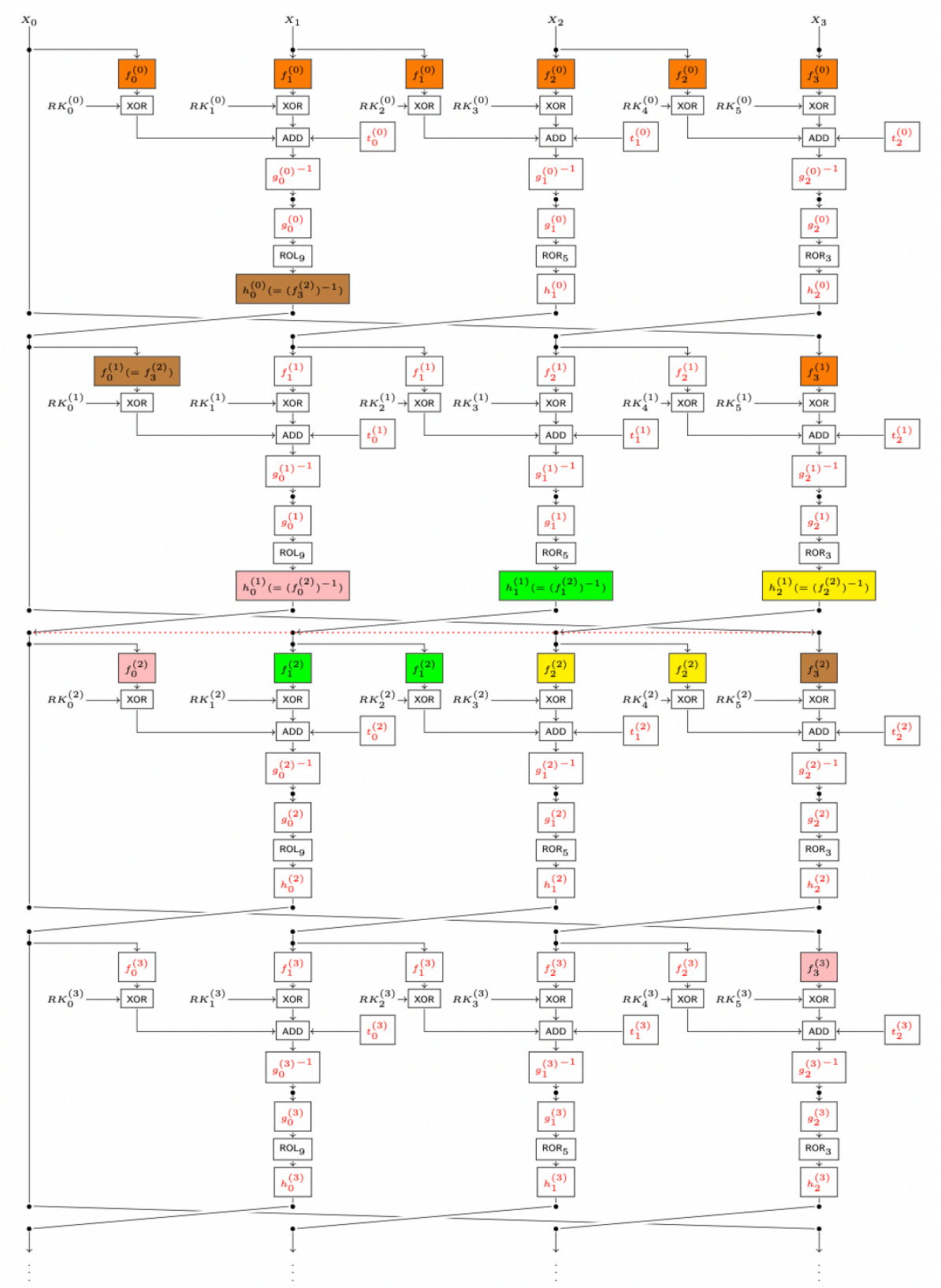
procedure GenNetRandEncodings4Dec($\mathsf{A}, \mathsf{B}, I$)
  $h^{(ind_s)} = (h_0^{(ind_s)}, h_1^{(ind_s)}, h_2^{(ind_s)}, h_3^{(ind_s)}) \leftarrow (A_3^{-1}, \mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8)$
  for $l = ind_s + 1$ to $ind_e - 2$ **do**
    $h^{(l)} = (h_0^{(l)}, h_1^{(l)}, h_2^{(l)}, h_3^{(l)}) \leftarrow ((h_3^{(l-1)})^{-1}, \mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8)$
  end for
  $h^{(ind_e-1)} = (h_0^{(ind_e-1)}, h_1^{(ind_e-1)}, h_2^{(ind_e-1)}, h_3^{(ind_e-1)}) \leftarrow ((h_3^{(l-1)})^{-1}, \mathcal{P}_4^8, \mathcal{P}_4^8, B_0^{-1})$
  $h^{(ind_e)} = (h_0^{(ind_e)}, h_1^{(ind_e)}, h_2^{(ind_e)}, h_3^{(ind_e)}) \leftarrow (B_0, B_1^{-1}, B_2^{-1}, B_3^{-1})$
  $f^{(ind_s)} = (f_0^{(ind_s)}, f_1^{(ind_s)}, f_2^{(ind_s)}) \leftarrow (A_0^{-1}, A_1^{-1}, A_2^{-1})$
  for $l = ind_s + 1$ to ind_e **do**
    $f^{(l)} = (f_0^{(l)}, f_1^{(l)}, f_2^{(l)}) \leftarrow (h_0^{(l-1)}, (h_1^{(l-1)})^{-1}, (h_2^{(l-1)})^{-1})$
  end for
  for $l = ind_s$ to ind_e **do**
    $g^{(l)} = (g_0^{(l)}, g_1^{(l)}, g_2^{(l)}) \leftarrow (\mathcal{P}_4^8, \mathcal{P}_4^8, \mathcal{P}_4^8)$
    $t^{(l)} = (t_0^{(l)}, t_1^{(l)}, t_2^{(l)}) \leftarrow (\mathcal{P}_1^8, \mathcal{P}_1^8, \mathcal{P}_1^8)$
  end for
end procedure

알고리듬 17은 랜덤인코딩을 사용하는 $\mathsf{Encrypt}^{\mathsf{WB}}$의 동작원리를 보여준다.

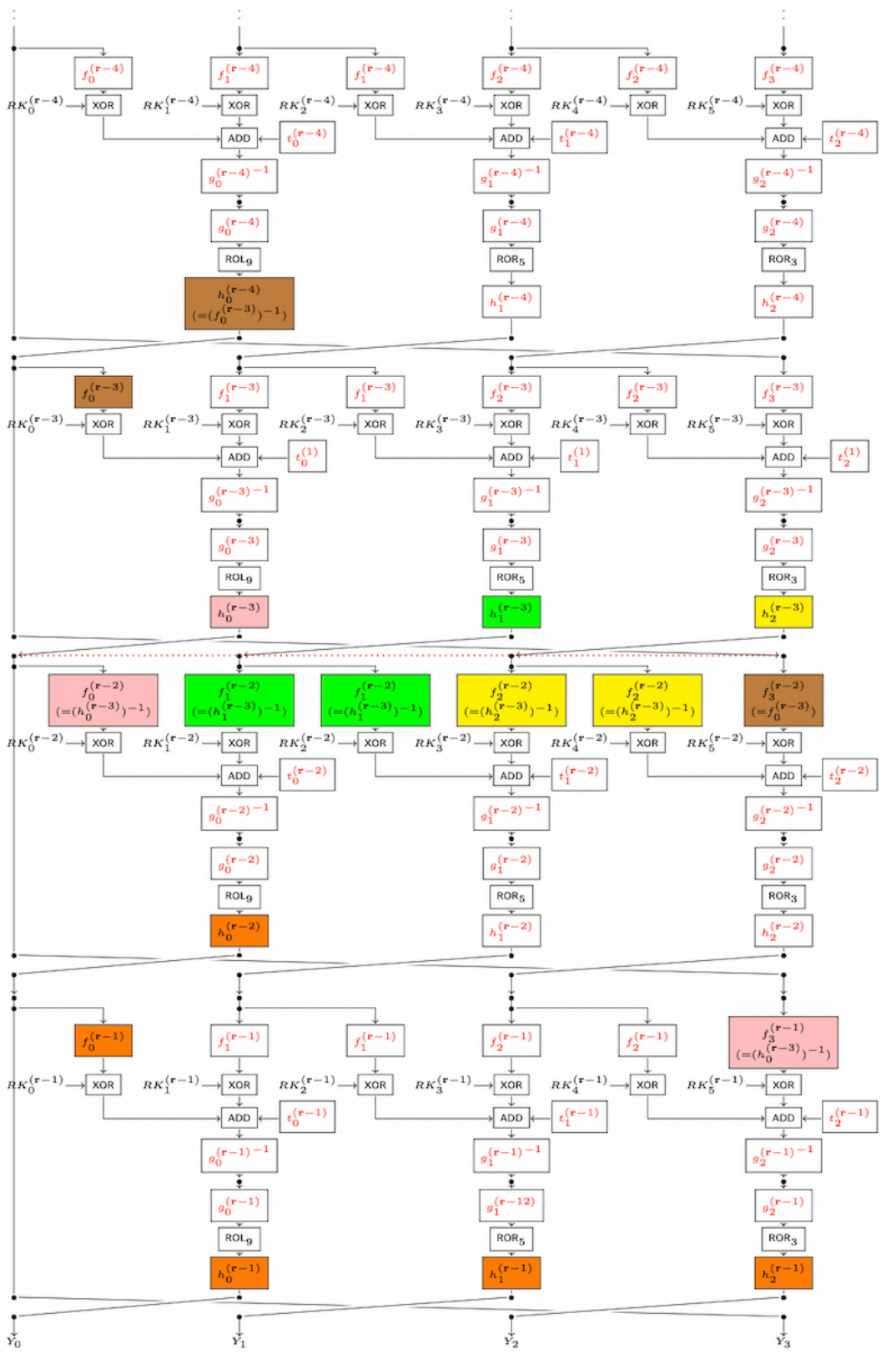
알고리듬 17 Whitebox WF-LEA: $\mathsf{Encrypt}_K^{\mathsf{WB}}$

Input: 128비트 평문 X, f, g, h, t, RK
Output: 128비트 암호문 Y

procedure $\mathsf{Encrypt}^{\mathsf{WB}}(f, g, h, t, RK; X)$
  $X_0 \leftarrow X$
  for $l = 0$ to $\mathbf{r}-1$ **do**
    $X_{l+1} \leftarrow \mathsf{Round}^{\mathsf{WB}}(f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)}, RK^{(l)}; X_l)$
  end for
  $Y \leftarrow X_{\mathbf{r}}$
  return Y
end procedure



[그림 3.10: $Whitebox\ WF-LEA$: $Encrypt_K^{WB}(X)$의 첫 4라운드 연산]



[그림 3.11: $Whitebox\ WF-LEA$: $Encrypt_K^{WB}(X)$의 마지막 4라운드 연산]

제 5절 테이블 참조 구현

Whitebox WF-LEA의 연산을 위해 다음을 준비한다.

(준비 1) 외부인코딩 A,B 생성 및 저장

(준비 2) 각 라운드 함수에 사용하는 네트워크 랜덤인코딩 $f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)}$ 생성

(준비 3) 키 K로부터 $\mathbf{r}$개의 192비트 라운드키 $RK^{(0)}, \ldots, RK^{(\mathbf{r}-1)}$ 생성

(준비 4) 함수 $\mathrm{add}^{\mathrm{WB}}, \mathrm{rol}_1^{\mathrm{WB}}, \mathrm{ror}_1^{\mathrm{WB}}, \mathrm{ror}_3^{\mathrm{WB}}$의 모든 입/출력을 계산해서 테이블로 저장 (화이트박스 키테이블)

함수 $\mathrm{ADD}^{\mathrm{WB}}, \mathrm{ROL}_1^{\mathrm{WB}}, \mathrm{ROR}_1^{\mathrm{WB}}, \mathrm{ROR}_3^{\mathrm{WB}}$를 화이트박스 키테이블 참조 방식으로 계산하여 암호 (또는 복호) 연산을 수행한다.

5.1 $\mathrm{add}^{\mathrm{WB}}$, $\mathrm{sub}^{\mathrm{WB}}$, $\mathrm{rol}^{\mathrm{WB}}$, $\mathrm{ror}^{\mathrm{WB}}$ 테이블 구현

함수 $\mathrm{add}^{\mathrm{WB}}, \mathrm{sub}^{\mathrm{WB}}, \mathrm{rol}^{\mathrm{WB}}, \mathrm{ror}^{\mathrm{WB}}$의 모든 입/출력을 사전계산하여 테이블로 저장한다.

5.1.1 암호 연산용 화이트박스 키테이블

암호 연산용 l번째 라운드 랜덤인코딩과 라운드키는 다음 표기를 따른다.

$$
\begin{aligned}
f^{(l)} &= \left(f_0^{(l)}, f_1^{(l)}, f_2^{(l)}, f_3^{(l)}\right) \in (\mathcal{P}_4^8)^4, \\
g^{(l)} &= \left(g_0^{(l)}, g_1^{(l)}, g_2^{(l)}\right) \in (\mathcal{P}_4^8)^3, \\
h^{(l)} &= \left(h_0^{(l)}, h_1^{(l)}, h_2^{(l)}\right) \in (\mathcal{P}_4^8)^3, \\
t^{(l)} &= \left(t_0^{(l)}, t_1^{(l)}, t_2^{(l)}\right) \in (\mathcal{P}_1^8)^3, \\
RK^{(l)} &= RK_0^{(l)} \parallel \cdots \parallel RK_5^{(l)} \in \mathcal{W}^6,
\end{aligned}
$$

where

$$
\begin{aligned}
f_i^{(l)} &= \mathrm{diag}\left(f_{i,7}^{(l)}, \ldots, f_{i,0}^{(l)}\right) \in \mathcal{P}_4^8, && i = 0,1,2,3, \\
g_i^{(l)} &= \mathrm{diag}\left(g_{i,7}^{(l)}, \ldots, g_{i,0}^{(l)}\right) \in \mathcal{P}_4^8, && i = 0,1,2, \\
h_i^{(l)} &= \mathrm{diag}\left(h_{i,7}^{(l)}, \ldots, h_{i,0}^{(l)}\right) \in \mathcal{P}_4^8, && i = 0,1,2, \\
t_i^{(l)} &= \mathrm{diag}\left(t_{i,7}^{(l)}, \ldots, t_{i,0}^{(l)}\right) \in \mathcal{P}_1^8, && i = 0,1,2, \\
RK_i^{(l)} &= RK_{i,7}^{(l)} \parallel \cdots \parallel RK_{i,0}^{(l)} \in \mathcal{N}^8, && i = 0,1,2,3,4,5.
\end{aligned}
$$

$l \in [\mathbf{r}], i \in [3], j \in [8]$에 대해 다음 연산의 모든 입/출력을 사전계산하여 테이블 $\mathrm{ETA}_{i,j}^{(l)}, \mathrm{ETR}_{i,j}^{(l)}$로 저장한다.

$$
\begin{aligned}
c, Z &\leftarrow \mathrm{ETA}_{0,j}^{(l)}(c, X, Y) &&:= \mathrm{add}^{\mathrm{WB}}\left(t_{0,j-1}^{(l)}, t_{0,j}^{(l)}, f_{0,j}^{(l)}, f_{1,j}^{(l)}, \left(g_{0,j}^{(l)}\right)^{-1}, RK_{0,j}^{(l)}, RK_{1,j}^{(l)}; c, X, Y\right),\\
c, Z &\leftarrow \mathrm{ETA}_{1,j}^{(l)}(c, X, Y) &&:= \mathrm{add}^{\mathrm{WB}}\left(t_{1,j-1}^{(l)}, t_{1,j}^{(l)}, f_{1,j}^{(l)}, f_{2,j}^{(l)}, \left(g_{1,j}^{(l)}\right)^{-1}, RK_{2,j}^{(l)}, RK_{3,j}^{(l)}; c, X, Y\right),\\
c, Z &\leftarrow \mathrm{ETA}_{2,j}^{(l)}(c, X, Y) &&:= \mathrm{add}^{\mathrm{WB}}\left(t_{2,j-1}^{(l)}, t_{2,j}^{(l)}, f_{2,j}^{(l)}, f_{3,j}^{(l)}, \left(g_{2,j}^{(l)}\right)^{-1}, RK_{4,j}^{(l)}, RK_{5,j}^{(l)}; c, X, Y\right),\\
Z &\leftarrow \mathrm{ETR}_{0,j}^{(l)}(X, Y) &&:= \mathrm{rol}_1^{\mathrm{WB}}\left(g_{0,j-2 \bmod 8}^{(l)}, g_{0,j-3 \bmod 8}^{(l)}, h_{0,j}^{(l)}; X, Y\right),\\
Z &\leftarrow \mathrm{ETR}_{1,j}^{(l)}(X, Y) &&:= \mathrm{ror}_1^{\mathrm{WB}}\left(g_{1,j+2 \bmod 8}^{(l)}, g_{1,j+1 \bmod 8}^{(l)}, h_{1,j}^{(l)}; X, Y\right),\\
Z &\leftarrow \mathrm{ETR}_{2,j}^{(l)}(X, Y) &&:= \mathrm{ror}_3^{\mathrm{WB}}\left(g_{2,j+1 \bmod 8}^{(l)}, g_{2,j \bmod 8}^{(l)}, h_{2,j}^{(l)}; X, Y\right).
\end{aligned}
$$

$\mathrm{ETA}_{i,j}^{(l)}$은 9비트 입력, 5비트 출력 테이블이며, $\mathrm{ETR}_{i,j}^{(l)}$는 8비트 입력, 4비트 출력 테이블이다.

알고리듬 18 Whitebox WF-LEA: GenWKE

Input: 외부인코딩 A, B, 키 K, 색인집합 $I_j = \{ind_s, \ldots, ind_e\}$

Output: 화이트박스 암호키 테이블 WKE (used in $\mathsf{B}^{-1} \circ \mathsf{Encrypt}_K{}^{(j)} \circ \mathsf{A}^{-1}$)

1: **procedure** GenWKE(A, B, K, I_j)
2: $(RK^{(l)})_{l \in I} \leftarrow$ KeySchedule(K, I_j)
3: $(f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)})_{l \in I} \leftarrow$ GenNetRandEncodings4Enc(A, B, I_j)
4: Generate $WKE = (\mathsf{ETA}_{i,j}^{(l)}, \mathsf{ETR}_{i,j}^{(l)})_{i \in [3], j \in [8], l \in I}$ derived from $(RK^{(l)})_{l \in I}$ and $(f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)})_{l \in I}$
5: **return** WKE
6: **end procedure**

5.1.2 복호 연산용 화이트박스 키테이블

복호 연산용 r라운드 랜덤인코딩과 라운드키는 다음의 표기를 따른다.

$$
\begin{aligned}
f^{(l)} &= \left(f_0^{(l)}, f_1^{(l)}, f_2^{(l)}\right) \in (\mathcal{P}_4^8)^3,\\
g^{(l)} &= \left(g_0^{(l)}, g_1^{(l)}, g_2^{(l)}\right) \in (\mathcal{P}_4^8)^3,\\
h^{(l)} &= \left(h_0^{(l)}, h_1^{(l)}, h_2^{(l)}, h_3^{(l)}\right) \in (\mathcal{P}_4^8)^4,\\
t^{(l)} &= \left(t_0^{(l)}, t_1^{(l)}, t_2^{(l)}\right) \in (\mathcal{P}_1^8)^3,\\
RK^{(l)} &= RK_0^{(l)} \parallel \cdots \parallel RK_5^{(l)} \in \mathcal{W}^6,
\end{aligned}
$$

where

$$
\begin{aligned}
f_i^{(l)} &= \mathrm{diag}\left(f_{i,7}^{(l)}, \ldots, f_{i,0}^{(l)}\right) \in \mathcal{P}_4^8, && i = 0,1,2,\\
g_i^{(l)} &= \mathrm{diag}\left(g_{i,7}^{(l)}, \ldots, g_{i,0}^{(l)}\right) \in \mathcal{P}_4^8, && i = 0,1,2,\\
h_i^{(l)} &= \mathrm{diag}\left(h_{i,7}^{(l)}, \ldots, h_{i,0}^{(l)}\right) \in \mathcal{P}_4^8, && i = 0,1,2,3,\\
t_i^{(l)} &= \mathrm{diag}\left(t_{i,7}^{(l)}, \ldots, t_{i,0}^{(l)}\right) \in \mathcal{P}_1^8, && i = 0,1,2,\\
RK_i^{(l)} &= RK_{i,7}^{(l)} \parallel \cdots \parallel RK_{i,0}^{(l)} \in \mathcal{N}^8, && i = 0,1,2,3,4,5.
\end{aligned}
$$

$l \in [\mathbf{r}], i \in [3], j \in [8]$에 대해 다음 연산의 모든 입/출력을 사전계산하여 테이블 $\mathrm{DTS}_{i,j}^{(l)}, \mathrm{DTR}_{i,j}^{(l)}$로 저장한다.

$$
\begin{aligned}
Z &\leftarrow \mathrm{DTR}_{0,j}^{(l)}(X,Y) &&:= \mathrm{ror}_1^{\mathrm{WB}}\left(f_{0,j+3 \bmod 8}^{(l)}, f_{0,j+2 \bmod 8}^{(l)}, \left(g_{0,j}^{(l)}\right)^{-1}; X, Y\right),\\
Z &\leftarrow \mathrm{DTR}_{1,j}^{(l)}(X,Y) &&:= \mathrm{rol}_1^{\mathrm{WB}}\left(f_{1,j-1 \bmod 8}^{(l)}, f_{1,j-2 \bmod 8}^{(l)}, \left(g_{1,j}^{(l)}\right)^{-1}; X, Y\right),\\
Z &\leftarrow \mathrm{DTR}_{2,j}^{(l)}(X,Y) &&:= \mathrm{rol}_3^{\mathrm{WB}}\left(f_{2,j \bmod 8}^{(l)}, f_{2,j-1 \bmod 8}^{(l)}, \left(g_{2,j}^{(l)}\right)^{-1}; X, Y\right),\\
b, Z &\leftarrow \mathrm{DTS}_{0,j}^{(l)}(b,X,Y) &&:= \mathrm{sub}^{\mathrm{WB}}\left(t_{0,j-1}^{(l)}, t_{0,j}^{(l)}, g_{0,j}^{(l)}, h_{0,j}^{(l)}, h_{1,j}^{(l)}, RK_{0,j}^{(\mathbf{r}-1-l)}, RK_{1,j}^{(\mathbf{r}-1-l)}; b, X, Y\right),\\
b, Z &\leftarrow \mathrm{DTS}_{1,j}^{(l)}(b,X,Y) &&:= \mathrm{sub}^{\mathrm{WB}}\left(t_{1,j-1}^{(l)}, t_{1,j}^{(l)}, g_{1,j}^{(l)}, \left(h_{1,j}^{(l)}\right)^{-1}, h_{2,j}^{(l)}, RK_{2,j}^{(\mathbf{r}-1-l)}, RK_{3,j}^{(\mathbf{r}-1-l)}; b, X, Y\right),\\
b, Z &\leftarrow \mathrm{DTS}_{2,j}^{(l)}(b,X,Y) &&:= \mathrm{sub}^{\mathrm{WB}}\left(t_{2,j-1}^{(l)}, t_{2,j}^{(l)}, g_{2,j}^{(l)}, \left(h_{2,j}^{(l)}\right)^{-1}, h_{3,j}^{(l)}, RK_{4,j}^{(\mathbf{r}-1-l)}, RK_{5,j}^{(\mathbf{r}-1-l)}; b, X, Y\right).
\end{aligned}
$$

$\mathrm{DTS}_{i,j}^{(l)}$은 9비트 입력, 5비트 출력 테이블이며, $\mathrm{DTR}_{i,j}^{(l)}$는 8비트 입력, 4비트 출력 테이블이다.

5.1.3 테이블 크기

표 3.1은Whitebox WF-LEA의 요소별 화이트박스 키테이블 크기를 보여준다.

알고리듬 19 Whitebox WF-LEA: GenWKD

Input: 외부인코딩 A, B, 키 K, 색인집합 $I_j = \{ind_s, \ldots, ind_e\}$
Output: 화이트박스 복호 테이블 WKE (used in $\mathsf{B}^{-1} \circ \mathsf{Decrypt}_K{}^{(j)} \circ \mathsf{A}^{-1}$)

1: **procedure** GenWKD(A, B, K, I_j)
2: $(RK^{(l)})_{l \in I} \leftarrow$ KeySchedule(K, I_j)
3: $(f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)})_{l \in I} \leftarrow$ GenNetRandEncodings4Dec(A, B, I_j)
4: Generate $WKD = (\mathrm{DTS}_{i,j}^{(l)}, \mathrm{DTR}_{i,j}^{(l)})_{i \in [3], j \in [8], l \in I}$ derived from $(RK^{(l)})_{l \in I}$ and $(f^{(l)}, g^{(l)}, h^{(l)}, t^{(l)})_{l \in I}$
5: **return** WKD
6: **end procedure**

[표 3.1 Whitebox WF-LEA 요소별 화이트박스 키테이블 크기 ($i \in [3]$, $j \in [8]$, $l \in [\boldsymbol{r}]$)]

테이블	입력 비트	출력 비트	테이블 크기 (바이트)	비고
$c, Z \leftarrow \mathsf{ETA}_{i,j}^{(l)}(c, X, Y)$	9비트	5비트	$2^9 (= 512)$	5비트를 1바이트에 저장
$Z \leftarrow \mathsf{ETR}_{i,j}^{(l)}(X, Y)$	8비트	4비트	$2^8 (= 256)$	4비트를 1바이트에 저장
$b, Z \leftarrow \mathsf{DTS}_{i,j}^{(l)}(c, X, Y)$	9비트	5비트	$2^9 (= 512)$	5비트를 1바이트에 저장
$Z \leftarrow \mathsf{DTR}_{i,j}^{(l)}(X, Y)$	8비트	4비트	$2^8 (= 256)$	4비트를 1바이트에 저장

5.2 $\mathsf{Round}^{\mathsf{WB}}$, $\mathsf{InvRound}^{\mathsf{WB}}$, $\mathsf{Encrypt}^{\mathsf{WB}}$, $\mathsf{Decrypt}^{\mathsf{WB}}$

알고리듬 20은 화이트박스 키테이블 참조 방식으로 동작하는 Whitebox WF-LEA의 라운드 함수 연산 $\mathsf{Round}^{\mathsf{WB}}$와 $\mathsf{InvRound}^{\mathsf{WB}}$을 보여준다.

알고리듬 20 Whitebox WF-LEA: 테이블 참조 방식의 l번째 라운드 함수 $\mathsf{Round}_l^{\mathsf{WB}}$와 $\mathsf{InvRound}_l^{\mathsf{WB}}$

Input: $X = X_0 \| X_1 \| X_2 \| X_3 \in \mathcal{W}^4$, $(\mathsf{ETA}_{i,j}^{(l)}, \mathsf{ETR}_{i,j}^{(l)})_{i\in[3],j\in[8]}$ (또는 $(\mathsf{DTS}_{i,j}^{(l)}, \mathsf{DTR}_{i,j}^{(l)})_{i\in[3],j\in[8]}$)
Output: $Y = Y_0 \| Y_1 \| Y_2 \| Y_3 \in \mathcal{W}^4$

procedure $\mathsf{Round}_l^{\mathsf{WB}}((\mathsf{ETA}_{i,j}^{(l)}, \mathsf{ETR}_{i,j}^{(l)})_{i\in[3],j\in[8]}; X)$
  $c \leftarrow 0$
  for $j = 0$ to 7 **do**
    $c, T_j \leftarrow \mathsf{ETA}_{0,j}^{(l)}(c, X_{0,j}, X_{1,j})$
  end for
  for $j = 0$ to 7 **do**
    $Y_{0,j} \leftarrow \mathsf{ETR}_{0,j}^{(l)}(T_{j-2 \bmod 8}, T_{j-3 \bmod 8})$
  end for
  $c \leftarrow 0$
  for $j = 0$ to 7 **do**
    $c, T_j \leftarrow \mathsf{ETA}_{1,j}^{(l)}(c, X_{1,j}, X_{2,j})$
  end for
  for $j = 0$ to 7 **do**
    $Y_{1,j} \leftarrow \mathsf{ETR}_{1,j}^{(l)}(T_{j+2 \bmod 8}, T_{j+1 \bmod 8})$
  end for
  $c \leftarrow 0$
  for $j = 0$ to 7 **do**
    $c, T_j \leftarrow \mathsf{ETA}_{2,j}^{(l)}(c, X_{2,j}, X_{3,j})$
  end for
  for $j = 0$ to 7 **do**
    $Y_{2,j} \leftarrow \mathsf{ETR}_{2,j}^{(l)}(T_{j+1 \bmod 8}, T_{j \bmod 8})$
  end for
  $Y_3 \leftarrow X_0$
  return $Y = Y_0 \| Y_1 \| Y_2 \| Y_3$
end procedure

procedure $\mathsf{InvRound}_l^{\mathsf{WB}}((\mathsf{DTS}_{i,j}^{(l)}, \mathsf{DTR}_{i,j}^{(l)})_{i\in[3],j\in[8]}; X)$
  $Y_0 \leftarrow X_3$
  for $j = 0$ to 7 **do**
    $T_j \leftarrow \mathsf{DTR}_{0,j}^{(l)}(X_{0,j+3 \bmod 8}, X_{0,j+2 \bmod 8})$
  end for
  $b \leftarrow 0$
  for $j = 0$ to 7 **do**
    $b, Y_{1,j} \leftarrow \mathsf{DTS}_{0,j}^{(l)}(b, T_j, Y_{0,j})$
  end for
  for $j = 0$ to 7 **do**
    $T_j \leftarrow \mathsf{DTR}_{1,j}^{(l)}(X_{1,j-1 \bmod 8}, X_{1,j-2 \bmod 8})$
  end for
  $b \leftarrow 0$
  for $j = 0$ to 7 **do**
    $b, Y_{2,j} \leftarrow \mathsf{DTS}_{1,j}^{(l)}(b, T_j, Y_{1,j})$
  end for
  for $j = 0$ to 7 **do**
    $T_j \leftarrow \mathsf{DTR}_{2,j}^{(l)}(X_{2,j \bmod 8}, X_{2,j-1 \bmod 8})$
  end for
  $b \leftarrow 0$
  for $j = 0$ to 7 **do**
    $b, Y_{3,j} \leftarrow \mathsf{DTS}_{2,j}^{(l)}(b, T_j, Y_{2,j})$
  end for
  return $Y = Y_0 \| Y_1 \| Y_2 \| Y_3$
end procedure

알고리듬 21 은 화이트박스 키테이블 참조 방식의 라운드 함수를 이용한 Whitebox WF-LEA의 암호 연산과 복호 연산을 보여준다.

알고리듬 21 Whitebox WF-LEA: $\mathsf{Encrypt}^{\mathsf{WB}}$와 $\mathsf{Decrypt}^{\mathsf{WB}}$

Input: 128비트 데이터 $X \in \mathcal{W}^4$, $(\mathsf{ETA}_{i,j}^{(l)}, \mathsf{ETR}_{i,j}^{(l)})_{i\in[3],j\in[8],l\in[\mathbf{r}]}$ (또는 $(\mathsf{DTS}_{i,j}^{(l)}, \mathsf{DTR}_{i,j}^{(l)})_{i\in[3],j\in[8],l\in[\mathbf{r}]}$)
Output: 128비트 데이터 $Y \in \mathcal{W}^4$

procedure $\mathsf{Encrypt}^{\mathsf{WB}}((\mathsf{ETA}_{i,j}^{(l)}, \mathsf{ETR}_{i,j}^{(l)})_{i\in[3],j\in[8],l\in[\mathbf{r}]}; X)$
  $X_0 \leftarrow X$
  for $l = 0$ to $\mathbf{r} - 1$ **do**
    $X_{l+1} \leftarrow \mathsf{Round}_l^{\mathsf{WB}}((\mathsf{ETA}_{i,j}^{(l)}, \mathsf{ETR}_{i,j}^{(l)})_{i\in[3],j\in[8]}; X_l)$
  end for
  $Y \leftarrow X_{\mathbf{r}}$
  return Y
end procedure

procedure $\mathsf{Decrypt}^{\mathsf{WB}}((\mathsf{DTS}_{i,j}^{(l)}, \mathsf{DTR}_{i,j}^{(l)})_{i\in[3],j\in[8],l\in[\mathbf{r}]}); X)$
  $X_0 \leftarrow X$
  for $l = 0$ to $\mathbf{r} - 1$ **do**
    $X_{l+1} \leftarrow \mathsf{InvRound}_l^{\mathsf{WB}}((\mathsf{DTS}_{i,j}^{(l)}, \mathsf{DTR}_{i,j}^{(l)})_{i\in[3],j\in[8]}); X_l)$
  end for
  $Y \leftarrow X_{\mathbf{r}}$
  return Y
end procedure

제 6절 성능

6.1 화이트박스 키테이블 크기

표 3.2는 연산에 필요한 전체 화이트박스 키테이블 크기를 보여준다.

[표 3.2: Whitebox WF-LEA 암호 (또는 복호) 연산용 참조 테이블 크기 (단위: 바이트)]

분류	바이너리 파일	(n-byte)	Base64 인코딩 텍스트 파일 ($4\lceil\frac{n}{3}\rceil$-byte)	gzip 파일
Whitebox WF-LEA 한 라운드	18,432	$(= (512 + 256) \times 3 \times 8)$	24,576	
Whitebox WF-LEA128	442,368	$(= 18,432 \times 24)$	589,824	268,834
Whitebox WF-LEA192	516,096	$(= 18,432 \times 28)$	688,128	315,312
Whitebox WF-LEA256	589,824	$(= 18,432 \times 32)$	786,432	359,625

6.2 속도

표 3.3은 다음 환경에서 측정한 연산 속도 성능을 보여준다.

- Device: MacBook Pro (13-inch, M1, 8GB RAM, 2020), macOS Monterey v12.4
- Programming Language: C

[표 3.3 : Whitebox WF-LEA: 속도 성능 (단위: 초)]

알고리듬	언어	외부인코딩(A,B) 생성	화이트박스 키테이블생성	암호연산	복호연산
Whitebox WF-LEA128	C	0.04	0.80	0.24	0.399

Whitebox WF-LEA192	C	0.05	1.03	0.29	0.48
Whitebox WF-LEA256	C	0.04	1.10	0.37	0.56

†100회 연산 속도, ‡100,000회 연산 속도

제 4장 Whitebox WF-LEA기반 기밀성 운영모드

Whitebox WF-LEA기반 데이터 암호 연산과 복호 연산 과정은 다음과 같다.

$$\underbrace{P \xrightarrow{\text{Encode}} EP \xrightarrow{\text{Encrypt}^{WB}} C}_{\text{암호 연산}} \quad \Rightarrow \quad \underbrace{C \xrightarrow{\text{Decrypt}^{WB}} EP \xrightarrow{\text{Decode}} P}_{\text{복호 연산}}.$$

이때, 데이터 인코딩 Encode와 디코딩 Decode는 화이트박스 공격이 불가한 영역에서 수행한다.

제 1절 평문 확장

평문길이가 블록길이의 배수가 되도록 평문을 다음과 같이 PKCS#7 패딩 규약으로 확장한다.

$$P \xrightarrow{\text{ExpMsg}} EP = P \| (\text{I2B}_8(l))^l, \quad (l = \mathbf{b} - (\text{ByteLen}(P) \bmod \mathbf{b})).$$

알고리듬 22는 평문 확장의 유사부호이다.

알고리듬 22 평문 확장 ExpMsg

Input: 평문 P
Output: 확장 평문 EP

```
1: procedure ExpMsg(P)
2:     l ← b − (ByteLen(P) mod b)
3:     EP ← P||(I2B_8(l))^l
4:     return EP
5: end procedure
```

Example. $\mathbf{b} = 16$일 때,

1. $\text{ByteLen}(P) \bmod 16 = 12$.

$$P \xrightarrow{\text{ExpMsg}} EP = P \| \text{0x04040404}.$$

2. $\text{ByteLen}(P) \bmod 16 = 0$.

$$P \xrightarrow{\text{ExpMsg}} EP = P \| \text{0x10101010101010101010101010101010}.$$

제 2절 WB-CBC

알고리듬 23은CBC 운영모드이다.

알고리듬 23 운영모드 CBC: CBC-Encrypt, CBC-Decrypt

Input: 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 평문 P
Output: 암호문 C

procedure CBC-Encrypt(IV, K, P)
  $EP = (EP_j)_{j\in[w]} \leftarrow \mathsf{ExpMsg}(P)$
  $T \leftarrow IV$
  for $j = 0$ to $w-1$ **do**
    $T \leftarrow T \oplus EP_j$
    $T \leftarrow \mathsf{Encrypt}_K(T)$
    $C_j \leftarrow T$
  end for
  return $C = C_0 \| C_1 \| \cdots \| C_{w-1}$
end procedure

Input: 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 암호문 C
Output: 평문 P

procedure CBC-Decrypt(IV, K, C)
  $C_{-1} \leftarrow IV$
  for $j = 0$ to $w-1$ **do**
    $T \leftarrow \mathsf{Decrypt}_K(C_j)$
    $EP_j \leftarrow T \oplus C_{j-1}$
  end for
  $P \leftarrow$ Remove the padding of $EP = (EP_j)_{j\in[w]}$
  return P
end procedure

알고리듬 24는Whitebox WF-LEA기반 WB-CBC 운영모드이고, 그림 4.1은 이를 도식화한 것이다.

알고리듬 24 화이트박스 운영모드 WB-CBC: $\mathsf{CBC\text{-}Encrypt}^{\mathsf{WB}} \circ \mathsf{Encode}$, $\mathsf{Decode} \circ \mathsf{CBC\text{-}Decrypt}^{\mathsf{WB}}$

Input: 치환 $\mathsf{U} \in (\mathcal{P}_4^8)^4$, 평문 P
Output: 인코딩 평문 $EP = (EP_j)_{j\in[w]}$ $(EP_j \in \mathcal{B}^{\mathbf{b}})$

procedure Encode(U, P)
  $EP = (EP_j)_{j\in[w]} \leftarrow \mathsf{ExpMsg}(P)$
  $EP_j \leftarrow \mathsf{U}(EP_j)$ for $j \in [w]$
  return EP
end procedure

Input: 치환 $\mathsf{V} \in (\mathcal{P}_4^8)^4$, 인코딩 평문 $EP = (EP_j)_{j\in[w]}$ $(EP_j \in \mathcal{B}^{\mathbf{b}})$
Output: 평문 P

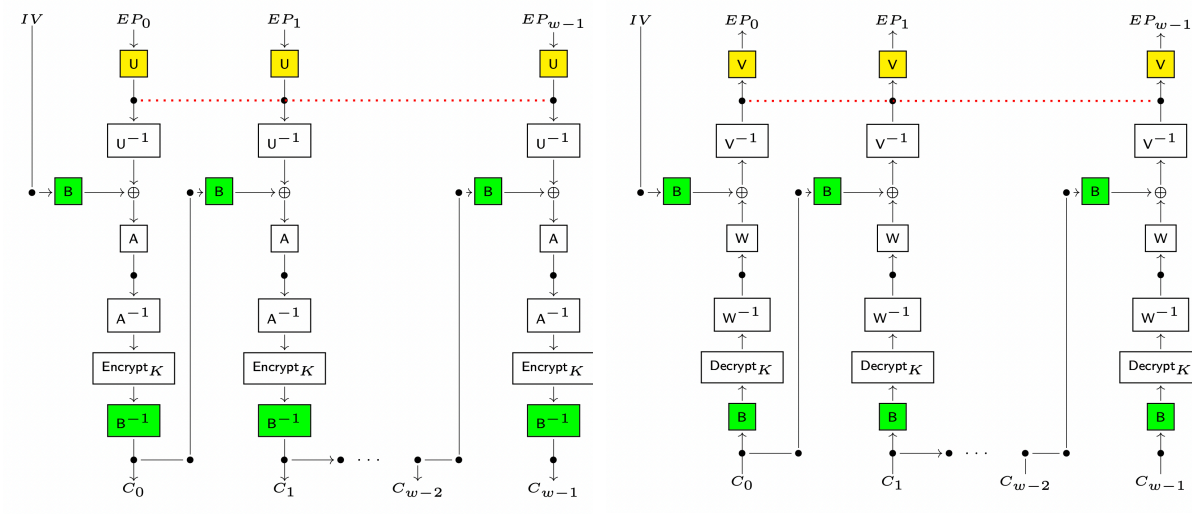
procedure Decode(V, P)
  $EP_j \leftarrow \mathsf{V}(EP_j)$ for $j \in [w]$
  $P \leftarrow$ Remove the padding of EP
  return P
end procedure

Input: 치환 $\mathsf{B}, \mathsf{A}, \mathsf{U} \in (\mathcal{P}_4^8)^4$, 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 인코딩 평문 $EP = (EP_j)_{j\in[w]}$ $(EP_j \in \mathcal{B}^{\mathbf{b}})$
Output: 암호문 $C = (C_j)_{j\in[w]}$ $(C_j \in \mathcal{B}^{\mathbf{b}})$

procedure $\mathsf{CBC\text{-}Encrypt}^{\mathsf{WB}}(\mathsf{B}, \mathsf{A}, \mathsf{U}; IV, K, EP)$
  $T \leftarrow IV$
  for $j = 0$ to $w-1$ **do**
    $T \leftarrow \mathsf{A}(\mathsf{B}(T) \oplus \mathsf{U}^{-1}(EP_j))$
    $T \leftarrow (\mathsf{B}^{-1} \circ \mathsf{Encrypt}_K \circ \mathsf{A}^{-1})(T)$
    $C_j \leftarrow T$
  end for
  return $C = C_0 \| C_1 \| \cdots \| C_{w-1}$
end procedure

Input: 치환 $\mathsf{B}, \mathsf{V}, \mathsf{W} \in (\mathcal{P}_4^8)^4$, 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 암호문 $C = (C_j)_{j\in[w]}$ $(C_j \in \mathcal{B}^{\mathbf{b}})$
Output: 인코딩 평문 $EP = (EP_j)_{j\in[w]}$ $(EP_j \in \mathcal{B}^{\mathbf{b}})$

procedure $\mathsf{CBC\text{-}Decrypt}^{\mathsf{WB}}(\mathsf{B}, \mathsf{V}, \mathsf{W}; IV, K, C)$
  $C_{-1} \leftarrow IV$
  for $j = 0$ to $w-1$ **do**
    $T \leftarrow (\mathsf{W}^{-1} \circ \mathsf{Decrypt}_K \circ \mathsf{B})(C_j)$
    $EP_j \leftarrow \mathsf{V}^{-1}(\mathsf{W}(T) \oplus \mathsf{B}(C_{j-1}))$
  end for
  return $EP = EP_0 \| EP_1 \| \cdots \| EP_{w-1}$
end procedure



$\mathrm{CBC}-\mathrm{Encrypt}^{\mathrm{WB}}$ $\mathrm{CBC}-\mathrm{Decrypt}^{\mathrm{WB}}$

[그림 4.1: 화이트박스 운영모드: WB-CBC]

암호 연산 $\mathrm{CBC}-\mathrm{Encrypt}^{\mathrm{WB}}$과 복호 연산$\mathrm{CBC}-\mathrm{Decrypt}^{\mathrm{WB}}$중 다음 연산을 테이블 참조 방식으로 구현한다.

$$\mathrm{A}\left(\mathrm{B}(T)\oplus \mathrm{U}^{-1}(EP_j)\right),\quad (\mathrm{B}^{-1}\circ \mathrm{Encrypt}_K\circ \mathrm{A}^{-1})(T),\quad (\mathrm{W}^{-1}\circ \mathrm{Decrypt}_K\circ \mathrm{B})(C_j),\quad \mathrm{V}^{-1}\left(\mathrm{W}(T)\oplus \mathrm{B}(C_{j-1})\right).$$

4비트 단위 치환을 사용하기 때문에 $\mathrm{A}\left(\mathrm{B}(T)\oplus \mathrm{U}^{-1}(EP_j)\right)$와 $\mathrm{V}^{-1}\left(\mathrm{W}(T)\oplus \mathrm{B}(C_{j-1})\right)$를 테이블 참조로 구현하기 위해서는 8비트 입력에 4비트 출력 테이블 32개가 필요하다. 출력 4비트를 1바이트에 저장할 때, 테이블 하나가 256바이트이기 때문에 총 테이블의 크기는 8,192바이트이다.

Memo 4.1. 일반 암호 연산인 $\mathrm{CBC}-\mathrm{Encrypt}$와 화이트박스 암호 연산인 $\mathrm{CBC}-\mathrm{Encrypt}^{\mathrm{WB}}\circ Encode$는 다음 관계를 가진다.

$$\mathrm{CBC}-\mathrm{Encrypt}(\mathrm{B}(IV),\mathrm{K},\mathrm{P})=\mathrm{B}(\mathrm{CBC}-\mathrm{Encrypt}^{\mathrm{WB}}(B,A,U;IV,K,Encode(P))).$$

이때, $\mathrm{B}\circ\mathrm{CBC}-\mathrm{Encrypt}^{\mathrm{WB}}$는 $\mathrm{CBC}-\mathrm{Encrypt}^{\mathrm{WB}}$의 출력 데이터의 모든 블록에 치환 B 연산을 수행함을 의미한다.

제 3절 WB-CTR

CTR 운영모드에서 사용하는 카운터(IV) 갱신 규약은 다음과 같다.

$$IV=IV_0\|IV_1\|\cdots\|IV_{\mathbf{b}-1}\in\mathcal{B}^{\mathbf{b}}\quad\xrightarrow{+1}\quad IV+1=\mathsf{I2B}_{8\mathbf{b}}(\mathsf{B2I}(IV)+1).$$

알고리듬 25는 CTR 운영모드이다.

알고리듬 25 운영모드 CTR: CTR-Encrypt, CTR-Decrypt

Input: 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 평문 P
Output: 암호문 C

procedure CTR-Encrypt(IV, K, P)
  Set $P = (P_j)_{j\in[w]}$ $(\mathsf{ByteLen}(P_j) \begin{cases} = \mathbf{b}, & j \in [0, w-1), \\ \leq \mathbf{b}, & j = w-1. \end{cases})$
  for $j = 0$ to $w-2$ **do**
    $T \leftarrow \mathsf{Encrypt}_K(IV + j)$
    $C_j \leftarrow T \oplus P_j$
  end for
  $T \leftarrow \mathsf{Encrypt}_K(IV + w - 1)$
  $C_{w-1} \leftarrow P_{w-1} \oplus$ (First $\mathsf{ByteLen}(P_{w-1})$-byte of T)
  return $C = C_0 \| C_1 \| \cdots \| C_{w-1}$
end procedure

Input: 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 암호문 C
Output: 평문 P

procedure CTR-Encrypt(IV, K, C)
  Set $C = (C_j)_{j\in[w]}$ $(\mathsf{ByteLen}(C_j) \begin{cases} = \mathbf{b}, & j \in [0, w-1), \\ \leq \mathbf{b}, & j = w-1. \end{cases})$
  for $j = 0$ to $w-2$ **do**
    $T \leftarrow \mathsf{Encrypt}_K(IV + j)$
    $P_j \leftarrow T \oplus C_j$
  end for
  $T \leftarrow \mathsf{Encrypt}_K(IV + w - 1)$
  $P_{w-1} \leftarrow C_{w-1} \oplus$ (First $\mathsf{ByteLen}(C_{w-1})$-byte of T)
  return $P = P_0 \| P_1 \| \cdots \| P_{w-1}$
end procedure

알고리듬 26은 Whitebox WF-LEA기반WB-CTR 운영모드이고, 그림 4.2는 이를 도식화한 것이다.

알고리듬 26 화이트박스 운영모드 WB-CTR: CTR-Encrypt$^{\mathsf{WB}}$ ∘ Encode, Decode ∘ CTR-Decrypt$^{\mathsf{WB}}$

Input: 치환 $\mathsf{V} \in (\mathcal{P}_4^8)^4$, 평문 P
Output: 인코딩 평문 $EP = (EP_j)_{j\in[w]}$ $(EP_j \in \mathcal{B}^{\mathbf{b}})$

procedure Encode(V, P)
  $EP = (EP_j)_{j\in[w]} \leftarrow \mathsf{ExpMsg}(P)$
  $EP_j \leftarrow \mathsf{V}(EP_j)$ for $j \in [w]$
  return EP
end procedure

Input: 치환 $\mathsf{U} \in (\mathcal{P}_4^8)^4$, 인코딩 평문 $EP = (EP_j)_{j\in[w]}$ $(EP_j \in \mathcal{B}^{\mathbf{b}})$
Output: 평문 P

procedure Decode(U, P)
  $EP_j \leftarrow \mathsf{U}(EP_j)$ for $j \in [w]$
  $P \leftarrow$ Remove the padding of EP
  return P
end procedure

Input: 치환 $\mathsf{A}, \mathsf{B}, \mathsf{V}, \mathsf{W} \in (\mathcal{P}_4^8)^4$, 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 인코딩 평문 $EP = (EP_j)_{j\in[w]}$ $(EP_j \in \mathcal{B}^{\mathbf{b}})$
Output: 암호문 $C = (C_j)_{j\in[w]}$ $(C_j \in \mathcal{B}^{\mathbf{b}})$

procedure CTR-Encrypt$^{\mathsf{WB}}$$(\mathsf{A}, \mathsf{B}, \mathsf{V}, \mathsf{W}; IV, K, EP)$
  for $j = 0$ to $w-1$ **do**
    $T \leftarrow IV + j$
    $T \leftarrow (\mathsf{B}^{-1} \circ \mathsf{Encrypt}_K \circ \mathsf{A})(T)$
    $C_j \leftarrow \mathsf{W}(\mathsf{A}(\mathsf{B}(T) \oplus \mathsf{V}^{-1}(EP_j)))$
  end for
  return $C = C_0 \| C_1 \| \cdots \| C_{w-1}$
end procedure

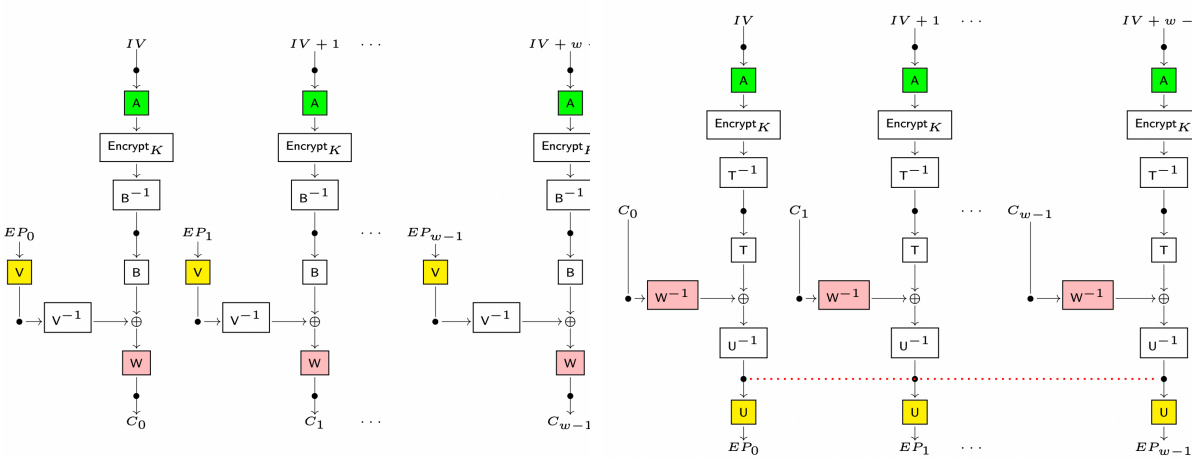
Input: 치환 $\mathsf{A}, \mathsf{T}, \mathsf{W}, \mathsf{U} \in (\mathcal{P}_4^8)^4$, 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 암호문 $C = (C_j)_{j\in[w]}$ $(C_j \in \mathcal{B}^{\mathbf{b}})$
Output: 인코딩 평문 $EP = (EP_j)_{j\in[w]}$ $(EP_j \in \mathcal{B}^{\mathbf{b}})$

procedure CTR-Decrypt$^{\mathsf{WB}}$$(\mathsf{A}, \mathsf{T}, \mathsf{W}, \mathsf{U}; IV, K, C)$
  for $j = 0$ to $w-1$ **do**
    $T \leftarrow IV + j$
    $T \leftarrow (\mathsf{T}^{-1} \circ \mathsf{Encrypt}_K \circ \mathsf{A})(T)$
    $EP_j \leftarrow \mathsf{U}^{-1}(\mathsf{T}(T) \oplus \mathsf{W}^{-1}(C_j))$
  end for
  return $EP = EP_0 \| EP_1 \| \cdots \| EP_{w-1}$
end procedure

암호 연산 $CTR - \mathrm{Encrypt}^{\mathsf{WB}}$과 복호 연산 $CTR - \mathrm{Decrypt}^{\mathsf{WB}}$ 중 다음 연산을 테이블 참조 방식으로 구현한다.

$$(\mathrm{B}^{-1} \circ \mathrm{Encrypt}_K \circ \mathrm{A})(T), \quad \mathrm{W}\left(\mathrm{A}\left(\mathrm{B}(T) \oplus \mathrm{V}^{-1}(EP_j)\right)\right), \quad (\mathrm{T}^{-1} \circ \mathrm{Encrypt}_K \circ \mathrm{A})(T), \quad \mathrm{U}^{-1}\left(\mathrm{T}(T) \oplus \mathrm{W}^{-1}(C_j)\right).$$

4비트 단위 치환을 사용하기 때문에 $\mathrm{W}\left(\mathrm{A}\left(\mathrm{B}(T) \oplus \mathrm{V}^{-1}(EP_j)\right)\right)$와 $\mathrm{U}^{-1}\left(\mathrm{T}(T) \oplus \mathrm{W}^{-1}(C_j)\right)$를 테이블 참조로 구현하기 위해서는 8비트 입력에 4비트 출력 테이블 32개가 필요하다. 출력 4비트를 1바이트에 저장할 때, 테이블 하나가 256바이트이기 때문에 총 테이블의 크기는 8,192바이트이다.



$\mathrm{CTR} - \mathrm{Encrypt}^{\mathrm{WB}}$ $\mathrm{CTR} - \mathrm{Decrypt}^{\mathrm{WB}}$

[그림 4.2: 화이트박스 운영모드: WB-CTR]

제 5장 Whitebox WF-LEA 기반 메시지 인증모드 VP-MAC

본 장에서는 Whitebox WF-LEA 기반 메시지 인증모드 VP-MAC에 대해 설명한다.

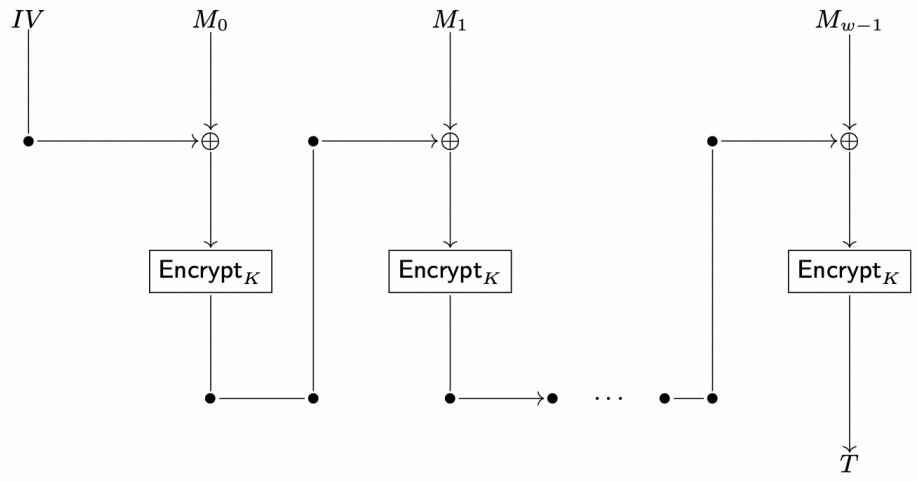
제 1절 CBC-MAC

알고리듬 27은 CBC-MAC의 인증값 생성 과정을 보여준다.

알고리듬 27 CBC-MAC: 인증값 생성 CBC-MAC-GenTag

Input: 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, w블록 메시지 $M = (M_j)_{j \in [w]}$ $(M_j \in \mathcal{B}^{\mathbf{b}})$.
Output: 인증값 $T \in \mathcal{B}^{\mathbf{b}}$

1: **procedure** CBC-MAC-GenTag(IV, K, M)
2: $T \leftarrow IV$
3: **for** $j = 0$ to $w - 1$ **do**
4: $T \leftarrow T \oplus M_j$
5: $T \leftarrow \mathsf{Encrypt}_K(T)$
6: **end for**
7: **return** T
8: **end procedure**



[그림 5.1: CBC-MAC: CBC-MAC-GenTag(Π;IV,K,M)]

제 2절 메시지 확장

다음과 같이 가변길이 메시지를 One-Zeros 패딩을 적용하여 N배수의 블록으로 확장한다.

n바이트 메시지 $M \rightarrow$ bN (1 + ⌊n/bN⌋)바이트 메시지 $EM = M$‖0x800...000

이때 N은 4이상의 값으로 설정한다. 알고리듬 28 은 VP-MAC의 메시지 확장 과정을 보여준다.

알고리듬 28 VP-MAC: 메시지 확장 VP-MAC-ExpMsg

Input: 기준 블록 개수 $N(\geq 4)$, 메시지 M
Output: 확장 메시지 $EM = EM_0 \| EM_1 \| \cdots \| EM_{w-1}$ $(EM_j \in \mathcal{B}^{\mathbf{b}})$

1: **procedure** VP-MAC-ExpMsg(N; M)
2: $w \leftarrow N\left(1 + \left\lfloor \frac{\mathsf{ByteLen}(M)}{\mathbf{b}N} \right\rfloor\right)$ ▷ w: EM의 **b**바이트 블록 개수
3: $EM \leftarrow M \| \mathsf{0x80} \| (\mathsf{0x00})^{\mathbf{b}w - (\mathsf{ByteLen}(M)+1)}$ ▷ One-Zeros 패딩
4: Set $EM = EM_0 \| \cdots \| EM_{w-1}$ where $EM_j \in \mathcal{B}^{\mathbf{b}}$
5: **return** EM
6: **end procedure**

Example 5.1. $\mathrm{b} = 16, N = 5$일 때,

메시지 바이트 크기	확장 메시지 바이트 크기	$N = 5$
$0 \sim 16N - 1$	$16N$	5블록
$16N \sim 32N - 1$	$32N$	10블록
$32N \sim 48N - 1$	$48N$	15블록
$\vdots$	$\vdots$	$\vdots$
$16kN \sim 16(k+1)N - 1$	$16(k+1)N$	$5(k+1)$블록

제 3절 인증값 생성 및 검증

$\mathrm{Encrypt}_K$를 다음과 같이 첫 **t**라운드 연산, 가운데 $\mathbf{r} - 2\mathbf{t}$개 라운드 연산, 마지막 **t**개 라운드 연산으로 구분할 수 있다.

$$\mathrm{Encrypt}_K = \underbrace{\mathrm{Encrypt}_K^{(2)}}_{\text{마지막 } \mathbf{t} \text{개 라운드 연산}} \circ \underbrace{\mathrm{Encrypt}_K^{(1)}}_{\text{가운데 } \mathbf{r}-2\mathbf{t} \text{개 라운드 연산}} \circ \underbrace{\mathrm{Encrypt}_K^{(0)}}_{\text{첫 } \mathbf{t} \text{개 라운드 연산}}.$$

이때 **t**는 다음 범위의 값으로 설정한다.

$$\mathbf{t} \in \{2,3,\ldots,10\}.$$

4비트 단위 **b**바이트 치환 A,B,U,V으로 다음과 같이 $\mathrm{Encrypt}_K$와 동치인 함수를 만들 수 있다.

$$\mathrm{Encrypt}_K = \mathrm{B} \circ \mathrm{B}^{-1} \circ \mathrm{Encrypt}_K^{(2)} \circ \mathrm{V} \circ \mathrm{V}^{-1} \circ \mathrm{Encrypt}_K^{(1)} \circ \mathrm{U}^{-1} \circ \mathrm{U} \circ \mathrm{Encrypt}_K^{(0)} \circ \mathrm{A}^{-1} \circ \mathrm{A}.$$

다음과 같이 $\mathrm{Encrypt}_K^{\mathrm{WB}}$, $\mathrm{Encrypt}_K^{\mathrm{WB}(2)}$, $\mathrm{Encrypt}_K^{\mathrm{WB}(1)}$, $\mathrm{Encrypt}_K^{\mathrm{WB}(0)}$를 정의하자.

$$\begin{aligned}
\mathrm{Encrypt}_K^{\mathrm{WB}} &:= \mathrm{B}^{-1} \circ \mathrm{Encrypt}_K \circ \mathrm{A}^{-1}, \\
\mathrm{Encrypt}_K^{\mathrm{WB}(2)} &:= \mathrm{B}^{-1} \circ \mathrm{Encrypt}_K^{(2)} \circ \mathrm{V}, \\
\mathrm{Encrypt}_K^{\mathrm{WB}(1)} &:= \mathrm{V}^{-1} \circ \mathrm{Encrypt}_K^{(1)} \circ \mathrm{U}^{-1}, \\
\mathrm{Encrypt}_K^{\mathrm{WB}(0)} &:= \mathrm{U} \circ \mathrm{Encrypt}_K^{(0)} \circ \mathrm{A}^{-1},
\end{aligned}$$

이 함수들은 다음을 만족한다.

$$\begin{aligned}\mathrm{Encrypt}_K^{\mathrm{WB}} &:= \mathrm{B}^{-1} \circ \mathrm{Encrypt}_K \circ \mathrm{A}^{-1} \\ &= \mathrm{B}^{-1} \circ \mathrm{Encrypt}_K^{(2)} \circ \mathrm{V} \circ \mathrm{V}^{-1} \circ \mathrm{Encrypt}_K^{(1)} \circ \mathrm{U}^{-1} \circ \mathrm{U} \circ \mathrm{Encrypt}_K^{(0)} \circ \mathrm{A}^{-1} \\ &= \mathrm{Encrypt}_K^{\mathrm{WB}(2)} \circ \mathrm{Encrypt}_K^{\mathrm{WB}(1)} \circ \mathrm{Encrypt}_K^{\mathrm{WB}(0)}.\end{aligned}$$

다음 라운드 색인집합을 정의한다.

$$\begin{aligned}I_0 &:= \{0,1,\dots,\mathbf{t}-1\} = [\mathbf{t}], \\ I_1 &:= \{\mathbf{t},\dots,\mathbf{r}-\mathbf{t}-1\} = [\mathbf{r}-\mathbf{t}]\backslash I_0, \\ I_2 &:= \{\mathbf{r}-\mathbf{t},\dots,\mathbf{r}-1\} = [\mathbf{r}]\backslash I_1.\end{aligned}$$

알고리듬 29는 VP-MAC의 인증값 생성 과정을 보여준다. 그림 5.2는 이 과정을 도식화한 것이다.

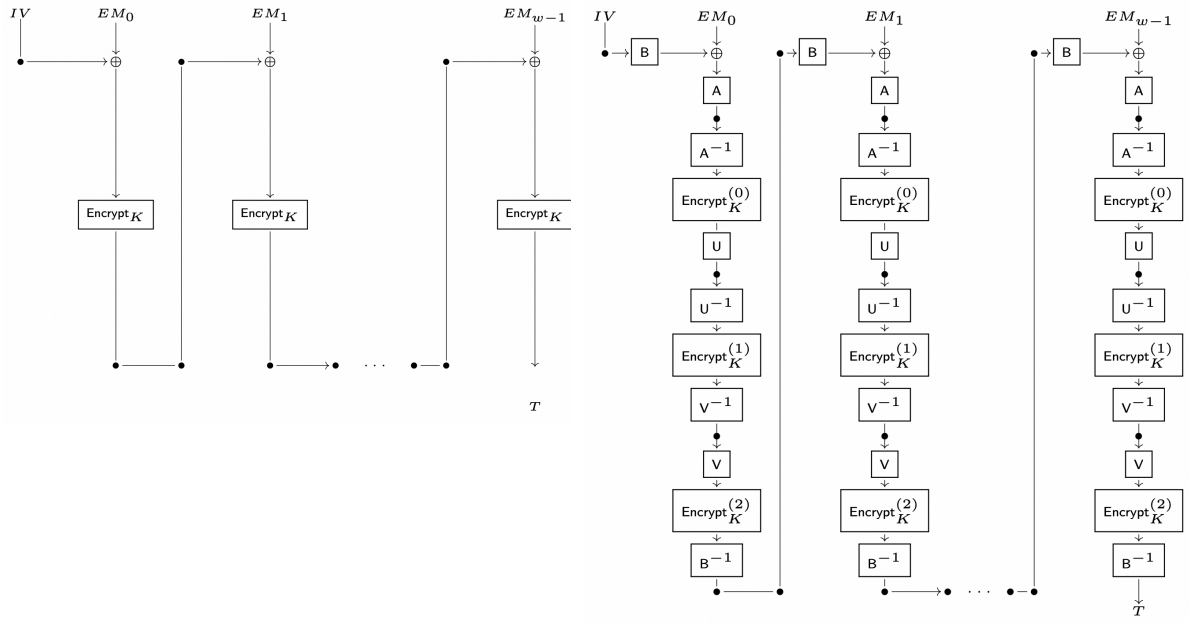
알고리듬 29 VP-MAC: 인증값 생성 VP-MAC-GenTag, VP-MAC-GenTag$^{\mathrm{WB}}$

Input: 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 메시지 M
Output: 인증값 $T \in \mathcal{B}^{\mathbf{b}}$

procedure VP-MAC-GenTag(IV, K, M)
  $EM \leftarrow$ VP-MAC-ExpMsg$(N; M)$
  $T \leftarrow$ CBC-MAC-GenTag$(\Pi; IV, K, EM)$
  return T
end procedure

Input: 4비트 단위 **b**바이트 치환 $\mathsf{A}, \mathsf{B}, \mathsf{U}, \mathsf{V} \in (\mathcal{P}_4^8)^4$, 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 메시지 M
Output: 인증값 $T \in \mathcal{B}^{\mathbf{b}}$

procedure VP-MAC-GenTag$^{\mathrm{WB}}(\mathsf{A}, \mathsf{B}, \mathsf{U}, \mathsf{V}, IV, K, M)$
  $EM = (EM_j)_{j\in[w]} \leftarrow$ VP-MAC-ExpMsg$(N; M)$
  $T \leftarrow IV$
  for $j = 0$ to $w-1$ **do**
    $T \leftarrow \mathsf{A}(\mathsf{B}(T) \oplus EM_j)$
    $T \leftarrow (\mathsf{U} \circ \mathrm{Encrypt}_K^{(0)} \circ \mathsf{A}^{-1})(T)$
    $T \leftarrow (\mathsf{V}^{-1} \circ \mathrm{Encrypt}_K^{(1)} \circ \mathsf{U}^{-1})(T)$
    $T \leftarrow (\mathsf{B}^{-1} \circ \mathrm{Encrypt}_K^{(2)} \circ \mathsf{V})(T)$
  end for
  return T
end procedure



VP − MAC − GenTag　　VP − MAC − GenTag^{WB}

[그림 5.2: VP-MAC: 인증값 생성]

VP − MAC − GenTag는 일반 연산이며, VP − MAC − GenTag^{WB}은 화이트박스 연산으로 다음 관계를 가진다.

VP-MAC-GenTag(B(IV), K, M) = B(VP-MAC-GenTag^{WB}(IV, K, M)).

알고리듬 30은 VP-MAC의 인증값 검증 과정을 보여준다.

알고리듬 30 VP-MAC: 인증값 검증 VP-MAC-VerifyTag

Input: 4비트 단위 b바이트 치환 B $\in (\mathcal{P}_4^8)^4$, 초깃값 $IV \in \mathcal{B}^{\mathbf{b}}$, 키 $K \in \mathcal{B}^{\mathbf{k}}$, 메시지 M, 인증값 $T \in \mathcal{B}^{\mathbf{b}}$
Output: `Valid` 또는 `Invalid`

1: **procedure** VP-MAC-VerifyTag(B, IV, K, M, T)
2: $T' \leftarrow$ VP-MAC-GenTag(B(IV), K, M)
3: **if** B(T) = T' **then**
4: **return** `Valid`
5: **else**
6: **return** `Invalid`
7: **end if**
8: **end procedure**

인증값 검증 과정에서 치환 B를 입력해야 한다. 테이블 형태의 B는 512바이트이다. 만일 이 테이블의 저장이 부담스러운 상황에서는 알고리듬 31과 같이 B를 생성하는 32비트 시드를 입력하는 것으로 대체할 수 있다.

알고리듬 31 VP-MAC: 인증값 검증 VP-MAC-VerifyTag (시드 입력)

Input: 32바이트 시드 $S \in \mathcal{B}^{32}$, 초깃값 $IV \in \mathcal{B}^{\mathsf{b}}$, 키 $K \in \mathcal{B}^{\mathsf{k}}$, 메시지 M, 인증값 $T \in \mathcal{B}^{\mathsf{b}}$
Output: `Valid` 또는 `Invalid`

procedure VP-MAC-VerifyTag(S, IV, K, M, T)
  $R \leftarrow$ GenRand512Bytes(S)
  B $\leftarrow$ GenRand4BitPerm(R)
  $T' \leftarrow$ VP-MAC-GenTag$(\mathrm{B}(IV), K, M)$
  if $\mathrm{B}(T) = T'$ **then**
    return `Valid`
  else
    return `Invalid`
  end if
end procedure

제 4절 $\mathrm{VP-MAC-GenTag}^{\mathrm{WB}}$의 구현

$\mathrm{VP-MAC-GenTag}^{\mathrm{WB}}$에서 다음 4개 연산은 테이블 참조 방식으로 처리한다.

$$\mathrm{A}\big(\mathrm{B}(T) \oplus EM_j\big), \quad \big(\mathrm{U} \circ \mathrm{Encrypt}_K^{(0)} \circ \mathrm{A}^{-1}\big)(T), \quad \big(\mathrm{V}^{-1} \circ \mathrm{Encrypt}_K^{(1)} \circ \mathrm{U}^{-1}\big)(T), \quad \big(\mathrm{B}^{-1} \circ \mathrm{Encrypt}_K^{(2)} \circ \mathrm{V}\big)(T).$$

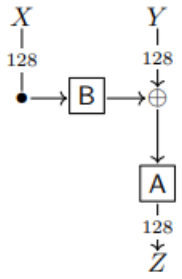
4.1 $\mathrm{A}(\mathrm{B}(X) \oplus Y)$

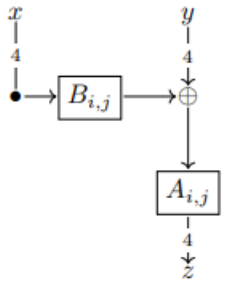
4비트 단위 16바이트 치환 A,B를 다음과 같이 정의할 때,

$$\begin{aligned} \mathrm{A} &:= \mathrm{diag}(A_0, A_1, A_2, A_3) \in (\mathcal{P}_4^8)^4, \quad \text{where } A_i := \mathrm{diag}\big(A_{i,7}, \dots, A_{i,0}\big) \in \mathcal{P}_4^8, \\ \mathrm{B} &:= \mathrm{diag}(B_0, B_1, B_2, B_3) \in (\mathcal{P}_4^8)^4, \quad \text{where } B_i := \mathrm{diag}\big(B_{i,7}, \dots, B_{i,0}\big) \in \mathcal{P}_4^8, \end{aligned}$$

모든 $i \in [4], j \in [8], x, y \in \mathcal{N}$에 대해 $A_{i,j}\big(B_{i,j}(x) \oplus y\big)$를 사전계산한 후 이를 테이블 참조하여 $\mathrm{A}(\mathrm{B}(X) \oplus Y)$를 계산한다. 즉,

$$z = A_{i,j}\big(B_{i,j}(x) \oplus y\big) \quad \Rightarrow \quad z = \mathrm{XT}_{i,j}(x, y).$$





알고리듬 32는 테이블 $\big(\mathrm{XT}_{i,j}\big)_{i \in [4], j \in [8]}$를 이용하여 $\mathrm{A}(\mathrm{B}(X) \oplus Y)$를 계산하는 과정을 보여준다.

알고리듬 32 VP-MAC: $\mathsf{A}(\mathsf{B}(X) \oplus Y)$ with A and B

Input: $X = (X_{i,j})_{i\in[4],j\in[8]}$, $Y = (Y_{i,j})_{i\in[4],j\in[8]} \in \mathcal{N}^{32}$
Output: 128비트 $Z = \mathsf{A}(\mathsf{B}(X) \oplus Y) = (Z_{i,j})_{i\in[4],j\in[8]} \in \mathcal{N}^{32}$

procedure $\mathsf{XT}(X, Y)$
  for $i = 0$ to 3 **do**
    for $j = 0$ to 7 **do**
      $Z_{i,j} \leftarrow \mathsf{XT}_{i,j}(X_{i,j}, Y_{i,j})$
    end for
  end for
  return Z
end procedure

4.2 화이트박스키 WK_s, WK_e 생성

$\text{Encrypt}_K^{\text{WB}}$ 연산에 필요한 화이트박스 키테이블 WK_s, WK_e를 다음의 과정으로 생성한다.

(단계1) 치환 U,V $\in (\mathcal{P}_4^8)^4$ 임의 생성

(단계2) 화이트박스키 WK_1 생성 (가운데 $\mathbf{r-2t}$ 라운드 연산에 사용하는 화이트박스 키테이블)

$$WK_1 = \left(\text{ETA}_{i,j}^{(l)}, \text{ETR}_{i,j}^{(l)}\right)_{i\in[3],j\in[8],l\in I_1} \leftarrow \text{GenWKE}(Seed; \text{U,V}, K, I_1).$$

(단계3) 치환 A $\in (\mathcal{P}_4^8)^4$ 임의 생성

(단계4) 화이트박스키 WK_0 생성 (첫 $\mathbf{t}$ 라운드 연산에 사용하는 화이트박스 키테이블)

$$WK_0 = \left(\text{ETA}_{i,j}^{(l)}, \text{ETR}_{i,j}^{(l)}\right)_{i\in[3],j\in[8],l\in I_0} \leftarrow \text{GenWKE}\left(Seed; \text{A}, \text{U}^{-1}, K, I_0\right).$$

(단계5) 치환 B $\in (\mathcal{P}_4^8)^4$ 임의 생성

(단계6) 화이트박스키 WK_2 생성 (마지막 $\mathbf{t}$ 라운드 연산에 사용하는 화이트박스 키테이블)

$$WK_2 = \left(\text{ETA}_{i,j}^{(l)}, \text{ETR}_{i,j}^{(l)}\right)_{i\in[3],j\in[8],l\in I_2} \leftarrow \text{GenWKE}\left(Seed; \text{V}^{-1}, \text{B}, K, I_2\right).$$

(단계7) 화이트박스키 XT 생성 (XOR 연산에 사용하는 화이트박스 키테이블)

$$\text{XT} = \left(\text{XT}_{i,j}\right)_{i\in[4],j\in[8]}, \leftarrow \text{GenXT(A,B)}.$$

(단계8) $WK_e \leftarrow < WK_0, WK_2, \text{XT} >, WK_s \leftarrow < WK_1 >$

4.3 VP-MAC-GenTag^{WB}

알고리듬 33은VP – MAC의 메시지 인증값 계산을 화이트박스 연산으로 처리하는 과정을 보여준다.

알고리듬 33 VP-MAC: 인증값 생성 VP-MAC-GenTag^{WB}

Input: 화이트박스 키테이블 $WK_s =< WK_1 >, WK_e =< WK_0, WK_2, \mathsf{XT} >$, 메시지 M
Output: 인증값 $T \in \mathcal{B}^{\mathbf{b}}$

procedure VP-MAC-GenTag$^{WB}(WK_s, WK_e; IV, M)$
  $EM = (EM_j)_{j\in[w]} \leftarrow$ VP-MAC-ExpMsg$(N; M)$
  $T \leftarrow IV$
  for $j = 0$ to $w - 1$ **do**
    $T \leftarrow \mathsf{XT}(T, EM_j)$ ▷ $T \leftarrow \mathsf{A}(\mathsf{B}(T) \oplus EM_j)$
    $T \leftarrow \mathsf{Encrypt}_K^{WB(0)}(WK_0; T)$ ▷ $T \leftarrow (\mathsf{U} \circ \mathsf{Encrypt}_K^{(0)} \circ \mathsf{A}^{-1})(T)$
    $T \leftarrow \mathsf{Encrypt}_K^{WB(1)}(WK_1; T)$ ▷ $T \leftarrow (\mathsf{V}^{-1} \circ \mathsf{Encrypt}_K^{(1)} \circ \mathsf{U}^{-1})(T)$
    $T \leftarrow \mathsf{Encrypt}_K^{WB(2)}(WK_2; T)$ ▷ $T \leftarrow (\mathsf{B}^{-1} \circ \mathsf{Encrypt}_K^{(2)} \circ \mathsf{V})(T)$
  end for
  return T
end procedure

제 6 장 VP-MAC 라이브러리

제 1 절 개발환경

개발환경은 다음과 같다.

- Device: MacBook Pro (13-inch, M1, 8GB RAM, 2020), macOS Monterey v12.4
- Programming Language: C
- Compiler: GCC -O2

제2절 라이브러리 구성

표 6.1은 Whitebox WF – LEA 라이브러리 구성을 보여준다

[표 6.1: 라이브러리 구성 소스코드]

코드명	설명
wflea.*	WF-LEA 암/복호 연산
wbwflea_config.h	Whitebox WF-LEA 키길이 설정 (16 or 24 or 32바이트)
wbwflea_encodings.*	Whitebox WF-LEA용 네트워크 랜덤인코딩 생성
wbwflea_ext_transform.*	Whitebox WF-LEA용 외부인코딩 생성 및 변환연산
wbwflea_tables.*	Whitebox WF-LEA용 암/복호 테이블 생성
wbwflea_encrypt.*	Whitebox WF-LEA용 암/복호 연산
randperm.*	4비트 랜덤치환 생성
drbg.*	NIST DRBG를 이용한 난수생성 (KISA 오픈소스)
seedcbc.*	NIST DRBG 연산용 블록암호 SEED 연산 (KISA 오픈소스)
ariacbc.*	NIST DRBG 연산용 블록암호 ARIA 연산 (KISA 오픈소스)
sha256.*	SHA256 연산 (Brad Conte 오픈소스)
errors.h	오류 메시지 정의
main_test.c	예제코드 모음

제3절 사용법

컴파일

본 라이브러리는 다음과 같이 GCC 컴파일한다.

```
$ gcc main_test.c sha256.c wflea.c randperm.c \
        wbwflea_encodings.c wbwflea_ext_transform.c \
```

```
wbwflea_tables.c wbwflea_encrypt.c \
drbg.c seedcbc.c ariacbc.c -O2 -Wall
```

3.2 난수발생기 설정

난수발생기는 KISA에서 배포한 NIST DRBG 코드 drbg.h/drbg.c에 정의되어 있다. 블록암호 ARIA128 기반 DRBG로 기본 설정되어 있으며, 다음의 함수 drbg_setup에서 알고리듬 수정이 가능하다.

Code 6.1: drbg.c

```c
void drbg_setup(unsigned char* entropy)
{
  //unsigned char entropy[32] = {0,};
  unsigned char nonce[16] = {0,};
  char per_str[] = "Whitebox WFLEA 2022";

  /* WBWFLEA_STATE defined in drbg.h (KISA_CTR_DRBG_STATE WBWFLEA_STATE;) */
  KISA_CTR_DRBG_Instantiate(
    &WBWFLEA_STATE,
    ALGO_ARIA128,
    entropy, 32,
    nonce, 16,
    (unsigned char*)per_str, strlen(per_str),
    USE_DERIVATION_FUNCTION);
}
```

함수 drbg_setup는 32바이트 seed를 입력받아, 난수발생기 상태값 WBWFLEA_STATE이 갱신한다. 함수 drbg_get_randbyte는 drbg_setup을 통해 설정한 WBWFLEA_STATE을 이용하여 8비트 난수를 생성한다.

Code 6.2: drbg.c

```c
unsigned char drbg_get_randbyte()
{
  unsigned char ret;
  KISA_CTR_DRBG_Generate(&WBWFLEA_STATE, &ret, 8, NULL, 0);
  return ret;
}
```

3.3 키길이 설정

wbwflea_config.h 파일에서 WBWFLEA_KEY_BYTES의 값을 원하는 키로 설정한다.

Code 6.3: wbwflea config.h

```c
/*!< choose the key size: 16 or 24 or 32 */
#define WBWFLEA_KEY_BYTES   32
```

3.4 LEA로 설정하기

wbwflea_config.h 파일에서 아닌 설정할 수 있다.

Code 6.4: wbwflea config.h

```
/* -- if you want the LEA, define "LEA" below -- */
#if 0
#define _WFLEA_
#else
#define _LEA_
#endif
```

3.5 WF-LEA 암/복호 연산

WFLEA는 wflea.*에 정의되어 있다.

동작 순서

다음과 같이 라운드키 생성 후 암/복호 연산을 수행하도록 구현되어 있다.

단계 1		단계 2
WF-LEA context 생성 (라운드키 생성)	⟶	암/복호 함수 호출
wflea_gen_ctx		wflea_encryptblk
		wflea_decryptblk

Code 6.5: wflea.h

```c
typedef struct {

  /* fixed parameters */
  int blk_bytes;     /*!< number of bytes of a block = 16 */
  int blk_words;     /*!< number of words of a block = 4 */
  int rndkey_bytes; /*!< number of bytes of a roundkey = 24 */
  int rndkey_words; /*!< number of words of a roundkey = 6 */

  /* key */
  int key_bytes;          /*!< number of bytes of a key = 16 or 24 or 32*/
  byte key[WFLEA_MAX_KEY_BYTES];  /*!< key */

  /* number of rounds */
  int num_rounds;         /*!< number of rounds = 24 or 28 or 32*/

  /* round keys */
  word rndkey[WFLEA_MAX_ROUNDS][6]; /*!< Nr roundkeys */

} WFLEA_CTX;
```

함수 입/출력

함수명	입력	출력
wflea_gen_ctx	key, key length	WF-LEA context
wflea_encryptblk	WF-LEA context, 128-bit data	128-bit encrypted data
wflea_decryptblk	WF-LEA context, 128-bit data	128-bit decrypted data

- 함수 wflea_gen_ctx키와 키길이를 입력받아 구조체 WF-LEA context를 생성한다. 유효한 키길이 입력 시 NO_ERROR반환하고, 아닐 시 ERR_INPUT반환한다.

- 함수 wflea_encryptblkwflea_decryptblkWF-LEA context와 16바이트 데이터를 입력받아, 암/복호 연산 수행 후 16바이트 결과 데이터를 출력한다.

예제코드

Code 6.6: main test.c

```
/* setup: key */
byte key[32] = {0x0,};
ge{\Nr}and_bytes(key, 32);

/* setup: WFLEA context (roundkey generation) */
WFLEA_CTX wflea_ctx;
wflea_gen_ctx(&wflea_ctx, WBWFLEA_KEY_BYTES*8, key);
wflea_show_ctx(&wflea_ctx);

/* test: WFLEA round trip */
byte dat1[16] = {0x0,};
byte dat2[16] = {0x0,};
byte dat3[16] = {0x0,};

/* plaintext generation */
ge{\Nr}and_bytes(dat1, 16);

printf("P"); show_bytes(dat1, 16);

wflea_encryptblk(dat2, dat1, &wflea_ctx); /* WFLEA encryption */
printf("E(P) "); show_bytes(dat2, 16);

wflea_decryptblk(dat3, dat2, &wflea_ctx); /* WFLEA decryption */
printf("D(E(P))  "); show_bytes(dat3, 16);
```

3.6 Whitebox WF-LEA

외부인코딩 생성 및 저장

`wbwflea_ext_transform.*`에 정의되어 있다.

외부인코딩 구조체

외부인코딩은 4비트 치환 32개로 구성한 $f \in (\mathcal{P}_4^8)^4$와 그 역변환 $f^{-1} \in (\mathcal{P}_4^8)^4$을 구조체 `WBWFLEA_EXT_ENCODING`로 정의한다.

Code 6.7: 외부인코딩 구조체

```
typedef struct {
/*
f: input(or output) encoding
f = (f[0], f[1], f[2], f[3])
f[i] = diag(f[i][7],...,f[i][0])i = 0,1,2,3
f[i][j]: 4-bit permutation  j = 0,1,2,3,4,5,6,7
*/
byte f[4][8][16];
byte f_inv[4][8][16];
} WBWFLEA_EXT_ENCODING;
```

동작 순서

다음과 같이 외부인코딩 A,B을 각각 생성하여 서로 다른 바이너리로 저장한다.

단계 1		단계 2		단계 3
DRBG 설정 (32바이트 시드 입력) drbg_setup	⟶	외부인코딩 A, B 생성 wbwflea_gen_ext_encoding	⟶	외부인코딩 A, B 저장 (바이너리 파일) wbwflea_write_ext_encoding

함수 입/출력

함수명	입력	출력
wbwflea_gen_ext_encoding		WBWFLEA_EXT_ENCODING
wbwflea_write_ext_encoding	WBWFLEA_EXT_ENCODING	바이너리 파일

- 함수 wbwflea_gen_ext_encoding은 Fisher-Yates Shuffle 알고리듬을 이용하여 외부인코딩 A,B를 생성하고, 이를 구조체 WBWFLEA_EXT_ENCODING에 저장한다. 이때 함수 drbg_get_randbyte로 생성한 8비트 난수를 Fisher-Yates Shuffle 알고리듬에 사용한다.

- 함수 wbwflea_write_ext_encoding외부인코딩 값이 저장되어 있는 구조체를 바이너리 파일로 저장한다.

예제코드

Code 6.8: 암/복호 연산용 외부인코딩 A, B 생성

```
/* drbg setup */
ge{\Nr}and_bytes_using_randf(seed, 32);drbg_setup(seed);
/* external encoding A generation (for encryption) */
WBWFLEA_EXT_ENCODING Ae;
wbwflea_gen_ext_encoding(&Ae);

/* drbg setup */
ge{\Nr}and_bytes_using_randf(seed, 32);drbg_setup(seed);
/* external encoding B generation (for encryption) */
WBWFLEA_EXT_ENCODING Be;
wbwflea_gen_ext_encoding(&Be);

wbwflea_write_ext_encoding(&Ae, "encA.bin");
wbwflea_write_ext_encoding(&Be, "encB.bin");
```

```
15
16 /* drbg setup */
17 ge{\Nr}and_bytes_using_randf(seed, 32);drbg_setup(seed);
18 /* external encoding A generation (for decryption) */
19 WBWFLEA_EXT_ENCODING Ad;
20 wbwflea_gen_ext_encoding(&Ad);
21 wbwflea_write_ext_encoding(&Ad, "decA.bin");
22
23 /* drbg setup */
24 ge{\Nr}and_bytes_using_randf(seed, 32);drbg_setup(seed);
25 /* external encoding B generation (for decryption) */
26 WBWFLEA_EXT_ENCODING Bd;
27 wbwflea_gen_ext_encoding(&Bd);
28 wbwflea_write_ext_encoding(&Bd, "decB.bin");
```

3.6.2 네트워크 랜덤인코딩 생성

wbwflea_encodings.*에 정의되어 있다.

네트워크 랜덤인코딩 구조체

암호 연산용 네트워크 랜덤인코딩 구조체 WBWFLEA_ENCODINGS_FOR_ENCRYPTION와 복호 연산용 네트워크 랜덤인코딩 구조체 WBWFLEA_ENCODINGS_FOR_DECRYPTION는 다음과 같다.

Code 6.9: 네트워크 랜덤인코딩 구조체

```
1  typedef struct {
2  byte f[WBWFLEA_ROUNDS][4][8][16];
3  byte f_inv[WBWFLEA_ROUNDS][4][8][16];
4  byte g[WBWFLEA_ROUNDS][3][8][16];
5  byte g_inv[WBWFLEA_ROUNDS][3][8][16];
6  byte h[WBWFLEA_ROUNDS][3][8][16];
7  byte h_inv[WBWFLEA_ROUNDS][3][8][16];
8  byte t[WBWFLEA_ROUNDS][3];
9  } WBWFLEA_ENCODINGS_FOR_ENCRYPTION;
10
11 typedef struct {
12 byte f[WBWFLEA_ROUNDS][3][8][16];
13 byte f_inv[WBWFLEA_ROUNDS][3][8][16];
14 byte g[WBWFLEA_ROUNDS][3][8][16];
15 byte g_inv[WBWFLEA_ROUNDS][3][8][16];
16 byte h[WBWFLEA_ROUNDS][4][8][16];
17 byte h_inv[WBWFLEA_ROUNDS][4][8][16];
18 byte t[WBWFLEA_ROUNDS][3];
19 } WBWFLEA_ENCODINGS_FOR_DECRYPTION;
```

동작 순서

다음과 같이 외부인코딩 A와 B 바이너리를 입력받아 암호 연산용 또는 복호 연산용 네트워크 랜덤인코딩을 생성한다.

단계 1		단계 2
DRBG 설정 (32바이트 시드 입력) drbg_setup	⟶	외부인코딩 A, B 바이너리를 이용하여 랜덤인코딩 생성 wbwflea_gen_encodings_for_encryption wbwflea_gen_encodings_for_decryption

함수 입/출력

함수명	입력	출력
wbwflea_gen_encodings_for_encryption	암호 연산용 외부인코딩 바이너리	WBWFLEA_ENCODINGS_FOR_ENCRYPTION
wbwflea_gen_encodings_for_decryption	복호 연산용 외부인코딩 바이너리	WBWFLEA_ENCODINGS_FOR_DECRYPTION

예제코드

Code 6.10: 암/복호 연산용 네트워크 랜덤인코딩 생성

```
/* drbg setup */
ge{\Nr}and_bytes_using_randf(seed, 32);drbg_setup(seed);

/* random networked encodings generation (for encryption) */
WBWFLEA_ENCODINGS_FOR_ENCRYPTION enc_ctx;
wbwflea_gen_encodings_for_encryption(&enc_ctx, "encA.bin", "encB.bin");

/* drbg setup */
ge{\Nr}and_bytes_using_randf(seed, 32);drbg_setup(seed);

/* random networked encodings generation (for decryption) */
WBWFLEA_ENCODINGS_FOR_DECRYPTION dec_ctx;
wbwflea_gen_encodings_for_decryption(&dec_ctx, "decA.bin", "decB.bin");
```

3.6.3 암/복호 연산용 화이트박스 키테이블 생성

`wbwflea_tables.*`에 정의되어 있다.

암/복호 연산용 화이트박스 키테이블 구조체

암/복호 연산용 화이트박스 키테이블 구조체는 다음과 같다.

Code 6.11: 화이트박스 키테이블 구조체

```
typedef struct {
byte ETA[WBWFLEA_ROUNDS][3][8][512];
byte ETR[WBWFLEA_ROUNDS][3][8][256];
} WBWFLEA_ENCRYPTION_TABLE;

typedef struct {
byte DTR[WBWFLEA_ROUNDS][3][8][256];
byte DTS[WBWFLEA_ROUNDS][3][8][512];
} WBWFLEA_DECRYPTION_TABLE;
```

동작 순서

다음과 같이 라운드키와 네트워크 랜덤인코딩을 입력받아 암/복호 연산용 화이트박스 키테이블 바이너리를 생성한다.

단계 1		단계 2		단계 3
라운드키, 네트워크 랜덤인코딩 입력	⟶	암/복호 연산용 화이트박스 키테이블 생성	⟶	테이블 저장 (바이너리 파일)
		wbwflea_gen_encryption_table		wbwflea_write_encryption_table
		wbwflea_gen_decryption_table		wbwflea_write_decryption_table

함수 입/출력

함수명	입력	출력
wbwflea_gen_encryption_table	WBWFLEA_ENCODINGS_FOR_ENCRYPTION, WFLEA_CTX	WBWFLEA_ENCRYPTION_TABLE
wbwflea_gen_decryption_table	WBWFLEA_ENCODINGS_FOR_DECRYPTION, WFLEA_CTX	WBWFLEA_DECRYPTION_TABLE
wbwflea_write_encryption_table	WBWFLEA_ENCRYPTION_TABLE	바이너리 파일
wbwflea_write_decryption_table	WBWFLEA_DECRYPTION_TABLE	바이너리 파일

- 함수 wbwflea_gen_encryption_table은 라운드키 구조체 WFLEA_CTX와 암호 연산에 사용되는 랜덤인코딩 구조체 WBWFLEA_ENCODINGS_FOR_ENCRYPTION을 입력받아 생성한다. 생성된 구조체 WBWFLEA_ENCRYPTION_TABLE에 저장한다.

- 함수 wbwflea_gen_decryption_table는 라운드키 구조체 WFLEA_CTX와 복호 연산에 사용되는 랜덤인코딩 구조체 WBWFLEA_ENCODINGS_FOR_DECRYPTION을 입력받고 생성한다. 생성된 DTS와 구조체 WBWFLEA_DECRYPTION_TABLE에 저장한다.

- 함수 wbwflea_write_encryption_table와 wbwflea_write_decryption_table각각 구조체 WBWFLEA_ENCRYPTION_TABLE와 WBWFLEA_DECRYPTION_TABLE을 바이너리 파일로 저장하는 함수이다.

예제코드

Code 6.12: 암/복호 연산용 화이트박스 키테이블 바이너리 생성

```
/* encryption table generation with external encoding,
 random networked encodings, roundkey */
WBWFLEA_ENCRYPTION_TABLE enc_tab;
wbwflea_gen_encryption_table(&enc_tab, &enc_ctx, &wflea_ctx);
wbwflea_write_encryption_table(&enc_tab, "encTab.bin");

/* decryption table generation with external encoding,
 random networked encodings, roundkey */
WBWFLEA_DECRYPTION_TABLE dec_tab;
wbwflea_gen_decryption_table(&dec_tab, &dec_ctx, &wflea_ctx);
wbwflea_write_decryption_table(&dec_tab, "decTab.bin");
```

3.6.4 화이트박스 암/복호 연산

`wbwflea_encrypt.*`에 정의되어 있다.

동작 순서

다음과 같이 화이트박스 연산을 수행한다.

단계 1. 외부인코딩 준비 (`wbwflea_read_ext_encoding`)

단계 2. 외부인코딩 연산 (`wbwflea_ext_transform`)

단계 3. 화이트박스 키테이블 준비 (`wbwflea_read_encryption_table` 또는 `wbwflea_read_decryption_table`)

단계 4. 화이트박스 암/복호 연산 (`wbwflea_encryptwb` 또는 `wbwflea_decryptwb`)

단계 5. 외부인코딩 연산 (`wbwflea_ext_transform`)

함수 입/출력

함수명	입력	출력
wbwflea_read_ext_encoding (wbwflea_ext_transform.c)	외부인코딩 바이너리	WBWFLEA_EXT_ENCODING
wbwflea_ext_transform (wbwflea_ext_transform.c)	WBWFLEA_EXT_ENCODING, flag, 128-bit data	외부인코딩을 적용한 128-bit data
wbwflea_read_encryption_table (wbwflea_tables.c)	암호 연산용 랜덤테이블 바이너리	WBWFLEA_ENCRYPTION_TABLE
wbwflea_read_decryption_table (wbwflea_tables.c)	복호 연산용 랜덤테이블 바이너리	WBWFLEA_DECRYPTION_TABLE
wbwflea_encryptwb	WBWFLEA_ENCRYPTION_TABLE, 128-bit data	128-bit encrypted data
wbwflea_decryptwb	WBWFLEA_DECRYPTION_TABLE, 128-bit data	128-bit decrypted data

- 함수 wbwflea_read_ext_encoding는 외부인코딩이 저장된 바이너리 파일을 입력받아 구조체 WBWFLEA_EXT_ENCODING에 저장한다.

- 함수 wbwflea_ext_transform는 데이터, flag, 구조체 WBWFLEA_EXT_ENCODING을 입력받아 데이터에 외부인코딩을 적용하는 함수이다. 이때 flag = 0 이면 구조체 WBWFLEA_EXT_ENCODING의 f를 사용하고, flag = 1 이면 구조체 WBWFLEA_EXT_ENCODING의 f_inv를 사용한다.

- 함수 wbwflea_read_encryption_table과 함수 wbwflea_read_decryption_table은 각각 랜덤인코딩이 저장된 암/복호 연산용 바이너리 파일을 입력받아, 구조체 WBWFLEA_ENCRYPTION_TABLE과 WBWFLEA_DECRYPTION_TABLE에 저장하는 함수이다.

- 함수 wbwflea_encryptwb와 함수 wbwflea_decryptwb는 각각 구조체 WBWFLEA_ENCRYPTION_TABLE과 WBWFLEA_DECRYPTION_TABLE를 입력받아 각각 16바이트 데이터의 Whitebox WF-LEA 암호 연산, Whitebox WF-LEA 복호 연산을 수행한다. 이때 Whitebox WF-LEA 암호 연산에서는 라운드 함수 wbwflea_roundwb를 사용하며, Whitebox WF-LEA 복호 연산에서는 라운드 함수 wbwflea_invroundwb를 사용한다.

예제코드

Code 6.13: 화이트박스 암호 연산

```
byte wbd_dat[16] = {0x0,}; memcpy(wbd_dat, dat2, 16);
printf("C"); show_bytes(wbd_dat, 16);

/* external excoding A performed in whitebox-secure zone */
WBWFLEA_EXT_ENCODING Ad;
wbwflea_read_ext_encoding(&Ad, "decA.bin");
wbwflea_ext_transform(&Ad, (word*)wbd_dat, 0);
printf("A(C) "); show_bytes(wbd_dat, 16);

/* whitebox encryption */
WBWFLEA_DECRYPTION_TABLE dec_tab;
wbwflea_read_decryption_table(&dec_tab, "decTab.bin");
wbwflea_decryptwb(&dec_tab, (byte*)wbd_dat);
printf("Dwb(A(C))"); show_bytes(wbd_dat, 16);

/* external excoding B performed in whitebox-secure zone */
WBWFLEA_EXT_ENCODING Bd;
wbwflea_read_ext_encoding(&Bd, "decA.bin");
wbwflea_ext_transform(&Bd, (word*)wbd_dat, 0);
printf("B(Dwb(A(C))) "); show_bytes(wbd_dat, 16);
```

Code 6.14: 화이트박스 복호 연산


```
byte wbd_dat[16] = {0x0,}; memcpy(wbd_dat, dat2, 16);
printf("C"); show_bytes(wbd_dat, 16);

/* external excoding A performed in whitebox-secure zone */
WBWFLEA_EXT_ENCODING Ad;
wbwflea_read_ext_encoding(&Ad, "decA.bin");
wbwflea_ext_transform(&Ad, (word*)wbd_dat, 0);
printf("A(C) "); show_bytes(wbd_dat, 16);

/* whitebox encryption */
WBWFLEA_DECRYPTION_TABLE dec_tab;
wbwflea_read_decryption_table(&dec_tab, "decTab.bin");
wbwflea_decryptwb(&dec_tab, (byte*)wbd_dat);
printf("Dwb(A(C))"); show_bytes(wbd_dat, 16);

/* external excoding B performed in whitebox-secure zone */
WBWFLEA_EXT_ENCODING Bd;
wbwflea_read_ext_encoding(&Bd, "decA.bin");
wbwflea_ext_transform(&Bd, (word*)wbd_dat, 0);
printf("B(Dwb(A(C))) "); show_bytes(wbd_dat, 16);
```


부록 A 테스트벡터

제 1절 Whitebox WF-LEA

1.1 Whitebox WF-LEA128

```
[test: Whitebox WF-LEA128 encryption]
block bit length = 128
key bit length = 128
[key]
```

```
9c:c4:ea:59:4b:85:f8:15:7e:e0:d1:82:e9:9d:4a:1b:
[roundkeys]
[00] d3ef4242:ed7deb80:ffe83ccb:3df18824:d0b1bc7b:15c2ba3f:
[01] f16764cb:dff4ed11:62784429:576b13fa:21a31cde:97a7cf09:
[02] 0f11a915:77c40dd1:c6c85504:46b186f0:874c3fd4:cc21a103:
[03] 8097eb81:bed72c93:d2d54071:af315bf5:5fe4d964:62e933cc:
[04] 4fdb9ce5:41b74e1a:1afcc564:ecaf31be:afcf2d3e:93b220bd:
[05] 8cf1cd37:89905f59:693f74e7:1613ab75:55300548:4b1ef5d5:
[06] f3f19cd0:0ea8465b:13927d2c:d437827d:2e90d91a:082e679d:
[07] 991ff790:d35a274e:a85a3ce5:56ea93d6:6f3d1bad:17cdfb79:
[08] 74ae801b:adfea155:79c5ada9:dde7d9ba:fda19351:d06cc89f:
[09] 169be75e:6176b6f3:75903352:8550859e:55eceb8f:9c7d44ad:
[10] 9c36ddf2:4c78b561:ad66c85b:2c38011d:71e5d60f:31c4ac40:
[11] e8869386:8b76c240:f2bac335:1d75a646:a185e80c:7d499972:
[12] eb60698c:bfb5f2c4:1bebb4c0:f90cb424:e23ba781:3e472e52:
[13] 4bfa4393:578c0d3d:9a15ab89:41c3f672:7eb4fc52:9809547a:
[14] 9da633cc:64ddb72b:a50c14ba:61affa76:722ce3f3:3b25c975:
[15] d9081879:f4b43796:759b7857:87280b3e:83ea79d2:b626e54b:
[16] 2c0f2c38:c214318e:39b35043:cf8425b6:1d06b7a8:41f93a25:
[17] a7455c33:3200f5af:53d3d5bb:02a64dfa:f39beacc:04451667:
[18] 65ce2d93:f187a221:116e0292:e023ee1d:6dadfec8:8acfdb6a:
[19] 9cbb7df6:4fd3425c:b1dadc3c:072dfc14:af443f7a:d9cbb504:
[20] 233d166c:71a01071:d4bb959b:b813c4b8:de417716:7836d490:
[21] 8da04f91:7366c575:10cc209b:e073e10c:39a4d400:b3ec5bd6:
[22] abad5577:7e46eb14:96b87406:22ade1d2:0acf86a6:456da989:
[23] a97dc3f4:04e9c4b1:0da6e70c:e6b24f0b:c3f3c779:5cb0bbbb:

[WF-LEA128 encryption]
P 06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:

[ 0] a78c6500:6fb99c41:c37671e5:771ce306:
[ 1] 70e6360e:3d0ef9d4:58721329:a78c6500:
[ 2] 85264195:48d45215:095d7e20:70e6360e:
[ 3] 6a5135f7:ca0369c1:cd1915a0:85264195:
[ 4] 7da1db62:1f95ae86:cf2d5338:6a5135f7:
[ 5] ac10690e:727f4e95:576da2d2:7da1db62:
[ 6] 71fd59b9:472a3aa3:fdf1a718:ac10690e:
[ 7] a5982cfa:a4d459d8:89d549e5:71fd59b9:
[ 8] c34addb4:818a2426:42c0cd7b:a5982cfa:
[ 9] 9b9b7f6d:cddd5302:6a2231e9:c34addb4:
[10] a7120512:6d36ae62:41cacb3b:9b9b7f6d:
[11] aa056c6b:a7e25ed6:d8e4412a:a7120512:
[12] 7963f2b3:26ef96f9:7a86a23d:aa056c6b:
[13] fa99c947:ffc1fc8d:06c7d2d0:7963f2b3:
[14] b88c6204:ee11b086:36e62d9d:fa99c947:
[15] 54031af8:a26ac77b:6039700b:b88c6204:
[16] 165b6bb1:aa5cb767:8ed6a3f8:54031af8:
```

```
[17] f4f49492:f4300286:79b26aba:165b6bb1:
[18] e4b3512d:dbf77c25:b61688a9:f4f49492:
[19] 58d6a818:b0db48a6:28d23b2d:e4b3512d:
[20] ce2e967a:97a916e6:12631a3f:58d6a818:
[21] bd5afc50:83cbb18d:a2e05841:ce2e967a:
[22] 0a078029:f4ae0bf8:466e63db:bd5afc50:
[23] 84264d27:24cf28ce:acf0fd91:0a078029:

E(P) 27:4d:26:84:ce:28:cf:24:91:fd:f0:ac:29:80:07:0a:

[ 0] 0a078029:f4ae0bf8:466e63db:bd5afc50:
[ 1] bd5afc50:83cbb18d:a2e05841:ce2e967a:
[ 2] ce2e967a:97a916e6:12631a3f:58d6a818:
[ 3] 58d6a818:b0db48a6:28d23b2d:e4b3512d:
[ 4] e4b3512d:dbf77c25:b61688a9:f4f49492:
[ 5] f4f49492:f4300286:79b26aba:165b6bb1:
[ 6] 165b6bb1:aa5cb767:8ed6a3f8:54031af8:
[ 7] 54031af8:a26ac77b:6039700b:b88c6204:
[ 8] b88c6204:ee11b086:36e62d9d:fa99c947:
[ 9] fa99c947:ffc1fc8d:06c7d2d0:7963f2b3:
[10] 7963f2b3:26ef96f9:7a86a23d:aa056c6b:
[11] aa056c6b:a7e25ed6:d8e4412a:a7120512:
[12] a7120512:6d36ae62:41cacb3b:9b9b7f6d:
[13] 9b9b7f6d:cddd5302:6a2231e9:c34addb4:
[14] c34addb4:818a2426:42c0cd7b:a5982cfa:
[15] a5982cfa:a4d459d8:89d549e5:71fd59b9:
[16] 71fd59b9:472a3aa3:fdf1a718:ac10690e:
[17] ac10690e:727f4e95:576da2d2:7da1db62:
[18] 7da1db62:1f95ae86:cf2d5338:6a5135f7:
[19] 6a5135f7:ca0369c1:cd1915a0:85264195:
[20] 85264195:48d45215:095d7e20:70e6360e:
[21] 70e6360e:3d0ef9d4:58721329:a78c6500:
[22] a78c6500:6fb99c41:c37671e5:771ce306:
[23] 771ce306:361dcf6e:10cc1cac:4ef75468:

D(E(P))  06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
equal.

[seed for external encoding A]
70:bb:a6:40:b9:96:14:7d:0c:c1:b8:10:ec:93:fa:9a:
cf:8a:c6:0f:55:5d:e7:eb:a4:58:63:e9:ea:68:21:c2:
== External encoding A for encryption ==

[0,0] fc 2 4 6 7 5 9 d b f 1 3 8 0 a e
  finv d a 1 b 2 5 3 4 c 6 e 8 0 7 f 9
[0,1] fd 6 3 f 5 c a 4 1 e 9 7 0 8 b 2
  finv c 8 f 2 7 4 1 b d a 6 e 5 0 9 3
[0,2] fd 6 5 c 0 f b 1 a 8 4 3 7 9 2 e
```

```
  finv 4 7 e b a 2 1 c 9 d 8 6 3 0 f 5
[0,3] fd 6 9 7 5 e 0 4 3 8 2 c a 1 b f
  finv 6 d a 8 7 4 1 3 9 2 c e b 0 5 f
[0,4] f8 5 e d 9 6 2 1 3 7 4 b a c f 0
  finv f 7 6 8 a 1 5 9 0 4 c b d 3 2 e
[0,5] f4 e 8 2 0 1 9 b 7 c a f 6 3 5 d
  finv 4 5 3 d 0 e c 8 2 6 a 7 9 f 1 b
[0,6] f9 2 8 e 5 4 6 b c 1 d 0 3 a 7 f
  finv b 9 1 c 5 4 6 e 2 0 d 7 8 a 3 f
[0,7] f3 c 7 b e f 1 9 6 a d 0 4 5 8 2
  finv b 6 f 0 c d 8 2 e 7 9 3 1 a 4 5
[1,0] fb c 0 9 2 3 4 d 5 a 7 1 6 f 8 e
  finv 2 b 4 5 6 8 c a e 3 9 0 1 7 f d
[1,1] fc e d 8 4 5 a b 2 f 0 9 6 3 1 7
  finv a e 8 d 4 5 c f 3 b 6 7 0 2 1 9
[1,2] f0 b 1 3 5 6 8 e d f 9 4 c 2 a 7
  finv 0 2 d 3 b 4 5 f 6 a e 1 c 8 7 9
[1,3] fe 5 7 2 3 c 8 0 a d 6 9 b 4 f 1
  finv 7 f 3 4 d 1 a 2 6 b 8 c 5 9 0 e
[1,4] fd c 7 e 9 5 2 b 0 a 4 1 8 6 3 f
  finv 8 b 6 e a 5 d 2 c 4 9 7 1 0 3 f
[1,5] fb 4 5 f 6 8 d a e 7 9 c 3 1 2 0
  finv f d e c 1 2 4 9 5 a 7 0 b 6 8 3
[1,6] f2 1 0 4 8 a b d f 3 9 6 c e 5 7
  finv 2 1 0 9 3 e b f 4 a 5 6 c 7 d 8
[1,7] f6 0 8 b 1 c f 7 3 5 9 e a 4 d 2
  finv 1 4 f 8 d 9 0 7 2 a c 3 5 e b 6
[2,0] fc 2 e 8 b 1 4 0 6 5 9 7 d 3 f a
  finv 7 5 1 d 6 9 8 b 3 a f 4 0 c 2 e
[2,1] f7 e 8 f 2 3 9 a 6 5 4 b d 1 c 0
  finv f d 4 5 a 9 8 0 2 6 7 b e c 1 3
[2,2] f1 a 9 6 e 8 0 b f 3 d 7 c 2 4 5
  finv 6 0 d 9 e f 3 b 5 2 1 7 c a 4 8
[2,3] fd a 0 5 3 8 6 7 f 1 c e 2 b 9 4
  finv 2 9 c 4 f 3 6 7 5 e 1 d a 0 b 8
[2,4] f8 a 0 9 7 5 f d 2 b 3 e c 1 6 4
  finv 2 d 8 a f 5 e 4 0 3 1 9 c 7 b 6
[2,5] fd 7 b 0 f 8 1 5 3 2 a e 9 6 4 c
  finv 3 6 9 8 e 7 d 1 5 c a 2 f 0 b 4
[2,6] f8 1 c 7 f 5 3 a e d 2 9 b 0 6 4
  finv d 1 a 6 f 5 e 3 0 b 7 c 2 9 8 4
[2,7] fa 9 3 e f d b 7 1 6 2 4 0 8 5 c
  finv c 8 a 2 b e 9 7 d 1 0 6 f 5 3 4
[3,0] f0 a 5 9 1 f b 3 d c 7 8 2 e 6 4
  finv 0 4 c 7 f 2 e a b 3 1 6 9 8 d 5
[3,1] f3 b f a 5 6 2 c 9 d 4 1 7 8 0 e
  finv e b 6 0 a 4 5 c d 8 3 1 7 9 f 2
```

```
[3,2] f8 3 7 a 4 c f e 6 d 9 b 2 1 0 5
  finv e d c 1 4 f 8 2 0 a 3 b 5 9 7 6
[3,3] f6 4 0 3 a 5 9 f 2 e c 1 b 7 8 d
  finv 2 b 8 3 1 5 0 d e 6 4 c a f 9 7
[3,4] fa 0 9 7 b 5 2 3 4 e 1 f d c 6 8
  finv 1 a 6 7 8 5 e 3 f 2 0 4 d c 9 b
[3,5] f3 0 b e 6 d c 7 9 5 a 4 f 8 2 1
  finv 1 f e 0 b 9 4 7 d 8 a 2 6 5 3 c
[3,6] f3 a 6 1 5 f c e b 2 7 4 9 8 0 d
  finv e 3 9 0 b 4 2 a d c 1 8 6 f 7 5
[3,7] f4 9 d 0 e c 1 2 3 6 b a f 7 8 5
  finv 3 6 7 8 0 f 9 d e 1 b a 5 2 4 c

[seed for external encoding B]
eb:40:e9:0b:84:68:57:45:0d:e2:54:2f:d6:f9:63:72:
de:15:1b:65:b5:3f:1c:96:c1:58:81:61:76:e6:06:f3:
== External encoding B for encryption ==

[0,0] fd 0 2 6 7 8 1 f 4 b 5 9 a c e 3
  finv 1 6 2 f 8 a 3 4 5 b c 9 d 0 e 7
[0,1] fe d c 5 9 b 1 7 3 f 0 6 4 8 2 a
  finv a 6 e 8 c 3 b 7 d 4 f 5 2 1 0 9
[0,2] f6 8 a 7 d 9 1 0 b c f 4 3 5 e 2
  finv 7 6 f c b d 0 3 1 5 2 8 9 4 e a
[0,3] f1 c 5 6 2 9 e 8 3 0 a 4 d b f 7
  finv 9 0 4 8 b 2 3 f 7 5 a d 1 c 6 e
[0,4] f5 c 0 2 9 6 f 3 4 b 7 1 e a d 8
  finv 2 b 3 7 8 0 5 a f 4 d 9 1 e c 6
[0,5] f9 f d c b 3 4 0 1 2 a 8 5 e 7 6
  finv 7 8 9 5 6 c f e b 0 a 4 3 2 d 1
[0,6] f5 f 6 9 e b 8 3 c 2 7 4 d 1 0 a
  finv e d 9 7 b 0 2 a 6 3 f 5 8 c 4 1
[0,7] f2 9 0 3 f a 7 c b e 1 8 6 d 4 5
  finv 2 a 0 3 e f c 6 b 1 5 8 7 d 9 4
[1,0] f8 e 5 c f 6 d 4 b 3 7 a 0 9 2 1
  finv c f e 9 7 2 5 a 0 d b 8 3 6 1 4
[1,1] f3 f 2 d c b 6 4 7 0 1 e 8 a 9 5
  finv 9 a 2 0 7 f 6 8 c e d 5 4 3 b 1
[1,2] f3 6 d 0 b 5 c 1 4 2 a f 8 e 9 7
  finv 3 7 9 0 8 5 1 f c e a 4 6 2 d b
[1,3] fb 2 9 6 f 7 e a 0 d 5 3 1 8 4 c
  finv 8 c 1 b e a 3 5 d 2 7 0 f 9 6 4
[1,4] f8 d e f 3 5 9 6 c 2 a 1 0 b 7 4
  finv c b 9 4 f 5 7 e 0 6 a d 8 1 2 3
[1,5] fe 4 d 3 f 9 b 7 2 6 0 c 8 5 1 a
  finv a e 8 3 1 d 9 7 c 5 f 6 b 2 0 4
[1,6] f2 0 b 5 8 e d c 7 6 1 a 9 4 3 f
```

```
  finv 1 a 0 e d 3 9 8 4 c b 2 7 6 5 f
[1,7] f3 0 7 b 9 4 5 a f 8 d 1 c 6 e 2
  finv 1 b f 0 5 6 d 2 9 4 7 3 c a e 8
[2,0] f9 c b 5 0 e 4 d f 7 1 2 a 8 6 3
  finv 4 a b f 6 3 e 9 d 0 c 2 1 7 5 8
[2,1] fa f 8 5 b 7 1 9 0 4 6 d e 2 c 3
  finv 8 6 d f 9 3 a 5 2 7 0 4 e b c 1
[2,2] fc 0 a 8 f e 2 1 3 7 9 4 b d 5 6
  finv 1 7 6 8 b e f 9 3 a 2 c 0 d 5 4
[2,3] f9 8 5 1 2 0 f 3 b 4 a 6 c 7 e d
  finv 5 3 4 7 9 2 b d 1 0 a 8 c f e 6
[2,4] f1 7 0 d f 8 b 2 4 3 5 9 6 c e a
  finv 2 0 7 9 8 a c 1 5 b f 6 d 3 e 4
[2,5] ff 9 e b 5 0 3 2 c 6 7 d a 4 1 8
  finv 5 e 7 6 d 4 9 a f 1 c 3 8 b 2 0
[2,6] fc 6 e b d f 1 2 4 9 0 5 8 7 3 a
  finv a 6 7 e 8 b 1 d c 9 f 3 0 4 2 5
[2,7] fc 8 e 9 6 3 7 b 5 0 d 1 f 4 a 2
  finv 9 b f 5 d 8 4 6 1 3 e 7 0 a 2 c
[3,0] f5 4 7 c f 6 2 b 3 a 0 e d 9 1 8
  finv a e 6 8 1 0 5 2 f d 9 7 3 c b 4
[3,1] f1 f 4 8 b 7 e 3 2 d 6 5 0 c 9 a
  finv c 0 8 7 2 b a 5 3 e f 4 d 9 6 1
[3,2] fb 9 f d 0 3 5 6 4 a e c 1 7 2 8
  finv 4 c e 5 8 6 7 d f 1 9 0 b 3 a 2
[3,3] ff 6 c a 7 0 d 1 2 5 e 9 3 b 4 8
  finv 5 7 8 c e 9 1 4 f b 3 d 2 6 a 0
[3,4] f3 8 9 6 2 1 7 a b 0 f e 4 c 5 d
  finv 9 5 4 0 c e 3 6 1 2 7 8 d f b a
[3,5] fa e 9 6 8 4 7 0 c f d 1 2 3 5 b
  finv 7 b c d 5 e 3 6 4 2 0 f 8 a 1 9
[3,6] fc 1 5 8 d f 7 a 2 6 b 4 9 3 0 e
  finv e 1 8 d b 2 9 6 3 c 7 a 0 4 f 5
[3,7] f2 7 0 b 6 c 5 a 3 4 8 f e 1 9 d
  finv 2 d 0 8 9 6 4 1 a e 7 3 5 f c b

[seed for encryption encoding]
7a:10:51:b3:82:55:a1:30:3a:28:47:9a:80:5c:60:9d:
00:6a:6d:3a:b7:ac:fb:f7:e2:92:9c:da:4e:00:0f:61:

[Whitebox WF-LEA128 encryption]
P06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
A(P) d9:bc:ea:9b:a8:b7:46:bb:4d:ac:9c:98:2d:54:13:e0:

[ 0] 1d38b14f:0962b3e1:9c82d713:9beabcd9:
[ 1] a85e70c5:2351085a:5b33b558:1d38b14f:
[ 2] fde8ed92:c82ad631:51c261bc:a85e70c5:
[ 3] c7d32d9c:ebfd9a02:1070aac6:fde8ed92:
```

```
[ 4] d933c223:fa0a2a29:ce98ba4b:c7d32d9c:
[ 5] fd2301d6:f92af9a0:4dcb178a:d933c223:
[ 6] 00f2b155:c6e1c74d:d9dad244:fd2301d6:
[ 7] c515dcb0:48a7612a:ca4dd187:00f2b155:
[ 8] e43735d2:2855a9a3:e58a7e0c:c515dcb0:
[ 9] 3f78c8c0:e87685c8:fad72ba7:e43735d2:
[10] 34471025:1c357976:2aa720ba:3f78c8c0:
[11] 80a6a459:a10a57df:9a985686:34471025:
[12] 2b749f68:bb3f80aa:dc24f30b:80a6a459:
[13] 08fe91b9:f2e8f086:2515fe45:2b749f68:
[14] e992d3c3:44c40dc7:44b77536:08fe91b9:
[15] e8abac76:dce93f6f:a96234e7:e992d3c3:
[16] 835c9c03:fdcd2810:6a834265:e8abac76:
[17] 4429f74f:76c3c332:f48eec63:835c9c03:
[18] d06c94ca:b2037615:8cb7fe7a:4429f74f:
[19] 2b767e24:b6366de3:27167f57:d06c94ca:
[20] 5abf6e92:1c723f24:e7aed8be:2b767e24:
[21] 33e28358:4baf01ce:1bc908fc:5abf6e92:
[22] 2776f48d:3c9c19d0:a571a3d3:33e28358:
[23] bb95b4e4:fdb31c41:e0026d7a:2776f48d:

Ewb(A(P))e4:b4:95:bb:41:1c:b3:fd:7a:6d:02:e0:8d:f4:76:27:
B(Ewb(A(P))) 27:4d:26:84:ce:28:cf:24:91:fd:f0:ac:29:80:07:0a:
equal.

[seed for external encoding A]
a6:69:10:e3:4f:e9:8e:60:07:98:e6:f6:cf:84:38:b2:
ba:7f:29:51:13:4b:68:d0:00:c9:a4:eb:22:39:cd:44:
== External encoding A for decryption ==

[0,0] fe a 5 3 d 8 4 b 9 f c 0 7 2 1 6
  finv b e d 3 6 2 f c 5 8 1 7 a 4 0 9
[0,1] ff 3 9 0 2 4 b 6 5 7 a 8 e c d 1
  finv 3 f 4 1 5 8 7 9 b 2 a 6 d e c 0
[0,2] fd 5 9 a 1 4 7 0 6 b 2 f c e 3 8
  finv 7 4 a e 5 1 8 6 f 2 3 9 c 0 d b
[0,3] f5 4 1 c 0 f 7 8 a 2 e 9 3 d 6 b
  finv 4 2 9 c 1 0 e 6 7 b 8 f 3 d a 5
[0,4] f5 e 6 1 3 c d a 2 7 8 0 b f 9 4
  finv b 3 8 4 f 0 2 9 a e 7 c 5 6 1 d
[0,5] f6 c 1 a 9 7 3 4 2 e b 8 5 0 d f
  finv d 2 8 6 7 c 0 5 b 4 3 a 1 e 9 f
[0,6] fe 0 1 7 d c 2 3 8 a 6 5 f 4 b 9
  finv 1 2 6 7 d b a 3 8 f 9 e 5 4 0 c
[0,7] f6 1 b 4 7 0 c e 8 d 3 a 5 9 f 2
  finv 5 1 f a 3 c 0 4 8 d b 2 6 9 7 e
[1,0] f2 a 5 e c 8 b 1 f d 3 7 9 0 6 4
  finv d 7 0 a f 2 e b 5 c 1 6 4 9 3 8
```

```
[1,1] f0 7 a 4 e 5 9 f 8 d c b 1 2 3 6
  finv 0 c d e 3 5 f 1 8 6 2 b a 9 4 7
[1,2] f3 1 9 a d c 8 b e 6 4 f 0 2 7 5
  finv c 1 d 0 a f 9 e 6 2 3 7 5 4 8 b
[1,3] f7 c 5 8 2 3 a 6 0 d e 1 4 b 9 f
  finv 8 b 4 5 c 2 7 0 3 e 6 d 1 9 a f
[1,4] f5 9 e c 2 b 0 7 a 1 3 6 f d 8 4
  finv 6 9 4 a f 0 b 7 e 1 8 5 3 d 2 c
[1,5] f3 9 d f 2 0 1 5 4 6 b 8 c e a 7
  finv 5 6 4 0 8 7 9 f b 1 e a c 2 d 3
[1,6] f8 f c 2 0 5 b 3 6 1 e d 9 a 4 7
  finv 4 9 3 7 e 5 8 f 0 c d 6 2 b a 1
[1,7] f6 f d b 2 9 a e 5 c 1 3 0 8 7 4
  finv c a 4 b f 8 0 e d 5 6 3 9 2 7 1
[2,0] f6 3 8 b f 1 9 5 7 4 c d 0 2 a e
  finv c 5 d 1 9 7 0 8 2 6 e 3 a b f 4
[2,1] f1 c 3 2 5 0 b 7 9 a 6 d 4 f e 8
  finv 5 0 3 2 c 4 a 7 f 8 9 6 1 b e d
[2,2] fa c 6 8 3 2 7 4 e 5 d 9 1 f b 0
  finv f c 5 4 7 9 2 6 3 b 0 e 1 a 8 d
[2,3] fa 1 3 f 5 4 d 7 e 2 b 6 0 9 8 c
  finv c 1 9 2 5 4 b 7 e d 0 a f 6 8 3
[2,4] fc b 0 1 f d 4 2 9 e 6 5 8 3 7 a
  finv 2 3 7 d 6 b a e c 8 f 1 0 5 9 4
[2,5] f6 d 0 5 4 1 9 b 8 e a 3 f 7 2 c
  finv 2 5 e b 4 3 0 d 8 6 a 7 f 1 9 c
[2,6] f4 f 5 a 3 d 1 e 6 8 0 2 7 c 9 b
  finv a 6 b 4 0 2 8 c 9 e 3 f d 5 7 1
[2,7] fb 1 5 2 6 d 3 e 0 4 a f 8 7 9 c
  finv 8 1 3 6 9 2 4 d c e a 0 f 5 7 b
[3,0] f0 f 5 b 4 9 8 a c d 2 e 6 3 1 7
  finv 0 e a d 4 2 c f 6 5 7 3 8 9 b 1
[3,1] f8 6 c 2 7 3 1 5 0 e f 9 a b 4 d
  finv 8 6 3 5 e 7 1 4 0 b c d 2 f 9 a
[3,2] f4 c b 5 9 0 6 2 7 1 d e f 8 3 a
  finv 5 9 7 e 0 3 6 8 d 4 f 2 1 a b c
[3,3] f9 8 b 4 e 7 d 6 a 5 c 2 f 0 3 1
  finv d f b e 3 9 7 5 1 0 8 2 a 6 4 c
[3,4] fb 7 3 0 c 9 a 5 4 1 8 6 f e d 2
  finv 3 9 f 2 8 7 b 1 a 5 6 0 4 e d c
[3,5] fe b 3 c 5 2 a d 9 4 0 7 8 f 1 6
  finv a e 5 2 9 4 f b c 8 6 1 3 7 0 d
[3,6] f2 4 a 8 7 b 6 5 1 c e d 9 3 f 0
  finv f 8 0 d 1 7 6 4 3 c 2 5 9 b a e
[3,7] f0 8 1 2 b e f a 3 d 5 9 4 7 6 c
  finv 0 2 3 8 c a e d 1 b 7 4 f 9 5 6
```

```
[seed for external encoding B]
0a:bb:7a:ed:51:5d:48:9c:b0:3b:02:b8:e4:c3:10:19:
eb:e8:cf:3e:7c:d3:04:46:0b:aa:63:dd:ab:ce:04:0e:
== External encoding B for decryption ==

[0,0] fa 1 7 f 4 6 8 2 9 3 0 d 5 c b e
  finv a 1 7 9 4 c 5 2 6 8 0 e d b f 3
[0,1] f6 e b 5 a c 9 8 0 2 d f 7 1 4 3
  finv 8 d 9 f e 3 0 c 7 6 4 2 5 a 1 b
[0,2] ff 9 a b 8 2 5 6 7 d e 4 0 c 3 1
  finv c f 5 e b 6 7 8 4 1 2 3 d 9 a 0
[0,3] f5 1 9 0 3 a 8 d 4 6 e f c 2 7 b
  finv 3 1 d 4 8 0 9 e 6 2 5 f c 7 a b
[0,4] fb 1 5 a 6 2 7 c 9 8 f 3 0 4 d e
  finv c 1 5 b d 2 4 6 9 8 3 0 7 e f a
[0,5] f0 c 9 f 7 4 6 a 2 1 e 8 b d 5 3
  finv 0 9 8 f 5 e 6 4 b 2 7 c 1 d a 3
[0,6] f9 d 4 a 2 7 3 f 1 5 c e 8 6 b 0
  finv f 8 4 6 2 9 d 5 c 0 3 e a 1 b 7
[0,7] f6 2 7 0 a 5 b c d 9 8 4 e 1 f 3
  finv 3 d 1 f b 5 0 2 a 9 4 6 7 8 c e
[1,0] f4 0 7 9 e 2 8 f d a b 3 5 6 c 1
  finv 1 f 5 b 0 c d 2 6 3 9 a e 8 4 7
[1,1] f0 7 9 a b 6 8 5 3 4 c 2 e f 1 d
  finv 0 e b 8 9 7 5 1 6 2 3 4 a f c d
[1,2] f2 1 8 b d 5 c 0 a 4 f 3 e 6 9 7
  finv 7 1 0 b 9 5 d f 2 e 8 3 6 4 c a
[1,3] f4 9 0 3 7 1 a f 2 8 5 c 6 d b e
  finv 2 5 8 3 0 a c 4 9 1 6 e b d f 7
[1,4] fc 1 6 0 a f 3 b 8 5 e 2 9 d 4 7
  finv 3 1 b 6 e 9 2 f 8 c 4 7 0 d a 5
[1,5] fb 1 8 4 5 0 6 3 d c f a 9 2 e 7
  finv 5 1 d 7 3 4 6 f 2 c b 0 9 8 e a
[1,6] f6 d c 7 4 2 8 f e a 0 9 1 5 3 b
  finv a c 5 e 4 d 0 3 6 b 9 f 2 1 8 7
[1,7] f7 2 9 d 4 0 e 8 5 1 f a 6 3 b c
  finv 5 9 1 d 4 8 c 0 7 2 b e f 3 6 a
[2,0] f8 9 c b e d 1 7 3 a 0 2 5 4 6 f
  finv a 6 b 8 d c e 7 0 1 9 3 2 5 4 f
[2,1] fd 9 7 5 a 4 6 f 1 0 e c 2 3 b 8
  finv 9 8 c d 5 3 6 2 f 1 4 e b 0 a 7
[2,2] ff e 2 3 6 0 4 b a 8 5 7 9 c d 1
  finv 5 f 2 3 6 a 4 b 9 c 8 7 d e 1 0
[2,3] f8 d e 5 1 6 c f b 3 4 2 0 7 a 9
  finv c 4 b 9 a 3 5 d 0 f e 8 6 1 2 7
[2,4] fe d b 7 6 9 0 c 4 2 8 f 5 a 3 1
  finv 6 f 9 e 8 c 4 3 a 5 d 2 7 1 0 b
```

```
[2,5] fd e a 1 3 2 c 9 8 4 5 6 f b 0 7
  finv e 3 5 4 9 a b f 8 7 2 d 6 0 1 c
[2,6] f7 d a 1 b 4 5 2 9 f 8 c 6 0 3 e
  finv d 3 7 e 5 6 c 0 a 8 2 4 b 1 f 9
[2,7] ff c d 7 5 b 2 0 a 1 e 6 3 8 9 4
  finv 7 9 6 c f 4 b 3 d e 8 5 1 2 a 0
[3,0] fe a 7 2 f 4 8 d c 1 3 6 5 0 b 9
  finv d 9 3 a 5 c b 2 6 f 1 e 8 7 0 4
[3,1] f7 1 0 c 9 d 6 a 5 4 e 8 b f 2 3
  finv 2 1 e f 9 8 6 0 b 4 7 c 3 5 a d
[3,2] f6 c 0 3 8 d a 5 7 9 4 2 e b 1 f
  finv 2 e b 3 a 7 0 8 4 9 6 d 1 5 c f
[3,3] f3 9 1 f e 7 2 8 5 4 0 b c d a 6
  finv a 2 6 0 9 8 f 5 7 1 e b c d 4 3
[3,4] f7 f 4 8 9 6 b 0 d a 5 3 1 c e 2
  finv 7 c f b 2 a 5 0 3 4 9 6 d 8 e 1
[3,5] f2 4 b 6 5 a 3 d 7 9 c 0 8 1 f e
  finv b d 0 6 1 4 3 8 c 9 5 2 a 7 f e
[3,6] fd a 6 f 5 3 0 b 8 c 2 4 1 7 9 e
  finv 6 c a 5 b 4 2 d 8 e 1 7 9 0 f 3
[3,7] fe 1 c f 8 2 3 5 0 6 b 7 a 9 d 4
  finv 8 1 5 6 f 7 9 b 4 d c a 2 e 0 3

[seed for decryption encoding]
f0:6b:03:7e:93:a3:42:3e:57:5e:ce:99:83:2f:7e:8c:
50:eb:c4:da:7c:42:dc:e1:eb:57:85:fe:ee:cc:93:f1:

[Whitebox WF-LEA128 decryption]
C 27:4d:26:84:ce:28:cf:24:91:fd:f0:ac:29:80:07:0a:
A(C) 9b:0e:1d:8d:16:5e:c4:d0:a3:cf:cc:a7:cd:a4:e5:0e:

[ 0] 0ee5a4cd:15f886e3:480a4f48:d3185c7d:
[ 1] d3185c7d:01382c70:e6ad1462:003fbde0:
[ 2] 003fbde0:951f0cda:64f26f7e:3a8a3534:
[ 3] 3a8a3534:b31d0363:f78f34e4:fc438502:
[ 4] fc438502:118d26a6:0fa3434f:6bfbeb26:
[ 5] 6bfbeb26:f8cb12da:c0a2b34e:278690a6:
[ 6] 278690a6:88740a50:6b0160ae:a5cc3afb:
[ 7] a5cc3afb:58229670:8f000df3:0d8bdaec:
[ 8] 0d8bdaec:82be7efc:63d9e7ad:437b1261:
[ 9] 437b1261:a71775f2:1c4ba838:9fb7ee11:
[10] 9fb7ee11:abc6a02e:f74096b3:327c0c3b:
[11] 327c0c3b:61d42332:569f7b0a:78c7fc83:
[12] 78c7fc83:c11fcc63:21122e5d:3f744bb1:
[13] 3f744bb1:2edfa0f4:91d284a2:57f6b694:
[14] 57f6b694:af385381:61186c88:c8558e9d:
[15] c8558e9d:d7734d58:4fa4dac7:19607b88:
[16] 19607b88:ea655e34:50e233b8:d50ae47f:
```

```
[17] d50ae47f:cac54775:a53e18a2:cc369421:
[18] cc369421:2c5cd688:6c0187d7:a89ca6c9:
[19] a89ca6c9:58576001:11ea67a4:1f46ab89:
[20] 1f46ab89:ede532d3:21c212cc:60dce43b:
[21] 60dce43b:7d4fa5dc:19971a72:9a06f120:
[22] 9a06f120:ac7858dc:e7336803:2597ae85:
[23] 2597ae85:d01dba54:9d674d42:ffe08a66:

Dwb(A(C))85:ae:97:25:54:ba:1d:d0:42:4d:67:9d:66:8a:e0:ff:
B(Dwb(A(C))) 06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
equal.
```

1.2 Whitebox WF-LEA192

```
[test: Whitebox WF-LEA192 encryption]
block bit length = 128
key bit length = 192
[key]
9c:c4:ea:59:4b:85:f8:15:7e:e0:d1:82:e9:9d:4a:1b:32:36:c1:48:27:82:89:dd:

[roundkeys]
[00] 2887f56d:b4409d5d:dae9d9f1:a5540eea:5f57ae1a:c94ef979:
[01] 9875f9c3:dc21f5ba:3b9b7882:4f1ce658:d325d1fc:7441bbb3:
[02] 6bf6bedf:035ce1d4:4a5040c9:664875ed:a22d3e67:29fdb6c0:
[03] e82f58c3:6efda8de:731a3ae9:481cbff6:90eb238e:bdaa6b72:
[04] 2e0d14c0:d653333b:d4684fce:5d434ce8:dd656f8d:890fbe9c:
[05] 3200a58a:c9f34820:27a16d73:625d2938:1e77e44e:786bc5d9:
[06] 5d83664b:23d8c918:95d68942:55b42132:152cc2d6:606a89ae:
[07] 91d76a5f:8a88e36b:6b27865d:bbefecae:7b1cc720:58afee52:
[08] 120251ae:23b2b04c:4ac1517a:2e6de4bd:c63f9d6f:2121b380:
[09] f6cfcf6f:825ee339:c4c3ee13:71308a39:fb20f240:ad6c857a:
[10] 4332c80c:5c8abd0a:2dc0f6d8:bb164d5e:857ba558:ba0cb11e:
[11] e13b821f:6049e543:aa9b1df9:270bfdb3:219b014c:0d61d410:
[12] 9e33fe91:89255cdb:61ceee75:c6d7d98e:2eb450af:f8535bc3:
[13] 157803f1:1db8d890:87bb5c85:cbaf6a84:4799e800:0a06bc59:
[14] 1730d771:ef201fbc:732e795b:5a13f235:212edd6d:b64f7ad2:
[15] fac8585f:e7573654:3e964552:7e14c6ad:c046fa46:642e61a6:
[16] 14aaca0a:e13fb109:f418ef97:bfb0de22:cf06da07:3d23a7aa:
[17] ff1309a9:73cd3ce0:e7e7d905:49f443d0:acc1c385:70589ffb:
[18] 0bc2c351:8e265687:8218e796:77945816:81333e97:646967be:
[19] 2cebf04b:bbbbe6da:81e44b6e:d4d24df6:b53a28c1:d6e62d02:
[20] 9d728066:6f48c4bb:67994acb:63a53f10:e94fc51a:0a8e28f5:
[21] 0f0c7b20:9fd807cc:c5c4acb0:1d791e92:a18a3ad4:cd73c50f:
[22] bc1be53a:224533c4:4ce90763:08fe42d6:728d4f48:c6008ba1:
[23] 77aa757d:57b3cc81:4bf90cf6:fc89ff3d:9003322d:c77e2785:
[24] ff58e52f:1ba91149:7b3c8a07:acc67229:9b165d67:eba79ecd:
[25] 2e1194ba:0f072a42:de50fa4c:364b24b6:8a4b469a:a175b106:
[26] 45af3d33:a121c168:c3b1c58a:7b8c0a9f:6c2d512a:8bedb150:
[27] 04c4c6d7:ab626c6c:297f96ba:864ab48f:9adae251:4a235bfa:
```

```
[WF-LEA192 encryption]
P06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:

[ 0] f0d13dc3:2d146147:faeaabf8:771ce306:
[ 1] b4b1fab3:2e642b3b:25a59a57:f0d13dc3:
[ 2] 001eb619:654112dd:6c16a5e6:b4b1fab3:
[ 3] dd51bbe7:21d32a12:20c32305:001eb619:
[ 4] b990a1d5:4b99deae:b0d6eaa1:dd51bbe7:
[ 5] f535da1b:b1f623bb:aa7b71a5:b990a1d5:
[ 6] cb4de675:811f7fdc:d32a3b7d:f535da1b:
[ 7] 6451c2cc:a297f68a:caba2614:cb4de675:
[ 8] f1b451ee:ce697354:1ede422e:6451c2cc:
[ 9] 665ddca7:f3d4cb2a:95e77f04:f1b451ee:
[10] 9b1597a9:60682b82:8b8ab769:665ddca7:
[11] 9fc8eef4:abbba40a:82a9b7db:9b1597a9:
[12] 34126c49:a0779dc6:c1ec967b:9fc8eef4:
[13] 736a1dbe:119085f2:03887a25:34126c49:
[14] 16ca3ac6:cde2d425:74a077bc:736a1dbe:
[15] 708a142d:47f14a14:59856142:16ca3ac6:
[16] dfb28815:1cd0fb27:384dab16:708a142d:
[17] 7e93071f:43678857:32abde8d:dfb28815:
[18] 27463c85:e035f7b2:adee99f8:7e93071f:
[19] 77bc6cce:56d87487:d8293b6a:27463c85:
[20] bf39c848:37666a16:0be5e25c:77bc6cce:
[21] e84284b1:a049fe1b:2ca7f049:bf39c848:
[22] cc5ed5ac:b887d560:5aec805d:e84284b1:
[23] 51756556:b4cf22c7:9f458ab4:cc5ed5ac:
[24] 27680ebb:e81bbd0a:8589a466:51756556:
[25] 2c6293e1:b7506e40:9ff876e9:27680ebb:
[26] 7ebbf500:02cab142:d40b5cf5:2c6293e1:
[27] 50220a48:93efb883:f6a270d7:7ebbf500:

E(P) 48:0a:22:50:83:b8:ef:93:d7:70:a2:f6:00:f5:bb:7e:

[ 0] 7ebbf500:02cab142:d40b5cf5:2c6293e1:
[ 1] 2c6293e1:b7506e40:9ff876e9:27680ebb:
[ 2] 27680ebb:e81bbd0a:8589a466:51756556:
[ 3] 51756556:b4cf22c7:9f458ab4:cc5ed5ac:
[ 4] cc5ed5ac:b887d560:5aec805d:e84284b1:
[ 5] e84284b1:a049fe1b:2ca7f049:bf39c848:
[ 6] bf39c848:37666a16:0be5e25c:77bc6cce:
[ 7] 77bc6cce:56d87487:d8293b6a:27463c85:
[ 8] 27463c85:e035f7b2:adee99f8:7e93071f:
[ 9] 7e93071f:43678857:32abde8d:dfb28815:
[10] dfb28815:1cd0fb27:384dab16:708a142d:
[11] 708a142d:47f14a14:59856142:16ca3ac6:
[12] 16ca3ac6:cde2d425:74a077bc:736a1dbe:
[13] 736a1dbe:119085f2:03887a25:34126c49:
```

```
[14] 34126c49:a0779dc6:c1ec967b:9fc8eef4:
[15] 9fc8eef4:abbba40a:82a9b7db:9b1597a9:
[16] 9b1597a9:60682b82:8b8ab769:665ddca7:
[17] 665ddca7:f3d4cb2a:95e77f04:f1b451ee:
[18] f1b451ee:ce697354:1ede422e:6451c2cc:
[19] 6451c2cc:a297f68a:caba2614:cb4de675:
[20] cb4de675:811f7fdc:d32a3b7d:f535da1b:
[21] f535da1b:b1f623bb:aa7b71a5:b990a1d5:
[22] b990a1d5:4b99deae:b0d6eaa1:dd51bbe7:
[23] dd51bbe7:21d32a12:20c32305:001eb619:
[24] 001eb619:654112dd:6c16a5e6:b4b1fab3:
[25] b4b1fab3:2e642b3b:25a59a57:f0d13dc3:
[26] f0d13dc3:2d146147:faeaabf8:771ce306:
[27] 771ce306:361dcf6e:10cc1cac:4ef75468:

D(E(P)) 06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
equal.

[seed for external encoding A]
eb:f9:b5:68:5a:c1:0d:6c:7c:b6:75:68:12:0e:12:3c:
02:67:83:54:bd:43:38:38:27:d0:d4:cf:d8:cc:f2:36:
== External encoding A for encryption ==

[0,0] f2 0 7 4 a 3 6 f b 9 5 d e 1 c 8
  finv 1 d 0 5 3 a 6 2 f 9 4 8 e b c 7
[0,1] fe d b 6 2 8 9 0 c 4 f 5 1 a 3 7
  finv 7 c 4 e 9 b 3 f 5 6 d 2 8 1 0 a
[0,2] fe a d 6 9 b 0 4 c 8 5 3 7 2 1 f
  finv 6 e d b 7 a 3 c 9 4 1 5 8 2 0 f
[0,3] f8 f 0 6 7 3 d 5 4 c b 2 a 9 e 1
  finv 2 f b 5 8 7 3 4 0 d c a 9 6 e 1
[0,4] f9 1 2 e d f 6 a 7 0 b 5 8 3 4 c
  finv 9 1 2 d e b 6 8 c 0 7 a f 4 3 5
[0,5] fd 5 9 3 4 c e 8 a b 2 6 f 7 0 1
  finv e f a 3 4 1 b d 7 2 8 9 5 0 6 c
[0,6] fe 3 0 4 9 b 8 c 1 f 7 d 2 5 a 6
  finv 2 8 c 1 3 d f a 6 4 e 5 7 b 0 9
[0,7] f7 f d 2 b 5 6 e 1 c 9 8 a 4 0 3
  finv e 8 3 f d 5 6 0 b a c 4 9 2 7 1
[1,0] f3 5 0 d b c 4 f 1 6 a 8 9 e 2 7
  finv 2 8 e 0 6 1 9 f b c a 4 5 3 d 7
[1,1] fd 3 0 6 a 8 7 4 c 5 2 1 9 b e f
  finv 2 b a 1 7 9 3 6 5 c 4 d 8 0 e f
[1,2] f4 2 e 1 0 5 7 8 b 3 6 9 c a f d
  finv 4 3 1 9 0 5 a 6 7 b d 8 c f 2 e
[1,3] f6 1 c 2 0 d 8 9 e 3 7 b a 4 5 f
  finv 4 1 3 9 d e 0 a 6 7 c b 2 5 8 f
[1,4] f1 2 5 0 b 6 c a 7 9 f 4 e d 3 8
```

```
  finv 3 0 1 e b 2 5 8 f 9 7 4 6 d c a
[1,5] f4 f a e 3 d 1 b 0 7 c 8 6 5 9 2
  finv 8 6 f 4 0 d c 9 b e 2 7 a 5 3 1
[1,6] f6 d 3 b 5 c 9 f e 0 1 7 a 8 2 4
  finv 9 a e 2 f 4 0 b d 6 c 3 5 1 8 7
[1,7] fa f 2 5 b e 1 4 c 8 d 9 0 7 3 6
  finv c 6 2 e 7 3 f d 9 b 0 4 8 a 5 1
[2,0] f1 6 7 d e c 5 0 f 2 4 8 3 9 b a
  finv 7 0 9 c a 6 1 2 b d f e 5 3 4 8
[2,1] f4 1 f 2 8 9 e 6 5 3 7 a 0 d b c
  finv c 1 3 9 0 8 7 a 4 5 b e f d 6 2
[2,2] f5 a 0 b d 8 2 9 c 3 1 7 f 6 e 4
  finv 2 a 6 9 f 0 d b 5 7 1 3 8 4 e c
[2,3] f3 7 e f 4 9 2 0 c 6 8 b d 5 a 1
  finv 7 f 6 0 4 d 9 1 a 5 e b 8 c 2 3
[2,4] f1 4 c f 2 5 9 b a 6 3 8 e 7 0 d
  finv e 0 4 a 1 5 9 d b 6 8 7 2 f c 3
[2,5] ff 7 c 0 e 2 3 8 a 9 b 1 4 6 5 d
  finv 3 b 5 6 c e d 1 7 9 8 a 2 f 4 0
[2,6] f2 5 c 0 9 4 7 3 1 a 8 b f 6 e d
  finv 3 8 0 7 5 1 d 6 a 4 9 b 2 f e c
[2,7] f7 0 1 d 4 f c b 9 8 5 3 2 6 a e
  finv 1 2 c b 4 a d 0 9 8 e 7 6 3 f 5
[3,0] f1 d b 4 3 2 f c 6 7 e 9 5 8 a 0
  finv f 0 5 4 3 c 8 9 d b e 2 7 1 a 6
[3,1] fc 8 b 2 0 6 7 9 f 4 1 d 3 a 5 e
  finv 4 a 3 c 9 e 5 6 1 7 d 2 0 b f 8
[3,2] f2 0 f 9 c 6 a b 7 d e 3 1 8 5 4
  finv 1 c 0 b f e 5 8 d 3 6 7 4 9 a 2
[3,3] f9 f 0 6 5 e 4 2 a d 7 8 b c 3 1
  finv 2 f 7 e 6 4 3 a b 0 8 c d 9 5 1
[3,4] fc 8 4 9 7 5 6 3 2 e 0 b d a 1 f
  finv a e 8 7 2 5 6 4 1 3 d b 0 c 9 f
[3,5] f0 8 6 c 1 f 2 7 b e 3 5 d 9 4 a
  finv 0 4 6 a e b 2 7 1 d f 8 3 c 9 5
[3,6] f3 1 a 8 d b 9 5 0 c 7 4 2 e 6 f
  finv 8 1 c 0 b 7 e a 3 6 2 5 9 4 d f
[3,7] f2 7 f 3 a 0 d c b 9 5 6 1 e 4 8
  finv 5 c 0 3 e a b 1 f 9 4 8 7 6 d 2

[seed for external encoding B]
96:d7:ec:5c:e3:1b:2a:ea:e4:5f:fa:61:e8:e7:08:4d:
fc:cc:0a:f9:6c:9a:25:d2:b4:88:1f:9d:a4:2f:ae:9f:
== External encoding B for encryption ==

[0,0] f5 c 9 3 7 1 2 6 f d a e b 4 8 0
  finv f 5 6 3 d 0 7 4 e 2 a c 1 9 b 8
```

```
[0,1] f6 5 c 2 9 7 4 e 1 b 8 0 f a d 3
  finv b 8 3 f 6 1 0 5 a 4 d 9 2 e 7 c
[0,2] fa 5 0 b 9 2 c d 3 f 4 6 e 1 8 7
  finv 2 d 5 8 a 1 b f e 4 0 3 6 7 c 9
[0,3] f5 2 6 b 0 9 3 8 1 d f 7 4 c a e
  finv 4 8 1 6 c 0 2 b 7 5 e 3 d 9 f a
[0,4] f0 d 5 1 e 4 7 9 6 8 2 f 3 c a b
  finv 0 3 a c 5 2 8 6 9 7 e f d 1 4 b
[0,5] f6 1 e 7 a 2 d 0 b 4 5 c 8 3 f 9
  finv 7 1 5 d 9 a 0 3 c f 4 8 b 6 2 e
[0,6] f7 c 0 3 5 2 a d 1 9 e b 8 4 6 f
  finv 2 8 5 3 d 4 e 0 c 9 6 b 1 7 a f
[0,7] f8 1 4 3 f d 0 a 2 9 c 6 5 e 7 b
  finv 6 1 8 3 2 c b e 0 9 7 f a 5 d 4
[1,0] f8 7 1 9 f e 0 4 5 a d 6 c b 2 3
  finv 6 2 e f 7 8 b 1 0 3 9 d c a 5 4
[1,1] f1 8 2 a 0 e b 9 3 4 5 7 c d f 6
  finv 4 0 2 8 9 a f b 1 7 3 6 c d 5 e
[1,2] f1 b 8 9 0 c 7 e a d 6 2 4 f 5 3
  finv 4 0 b f c e a 6 2 3 8 1 5 9 7 d
[1,3] f3 8 b 4 2 0 e a 5 d 7 f 9 c 6 1
  finv 5 f 4 0 3 8 e a 1 c 7 2 d 9 6 b
[1,4] fa 5 1 9 6 c e d f 7 3 0 2 4 8 b
  finv b 2 c a d 1 4 9 e 3 0 f 5 7 6 8
[1,5] f9 7 1 6 b a 8 d f 0 2 4 3 e 5 c
  finv 9 2 a c b e 3 1 6 0 5 4 f 7 d 8
[1,6] fd 9 2 e f 4 0 3 8 6 a 7 1 c b 5
  finv 6 c 2 7 5 f 9 b 8 1 a e d 0 3 4
[1,7] f8 e d 9 6 a b 1 5 f 4 2 c 3 7 0
  finv f 7 b d a 8 4 e 0 3 5 6 c 2 1 9
[2,0] fe 0 7 d 1 8 f c 4 a 3 2 6 b 5 9
  finv 1 4 b a 8 e c 2 5 f 9 d 7 3 0 6
[2,1] fc f 2 e 0 1 8 b 9 5 7 a d 3 6 4
  finv 4 5 2 d f 9 e a 6 8 b 7 0 c 3 1
[2,2] ff c 3 4 5 8 2 1 e 9 d 0 6 b 7 a
  finv b 7 6 2 3 4 c e 5 9 f d 1 a 8 0
[2,3] f5 0 c 3 f d 2 b 6 1 e a 7 8 9 4
  finv 1 9 6 3 f 0 8 c d e b 7 2 5 a 4
[2,4] f9 b 0 3 c a 6 7 e 2 f 5 4 8 d 1
  finv 2 f 9 3 c b 6 7 d 0 5 1 4 e 8 a
[2,5] f4 f 6 7 9 8 c 3 e d a 1 b 2 0 5
  finv e b d 7 0 f 2 3 5 4 a c 6 9 8 1
[2,6] f2 f 1 7 0 3 8 5 c 9 4 e a d b 6
  finv 4 2 0 5 a 7 f 3 6 9 c e 8 d b 1
[2,7] f9 6 5 c 7 1 2 8 b 3 a f d 4 0 e
  finv e 5 6 9 d 2 1 4 7 0 a 8 3 c f b
[3,0] f6 f 3 7 d 9 8 5 0 4 c a 2 e 1 b
```

```
  finv 8 e c 2 9 7 0 3 6 5 b f a 4 d 1
[3,1] f3 9 8 d 1 2 4 5 a b e 7 0 f 6 c
  finv c 4 5 0 6 7 e b 2 1 8 9 f 3 a d
[3,2] fd 2 8 6 4 e 0 b 1 7 5 f 3 c 9 a
  finv 6 8 1 c 4 a 3 9 2 e f 7 d 0 5 b
[3,3] f4 3 d 7 f 5 1 c b a 6 0 e 8 9 2
  finv b 6 f 1 0 5 a 3 d e 9 8 7 2 c 4
[3,4] f3 d 9 c 2 4 5 0 1 f 7 a b e 8 6
  finv 7 8 4 0 5 6 f a e 2 b c 3 1 d 9
[3,5] fd 2 a f 1 e c 6 b 5 9 0 4 3 7 8
  finv b 4 1 d c 9 7 e f a 2 8 6 0 5 3
[3,6] fb 1 2 5 f e 8 0 4 6 a 3 d c 9 7
  finv 7 1 2 b 8 3 9 f 6 e a 0 d c 5 4
[3,7] fa 5 f e 8 0 7 b c d 4 6 3 1 9 2
  finv 5 d f c a 1 b 6 4 e 0 7 8 9 3 2

[seed for encryption encoding]
5c:17:92:a4:82:67:a9:d6:1c:1d:33:1e:17:25:97:e8:
c2:52:77:a4:94:0b:09:94:8e:50:1b:c5:70:80:b6:42:

[Whitebox WF-LEA192 encryption]
P 06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
A(P) e6:e6:58:ec:72:ad:fd:59:73:7f:4e:02:76:ec:a3:a6:

[ 0] 6dcdbe35:c1af5edf:cc5dd261:ec58e6e6:
[ 1] d19b3f56:8ea07e17:32bc6629:6dcdbe35:
[ 2] 558adb2b:82496774:cadb4caa:d19b3f56:
[ 3] 01e42a8a:2977f373:6b4e5f4c:558adb2b:
[ 4] 14c2a7d3:6da990fa:c5274b6c:01e42a8a:
[ 5] a2c272ee:4e04f86a:62eac141:14c2a7d3:
[ 6] f7009012:f0eda0c5:1838bf1f:a2c272ee:
[ 7] 4ac88166:de387a39:6d6a06e4:f7009012:
[ 8] fc14cc3e:c531dd07:bc25fa63:4ac88166:
[ 9] fc716d83:fa004530:e9ba2e1b:fc14cc3e:
[10] a27e2b41:bcb3cdb3:09a2f51d:fc716d83:
[11] 640dc739:a146aaa7:2755cba6:a27e2b41:
[12] 6a2b5081:5fb4a0df:fc91d355:640dc739:
[13] bc98e6b7:39bef6b2:7e6bd8d7:6a2b5081:
[14] 94c97c6d:e786bade:ed0f6dfe:bc98e6b7:
[15] 2ba1376d:3565266d:7b135589:94c97c6d:
[16] eb054828:278f12d6:022ebdef:2ba1376d:
[17] a79e8602:87a4f6cf:0d8f573d:eb054828:
[18] 98c4d49e:38883638:2cac71ad:a79e8602:
[19] 019cfceb:4af6e989:16f3832d:98c4d49e:
[20] 8393dbdb:ee199d5b:e7aa34e2:019cfceb:
[21] 350c3fc0:18cc1b89:e4f047af:8393dbdb:
[22] eeb23a16:7daa6fe8:571ab8c4:350c3fc0:
[23] 18f5276f:040e5d6b:62e1c6db:eeb23a16:
```

[24] f38e7588:2f80a381:69744b2c:18f5276f:
[25] 485b5595:6723dc79:66a061db:f38e7588:
[26] 658c4ac8:7038122e:6494b039:485b5595:
[27] c25a406e:37d8221f:bfa9cbc2:658c4ac8:

Ewb(A(P))6e:40:5a:c2:1f:22:d8:37:c2:cb:a9:bf:c8:4a:8c:65:
B(Ewb(A(P))) 48:0a:22:50:83:b8:ef:93:d7:70:a2:f6:00:f5:bb:7e:
equal.

[seed for external encoding A]
47:e1:61:a0:df:dc:85:06:36:25:35:d0:94:9b:1a:06:
4e:cd:18:4e:9b:11:31:0e:76:1e:d6:18:d2:52:0e:51:
== External encoding A for decryption ==

[0,0] fd 1 7 b c f 5 8 2 0 e 9 3 6 a 4
  finv 9 1 8 c f 6 d 2 7 b e 3 4 0 a 5
[0,1] fa 0 e 2 5 d 4 7 1 8 c f 6 3 b 9
  finv 1 8 3 d 6 4 c 7 9 f 0 e a 5 2 b
[0,2] f6 f 8 e 4 9 2 5 7 b c 1 a 0 3 d
  finv d b 6 e 4 7 0 8 2 5 c 9 a f 3 1
[0,3] f3 2 d 4 7 b c e 0 a 9 6 f 5 8 1
  finv 8 f 1 0 3 d b 4 e a 9 5 6 2 7 c
[0,4] ff b 8 2 1 4 9 5 0 3 a 6 7 c e d
  finv 8 4 3 9 5 7 b c 2 6 a 1 d f e 0
[0,5] fd 0 5 c a 2 7 8 3 9 6 b f 1 e 4
  finv 1 d 5 8 f 2 a 6 7 9 4 b 3 0 e c
[0,6] fb a d 0 3 c 5 9 6 1 e 4 f 2 7 8
  finv 3 9 d 4 b 6 8 e f 7 1 0 5 2 a c
[0,7] fc 1 5 b 8 f 6 3 4 2 d 9 7 a 0 e
  finv e 1 9 7 8 2 6 c 4 b d 3 0 a f 5
[1,0] f3 e f 1 2 8 b a 9 4 5 d c 6 7 0
  finv f 3 4 0 9 a d e 5 8 7 6 c b 1 2
[1,1] ff 6 a 9 8 1 c b 7 2 e 0 d 4 5 3
  finv b 5 9 f d e 1 8 4 3 2 7 6 c a 0
[1,2] fc 1 5 8 e 3 a 9 0 7 6 b 4 f d 2
  finv 8 1 f 5 c 2 a 9 3 7 6 b 0 e 4 d
[1,3] f9 2 0 5 c 6 e 7 d 4 f 1 a 3 8 b
  finv 2 b 1 d 9 3 5 7 e 0 c f 4 8 6 a
[1,4] f0 c 1 5 b 6 4 8 e a f 9 3 7 d 2
  finv 0 2 f c 6 3 5 d 7 b 9 4 1 e 8 a
[1,5] f7 4 6 9 2 3 0 1 5 b e c 8 f a d
  finv 6 7 4 5 1 8 2 0 c 3 e 9 b f a d
[1,6] fd e 9 4 6 0 b 7 1 f 5 c a 3 8 2
  finv 5 8 f d 3 a 4 7 e 2 c 6 b 0 1 9
[1,7] fe b d 2 a 1 5 c 6 4 8 f 9 7 0 3
  finv e 5 3 f 9 6 8 d a c 4 1 7 2 0 b
[2,0] fd c f 0 5 3 a b 2 e 8 6 7 9 4 1
  finv 3 f 8 5 e 4 b c a d 6 7 1 0 9 2

```
[2,1] f2 c b 3 4 7 e f 9 8 0 6 d a 5 1
  finv a f 0 3 4 e b 5 9 8 d 2 1 c 6 7
[2,2] fd e 4 6 3 1 a f c 9 0 7 5 8 2 b
  finv a 5 e 4 2 c 3 b d 9 6 f 8 0 1 7
[2,3] f5 b 8 3 0 a d 9 2 1 4 f e c 6 7
  finv 4 9 8 3 a 0 e f 2 7 5 1 d 6 c b
[2,4] f1 b e 7 4 c 2 0 5 f 9 8 d a 3 6
  finv 7 0 6 e 4 8 f 3 b a d 1 5 c 2 9
[2,5] f4 6 5 7 f 0 9 1 a e b 2 c 8 d 3
  finv 5 7 b f 0 2 1 3 d 6 8 a c e 9 4
[2,6] f8 5 0 6 d 2 f e 4 b 1 7 3 c a 9
  finv 2 a 5 c 8 1 3 b 0 f e 9 d 4 7 6
[2,7] fd 0 e 6 c 5 9 7 f 3 1 2 8 a 4 b
  finv 1 a b 9 e 5 3 7 c 6 d f 4 0 2 8
[3,0] fe d 5 4 c 9 7 2 0 3 6 b 8 a f 1
  finv 8 f 7 9 3 2 a 6 c 5 d b 4 1 0 e
[3,1] fe 9 a 1 3 6 7 4 0 f 8 5 b c 2 d
  finv 8 3 e 4 7 b 5 6 a 1 2 c d f 0 9
[3,2] fe 2 c 3 a 6 f 7 4 b 5 9 1 8 0 d
  finv e c 1 3 8 a 5 7 d b 4 9 2 f 0 6
[3,3] fa 4 d 5 6 c e 2 1 0 f 3 7 9 b 8
  finv 9 8 7 b 1 3 4 c f d 0 e 5 2 6 a
[3,4] f7 3 a d 4 9 0 b 5 f e 8 2 6 1 c
  finv 6 e c 1 4 8 d 0 b 5 2 7 f 3 a 9
[3,5] f3 4 d 9 7 1 8 5 b a e f 6 0 2 c
  finv d 5 e 0 1 7 c 4 6 3 9 8 f 2 a b
[3,6] f9 b 1 2 3 a 5 d c 8 4 0 7 f e 6
  finv b 2 3 4 a 6 f c 9 0 5 1 8 7 e d
[3,7] f2 0 6 3 9 7 e c 8 1 d f 5 b 4 a
  finv 1 9 0 3 e c 2 5 8 4 f d 7 a 6 b

[seed for external encoding B]
52:c9:04:27:cc:41:50:da:ca:24:45:5c:61:62:61:08:
f8:23:e9:f0:2b:8f:ae:4c:68:98:2d:80:da:4e:21:4f:
== External encoding B for decryption ==

[0,0] f5 9 8 4 3 c e 0 6 7 1 2 a f b d
  finv 7 a b 4 3 0 8 9 2 1 c e 5 f 6 d
[0,1] ff a c 7 d 5 b 8 3 e 6 0 2 9 4 1
  finv b f c 8 e 5 a 3 7 d 1 6 2 4 9 0
[0,2] f9 f 5 c 8 d 4 1 a 7 b 6 0 3 2 e
  finv c 7 e d 6 2 b 9 4 0 8 a 3 5 f 1
[0,3] f7 1 5 e f 9 b d a 8 6 c 2 0 4 3
  finv d 1 c f e 2 a 0 9 5 8 6 b 7 3 4
[0,4] f0 f 2 4 b c 7 6 a 9 d 1 e 8 3 5
  finv 0 b 2 e 3 f 7 6 d 9 8 4 5 a c 1
[0,5] f9 6 d 7 8 2 0 5 3 4 b a c 1 e f
```

```
  finv 6 d 5 8 9 7 1 3 4 0 b a c 2 e f
[0,6] fa f 4 b e 0 2 6 1 7 9 d 3 c 8 5
  finv 5 8 6 c 2 f 7 9 e a 0 3 d b 4 1
[0,7] f7 3 a 6 d 0 f 1 8 2 4 e b 9 c 5
  finv 5 7 9 1 a f 3 0 8 d 2 c e 4 b 6
[1,0] f0 b e a 6 3 1 7 9 d 4 8 5 c f 2
  finv 0 6 f 5 a c 4 7 b 8 3 1 d 9 2 e
[1,1] f2 1 0 d 5 3 9 6 e f c 4 7 b a 8
  finv 2 1 0 5 b 4 7 c f 6 e d a 3 8 9
[1,2] fb 4 5 3 9 1 d a c 2 8 7 0 6 f e
  finv c 5 9 3 1 2 d b a 4 7 0 8 6 f e
[1,3] f5 d b 6 f 1 9 c a 7 0 e 2 8 4 3
  finv a 5 c f e 0 3 9 d 6 8 2 7 1 b 4
[1,4] f4 0 8 6 5 c 9 d b 3 2 a 7 1 f e
  finv 1 d a 9 0 4 3 c 2 6 b 8 5 7 f e
[1,5] f3 1 c e d 6 f b 2 5 7 8 4 a 9 0
  finv f 1 8 0 c 9 5 a b e d 7 2 4 3 6
[1,6] ff 0 1 9 2 6 3 8 5 c a e b 4 7 d
  finv 1 2 4 6 d 8 5 e 7 3 a c 9 f b 0
[1,7] f0 5 1 e 6 3 b d 8 4 7 c 2 f a 9
  finv 0 2 c 5 9 1 4 a 8 f e 6 b 7 3 d
[2,0] f6 5 7 f b 2 e 0 c a 4 3 9 8 d 1
  finv 7 f 5 b a 1 0 2 d c 9 4 8 e 6 3
[2,1] ff 9 6 2 1 a 8 c 4 7 b e d 3 0 5
  finv e 4 3 d 8 f 2 9 6 1 5 a 7 c b 0
[2,2] fa 6 1 8 2 3 4 7 5 c b d f 0 9 e
  finv d 2 4 5 6 8 1 7 3 e 0 a 9 b f c
[2,3] f9 3 6 f b a 4 0 1 5 d 7 c 2 e 8
  finv 7 8 d 1 6 9 2 b f 0 5 4 c a e 3
[2,4] f4 b 6 8 f 2 7 c 9 3 5 d 0 a e 1
  finv c f 5 9 0 a 2 6 3 8 d 1 7 b e 4
[2,5] fb 9 e 1 d 3 8 4 a 2 5 7 c 0 f 6
  finv d 3 9 5 7 a f b 6 1 8 0 c 4 2 e
[2,6] f0 6 7 b 2 1 5 8 e a d 9 4 3 c f
  finv 0 5 4 d c 6 1 2 7 b 9 3 e a 8 f
[2,7] f0 5 b 1 6 4 9 c 3 e d f 2 8 a 7
  finv 0 3 c 8 5 1 4 f d 6 e 2 7 a 9 b
[3,0] ff a 1 0 e c 8 5 2 3 7 4 6 d 9 b
  finv 3 2 8 9 b 7 c a 6 e 1 f 5 d 4 0
[3,1] f1 0 7 b 2 f e 5 8 4 9 6 3 c a d
  finv 1 0 4 c 9 7 b 2 8 a e 3 d f 6 5
[3,2] f9 6 f e d 8 4 3 b 7 2 1 5 c a 0
  finv f b a 7 6 c 1 9 5 0 e 8 d 4 3 2
[3,3] f0 2 a 9 b 3 7 f d c 1 5 e 4 8 6
  finv 0 a 1 5 d b f 6 e 3 2 4 9 8 c 7
[3,4] f9 f a 1 2 d 5 7 4 8 b e 6 c 0 3
  finv e 3 4 f 8 6 c 7 9 0 2 a d 5 b 1
```

```
[3,5] fb 1 5 0 2 e 6 c a 4 3 f 8 7 9 d
  finv 3 1 4 a 9 2 6 d c e 8 0 7 f 5 b
[3,6] f9 a 2 5 8 7 0 4 b d f e 3 6 1 c
  finv 6 e 2 c 7 3 d 5 4 0 1 8 f 9 b a
[3,7] f6 0 f 3 a 1 e 5 d c 4 2 b 9 7 8
  finv 1 5 b 3 a 7 0 e f d 4 c 9 8 6 2

[seed for decryption encoding]
92:ac:79:7b:22:7d:33:34:b9:21:f2:bf:5b:1b:93:3f:
a1:c8:f1:17:f5:e5:4f:24:f7:77:25:2a:35:fb:c9:77:

[Whitebox WF-LEA192 decryption]
C 48:0a:22:50:83:b8:ef:93:d7:70:a2:f6:00:f5:bb:7e:
A(C) 52:3c:58:fb:71:10:a2:44:ab:9d:be:bf:ee:86:f8:ce:

[ 0] cef886ee:2f123468:41a27aeb:f2ef1e0e:
[ 1] f2ef1e0e:546e7b3a:39de2546:35a5369b:
[ 2] 35a5369b:bcf8cf2d:263380bb:a85244aa:
[ 3] a85244aa:195b6411:05008543:fe96376a:
[ 4] fe96376a:e513384e:620b3a8c:9c03d6c6:
[ 5] 9c03d6c6:ccd3f13a:bdd2e450:9f8f0a5b:
[ 6] 9f8f0a5b:22805652:5c2c1302:805d6da7:
[ 7] 805d6da7:b8edd36e:f6a25840:dc65cace:
[ 8] dc65cace:34916d75:88812055:feb9ee1a:
[ 9] feb9ee1a:b1f2dce7:af2e4e19:ee430f68:
[10] ee430f68:f220a006:206fff13:7c23a07a:
[11] 7c23a07a:f769fc55:482607c2:e1be2c26:
[12] e1be2c26:fb214fff:08fe04eb:58b25475:
[13] 58b25475:7752c683:0931e2c8:df81b7a1:
[14] df81b7a1:7fa0af80:a246d206:fc4849fb:
[15] fc4849fb:078f6c50:3da931f7:32d5cc82:
[16] 32d5cc82:094b9603:a37758aa:742a7936:
[17] 742a7936:271d82e9:dc61d682:d8222215:
[18] d8222215:09a72dc8:43581fc2:a56416f9:
[19] a56416f9:f089c42f:fc4b10df:13a6c6f1:
[20] 13a6c6f1:ceece1c8:f949b7d4:3521ae3d:
[21] 3521ae3d:028f76ba:44216b58:3bb7363d:
[22] 3bb7363d:d9c81e91:ce8952d2:3e103a4a:
[23] 3e103a4a:8fd681dd:9905bdc4:4839dc27:
[24] 4839dc27:52a7beea:13d1a274:e2caafc5:
[25] e2caafc5:f748ba8c:9665ea68:b6169440:
[26] b6169440:3a8dc846:2d521a49:09d53db8:
[27] 09d53db8:55177e72:30c78958:abb7b6b6:

Dwb(A(C))b8:3d:d5:09:72:7e:17:55:58:89:c7:30:b6:b6:b7:ab:
B(Dwb(A(C))) 06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
equal.
```

1.3 Whitebox WF-LEA256

```
[test: Whitebox WF-LEA256 encryption]
block bit length = 128
key bit length = 256

[key]
9c:c4:ea:59:4b:85:f8:15:7e:e0:d1:82:e9:9d:4a:1b:
32:36:c1:48:27:82:89:dd:5a:92:2e:50:64:78:9a:e7:

[roundkeys]
[00] c21c7cfb:2e64c4b6:483804a0:4034c0f6:619c5a4f:c623046f:
[01] 281f4bbe:57ea2472:35b2873a:676b8ec0:838af928:993c76a1:
[02] 36fd0f13:352cb17c:0f4c3817:f4c1304a:fc5c183c:ddc991e1:
[03] 24dd1521:01facec8:758cde51:b47575a6:025af562:46ef835a:
[04] ff3cf424:507fb369:65e0612f:0a6fa261:2bfc8919:58e4a9ff:
[05] 761292a4:faca4777:33533f44:85e3f539:6d773cb3:e14efbeb:
[06] 0d325676:4e31a4bb:b2faa44d:8fcf2547:133d9836:e8ed54a6:
[07] e40a6eb4:c56740ce:ab9750b6:84fe473c:cca49f29:f4c76f65:
[08] d0bbb1c1:aab1245e:78fa9504:55d0c54f:ba923b15:3502bc28:
[09] 7135e794:1723a18f:762f8b95:f7925412:e5ce5147:a6fc32ea:
[10] fc68fe88:e718eda9:2ec6a564:fd0e7642:f5c0299c:9d336d42:
[11] 9936f47a:b5c98c63:5e9ceb95:3ab6e4dc:d81317d5:4c04d455:
[12] c66b5e31:092428ce:6b0c6919:25957890:e03ee1cc:9af804c4:
[13] 0a5dacfc:0af1b69c:db32034f:b7337f75:1fd4f4a6:f87c29fd:
[14] 361df58d:37427a8b:75280b5b:cdc4ea97:b40e00b7:2bb060a1:
[15] ac12daa8:c88fe375:b51d72c1:d38d63d9:fb4343c3:38fa4517:
[16] 88e7c91b:3a9dcefd:e916aff7:d5a509fd:45d4195c:8065100b:
[17] 9b69be43:26697a7b:c21b3b12:452911c8:1eb3f784:009b8dde:
[18] ff871ff5:15c9cb42:42d45dbf:26fc5d02:abc38e30:df26021f:
[19] f2d93fd4:920f5f97:de443b6c:5d2813a5:8299e345:3091bf78:
[20] 1a55d12e:cf376cd9:592d08a0:314ebe4b:982c8c8a:30d66a04:
[21] 4bc57f8e:cca1131a:708a8e58:4daddf3d:c353e8ba:a0513003:
[22] 3b6f1785:dc8ad653:c4868382:97fbea27:28b07676:5fc08061:
[23] 5bf0cf8b:5168940b:db723f0e:7235a34a:7206224d:d993135c:
[24] b617994e:aaeea5c0:ab545b6d:9c6dd494:0d1a7fc6:f5c3066f:
[25] db2e2605:11723e6d:dd8717df:6432c776:0698d067:6a88f02b:
[26] bec95a90:3163a826:04e87b28:1b4cebe6:7e2591fc:e27e8b51:
[27] b1b1728c:ab31b889:1883fabb:566b21a5:1a773075:f0feb279:
[28] 4c64c8c5:92ed7107:b65b2711:f9eae5a3:d8d40480:d04a0635:
[29] 9331f447:0aabd340:df4f824d:096662d7:3ec97cc9:f38a2755:
[30] f5968590:6c778371:4e6d3a9c:d69294ca:9451600c:e1232baa:
[31] 903678dd:b30d7ad9:8542b343:95812cc3:38fd089e:f095ac01:

[WF-LEA256 encryption]
P06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:

[ 0] f357ab9a:4678f541:5f4492dd:771ce306:
[ 1] b762afd9:c55fcc74:995de033:f357ab9a:
```

```
[ 2] 263ba4e4:e1bd8626:52740651:b762afd9:
[ 3] 5bf567c6:73d19663:c83783f6:263ba4e4:
[ 4] ef71d990:1ec450c7:4c554301:5bf567c6:
[ 5] e2c5c8fa:dfba712d:fb7bc37b:ef71d990:
[ 6] 06e84503:e70faddc:7dfc5d10:e2c5c8fa:
[ 7] 9631920a:b22cd8bc:18eb6d3b:06e84503:
[ 8] 50415abe:60c08fb1:3acc89eb:9631920a:
[ 9] afd6d132:ef226f10:81fa0f31:50415abe:
[10] f164e6b7:39f6ca1f:28358bd5:afd6d132:
[11] 22b293e9:9bcf6c84:fa7f142c:f164e6b7:
[12] 8a2444ef:ce856b92:70bbdb0a:22b293e9:
[13] dd8a4289:e6ea006a:0947bd38:8a2444ef:
[14] 8063cb7a:02c22b17:abdbbc3b:dd8a4289:
[15] 7db469ed:c181b1cd:c6c120f2:8063cb7a:
[16] e0404de0:49dfda3a:f06382a3:7db469ed:
[17] c127c9d5:9a087ba4:4d800b2b:e0404de0:
[18] c50e0d9c:2222c3e2:44b53aa3:c127c9d5:
[19] 099d7bd0:a0b0210c:76fc6a12:c50e0d9c:
[20] 9ff1a706:2a0a7ff0:1c9529c6:099d7bd0:
[21] c08ae575:1d5dcf45:f13261a9:9ff1a706:
[22] 7a180d7b:aa0526c2:d33667e8:c08ae575:
[23] aceb723a:7093d6f3:d7494779:7a180d7b:
[24] f4bd4fea:5937610c:6d45c87a:aceb723a:
[25] b192a0ef:fc713c2e:c6481345:f4bd4fea:
[26] dd1d0fb8:4eacf1fd:99e628ee:b192a0ef:
[27] 938d50a4:892de0a4:389fa566:dd1d0fb8:
[28] 545409f7:d0076043:7db4556e:938d50a4:
[29] 25616744:3c20d8ce:14709433:545409f7:
[30] 9e7d2642:59a97f1a:86b322d3:25616744:
[31] dec8c5f1:4f80eed3:52885eb2:9e7d2642:

E(P) f1:c5:c8:de:d3:ee:80:4f:b2:5e:88:52:42:26:7d:9e:

[ 0] 9e7d2642:59a97f1a:86b322d3:25616744:
[ 1] 25616744:3c20d8ce:14709433:545409f7:
[ 2] 545409f7:d0076043:7db4556e:938d50a4:
[ 3] 938d50a4:892de0a4:389fa566:dd1d0fb8:
[ 4] dd1d0fb8:4eacf1fd:99e628ee:b192a0ef:
[ 5] b192a0ef:fc713c2e:c6481345:f4bd4fea:
[ 6] f4bd4fea:5937610c:6d45c87a:aceb723a:
[ 7] aceb723a:7093d6f3:d7494779:7a180d7b:
[ 8] 7a180d7b:aa0526c2:d33667e8:c08ae575:
[ 9] c08ae575:1d5dcf45:f13261a9:9ff1a706:
[10] 9ff1a706:2a0a7ff0:1c9529c6:099d7bd0:
[11] 099d7bd0:a0b0210c:76fc6a12:c50e0d9c:
[12] c50e0d9c:2222c3e2:44b53aa3:c127c9d5:
[13] c127c9d5:9a087ba4:4d800b2b:e0404de0:
```

```
[14] e0404de0:49dfda3a:f06382a3:7db469ed:
[15] 7db469ed:c181b1cd:c6c120f2:8063cb7a:
[16] 8063cb7a:02c22b17:abdbbc3b:dd8a4289:
[17] dd8a4289:e6ea006a:0947bd38:8a2444ef:
[18] 8a2444ef:ce856b92:70bbdb0a:22b293e9:
[19] 22b293e9:9bcf6c84:fa7f142c:f164e6b7:
[20] f164e6b7:39f6ca1f:28358bd5:afd6d132:
[21] afd6d132:ef226f10:81fa0f31:50415abe:
[22] 50415abe:60c08fb1:3acc89eb:9631920a:
[23] 9631920a:b22cd8bc:18eb6d3b:06e84503:
[24] 06e84503:e70faddc:7dfc5d10:e2c5c8fa:
[25] e2c5c8fa:dfba712d:fb7bc37b:ef71d990:
[26] ef71d990:1ec450c7:4c554301:5bf567c6:
[27] 5bf567c6:73d19663:c83783f6:263ba4e4:
[28] 263ba4e4:e1bd8626:52740651:b762afd9:
[29] b762afd9:c55fcc74:995de033:f357ab9a:
[30] f357ab9a:4678f541:5f4492dd:771ce306:
[31] 771ce306:361dcf6e:10cc1cac:4ef75468:

D(E(P))  06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
equal.

[seed for external encoding A]
ca:73:5d:e6:f1:cb:a5:60:dd:74:d1:69:06:00:5d:c0:
d7:00:df:48:68:e2:d2:6d:46:72:98:5b:a4:11:5b:5f:
== External encoding A for encryption ==

[0,0] f6 c b 1 0 4 3 9 e a 7 5 d f 8 2
  finv 4 3 f 6 5 b 0 a e 7 9 2 1 c 8 d
[0,1] f6 f 0 b 8 c e 5 d 9 3 4 a 2 7 1
  finv 2 f d a b 7 0 e 4 9 c 3 5 8 6 1
[0,2] f5 1 4 8 7 a c 0 f b d 2 9 3 6 e
  finv 7 1 b d 2 0 e 4 3 c 5 9 6 a f 8
[0,3] fe 7 6 c 4 1 5 a 0 d 8 2 9 f b 3
  finv 8 5 b f 4 6 2 1 a c 7 e 3 9 0 d
[0,4] f0 7 f 6 5 4 c 3 1 b 8 a d e 9 2
  finv 0 8 f 7 5 4 3 1 a e b 9 6 c d 2
[0,5] ff c e 8 5 3 4 b 9 6 a d 2 0 1 7
  finv d e c 5 6 4 9 f 3 8 a 7 1 b 2 0
[0,6] fa 8 1 6 5 d 0 9 3 4 f 7 c e 2 b
  finv 6 2 e 8 9 4 3 b 1 7 0 f c 5 d a
[0,7] f1 b a d 9 f e 6 2 8 4 0 c 3 7 5
  finv b 0 8 d a f 7 e 9 4 2 1 c 3 6 5
[1,0] f1 8 2 3 f c a d 6 9 4 0 e 7 5 b
  finv b 0 2 3 a e 8 d 1 9 6 f 5 7 c 4
[1,1] fa e f 2 3 b 0 4 d 7 6 8 1 c 9 5
  finv 6 c 3 4 7 f a 9 b e 0 5 d 8 1 2
[1,2] fb 3 9 0 7 5 e 8 f c d 1 4 6 2 a
```

```
  finv 3 b e 1 c 5 d 4 7 2 f 0 9 a 6 8
[1,3] fb a 6 5 4 7 1 3 8 9 d f e c 0 2
  finv e 6 f 7 4 3 2 5 8 9 1 0 d a c b
[1,4] f6 5 e 9 4 d f 1 0 a 7 3 b c 8 2
  finv 8 7 f b 4 1 0 a e 3 9 c d 5 2 6
[1,5] fd 4 c a f 9 e 0 5 3 6 b 7 2 8 1
  finv 7 f d 9 1 8 a c e 5 3 b 2 0 6 4
[1,6] f8 5 3 9 f e 1 6 a b 7 0 4 c 2 d
  finv b 6 e 2 c 1 7 a 0 3 8 9 d f 5 4
[1,7] f6 7 3 0 b e c 2 d 5 a f 9 8 4 1
  finv 3 f 7 2 e 9 0 1 d c a 4 6 8 5 b
[2,0] fc 1 b 9 0 7 6 8 d 2 e f 3 5 4 a
  finv 4 1 9 c e d 6 5 7 3 f 2 0 8 a b
[2,1] f7 2 8 b 9 c a 0 4 1 3 f 5 d 6 e
  finv 7 9 1 a 8 c e 0 2 4 6 3 5 d f b
[2,2] f1 b d c 6 5 a 8 4 3 2 9 e f 7 0
  finv f 0 a 9 8 5 4 e 7 b 6 1 3 2 c d
[2,3] f5 0 4 9 f b 1 d 6 a e 3 c 7 2 8
  finv 1 6 e b 2 0 8 d f 3 9 5 c 7 a 4
[2,4] ff 9 0 4 5 1 d e 7 3 c 8 b a 6 2
  finv 2 5 f 9 3 4 e 8 b 1 d c a 6 7 0
[2,5] f6 a 8 7 1 d 4 0 e f 5 3 c b 9 2
  finv 7 4 f b 6 a 0 3 2 e 1 d c 5 8 9
[2,6] f5 3 c 4 e 0 d 1 9 7 6 f 8 2 a b
  finv 5 7 d 1 3 0 a 9 c 8 e f 2 6 4 b
[2,7] f0 c 3 e 4 1 8 d 9 a b 2 6 f 5 7
  finv 0 5 b 2 4 e c f 6 8 9 a 1 7 3 d
[3,0] f7 6 c 2 5 4 e 9 0 8 d a b f 1 3
  finv 8 e 3 f 5 4 1 0 9 7 b c 2 a 6 d
[3,1] f0 a 7 3 f 8 6 2 1 9 e d 5 4 b c
  finv 0 8 7 3 d c 6 2 5 9 1 e f b a 4
[3,2] f1 3 7 9 4 5 b a f 2 0 6 c d 8 e
  finv a 0 9 1 4 5 b 2 e 3 7 6 c d f 8
[3,3] fe 0 2 5 d a f 7 3 8 9 1 b 4 c 6
  finv 1 b 2 8 d 3 f 7 9 a 5 c e 4 0 6
[3,4] fa f 0 5 9 7 1 8 4 6 3 e c 2 b d
  finv 2 6 d a 8 3 9 5 7 4 0 e c f b 1
[3,5] f1 0 c 7 3 f a 8 d 6 9 4 2 e b 5
  finv 1 0 c 4 b f 9 3 7 a 6 e 2 8 d 5
[3,6] fd 0 5 a c 3 9 f 8 2 e 4 7 b 6 1
  finv 1 f 9 5 b 2 e c 8 6 3 d 4 0 a 7
[3,7] f5 6 1 b c 0 3 d 7 4 8 f e 2 a 9
  finv 5 2 d 6 9 0 1 8 a f e 3 4 7 c b

[seed for external encoding B]
20:a3:65:a8:74:97:6e:fb:3d:18:59:70:43:51:79:f8:
```

```
10:ad:12:86:12:15:dc:85:34:e4:51:3c:26:4d:bf:2a:
== External encoding B for encryption ==

[0,0] fe 6 a d 1 f 4 c 3 0 9 7 2 5 8 b
  finv 9 4 c 8 6 d 1 b e a 2 f 7 3 0 5
[0,1] f3 0 e 5 c 1 2 6 7 9 8 4 f d a b
  finv 1 5 6 0 b 3 7 8 a 9 e f 4 d 2 c
[0,2] fa 9 b 8 3 e 6 7 1 4 c f 0 5 2 d
  finv c 8 e 4 9 d 6 7 3 1 0 2 a f 5 b
[0,3] f8 9 6 4 5 d 0 b 3 a 2 c 1 7 f e
  finv 6 c a 8 3 4 2 d 0 1 9 7 b 5 f e
[0,4] fd 7 1 b 4 0 2 9 8 6 c a e 5 3 f
  finv 5 2 6 e 4 d 9 1 8 7 b 3 a 0 c f
[0,5] fd 8 6 7 4 3 c a e 9 f 1 5 b 0 2
  finv e b f 5 4 c 2 3 1 9 7 d 6 0 8 a
[0,6] f5 a d 1 9 6 3 4 7 e b c 2 0 8 f
  finv d 3 c 6 7 0 5 8 e 4 1 a b 2 9 f
[0,7] f6 4 e 3 9 8 f b 1 5 d 0 7 2 c a
  finv b 8 d 3 1 9 0 c 5 4 f 7 e a 2 6
[1,0] f5 2 3 7 0 8 c 4 6 d 1 f a 9 b e
  finv 4 a 1 2 7 0 8 3 5 d c e 6 9 f b
[1,1] f2 5 8 1 d a 4 e f 9 6 c 7 3 0 b
  finv e 3 0 d 6 1 a c 2 9 5 f b 4 7 8
[1,2] f2 e b 5 c 1 3 7 a f d 4 6 0 8 9
  finv d 5 0 6 b 3 c 7 e f 8 2 4 a 1 9
[1,3] f1 f c 7 5 b 0 a 9 6 8 3 2 e 4 d
  finv 6 0 c b e 4 9 3 a 8 7 5 2 f d 1
[1,4] f9 6 8 d 2 f b 7 0 3 1 e c 5 4 a
  finv 8 a 4 9 e d 1 7 2 0 f 6 c 3 b 5
[1,5] fc a d 0 2 f e 8 4 9 1 3 b 7 5 6
  finv 3 a 4 b 8 e f d 7 9 1 c 0 2 6 5
[1,6] f8 c 6 b 4 e a f 0 7 2 9 d 3 1 5
  finv 8 e a d 4 f 2 9 0 b 6 3 1 c 5 7
[1,7] fa 8 9 b 5 1 f c d e 2 4 0 6 3 7
  finv c 5 a e b 4 d f 1 2 0 3 7 8 9 6
[2,0] f1 a e 0 4 c d b 7 f 9 5 6 2 8 3
  finv 3 0 d f 4 b c 8 e a 1 7 5 6 2 9
[2,1] f6 5 a 1 4 b c 7 f 2 e 3 d 8 9 0
  finv f 3 9 b 4 1 0 7 d e 2 5 6 c a 8
[2,2] f0 1 d 4 2 5 3 7 9 e b 6 a c 8 f
  finv 0 1 4 6 3 5 b 7 e 8 c a d 2 9 f
[2,3] f1 0 e 6 5 4 b f c 3 2 9 a 8 7 d
  finv 1 0 a 9 5 4 3 e d b c 6 8 f 2 7
[2,4] f7 1 e 3 4 b a 2 f 9 6 0 8 c d 5
  finv b 1 7 3 4 f a 0 c 9 6 5 d e 2 8
[2,5] ff d 4 8 c 1 6 5 3 b 2 e 0 7 a 9
  finv c 5 a 8 2 7 6 d 3 f e 9 4 1 b 0
```

```
[2,6] fc 5 b 7 0 e 8 2 4 1 a 6 3 f d 9
  finv 4 9 7 c 8 1 b 3 6 f a 2 0 e 5 d
[2,7] f8 f 1 e 6 5 a 3 d 4 7 2 9 0 c b
  finv d 2 b 7 9 5 4 a 0 c 6 f e 8 3 1
[3,0] f7 b 0 9 1 c 5 2 e a 3 d 6 f 8 4
  finv 2 4 7 a f 6 c 0 e 3 9 1 5 b 8 d
[3,1] f4 0 5 e d f 7 8 2 9 c 3 6 b 1 a
  finv 1 e 8 b 0 2 c 6 7 9 f d a 4 3 5
[3,2] fc f 3 e 4 9 0 6 d a 7 1 2 5 b 8
  finv 6 b c 2 4 d 7 a f 5 9 e 0 8 3 1
[3,3] f9 8 6 3 7 a c 5 e f b 0 1 4 2 d
  finv b c e 3 d 7 2 4 1 0 5 a 6 f 8 9
[3,4] f1 a 0 7 2 5 4 9 f e 8 6 3 d b c
  finv 2 0 4 c 6 5 b 3 a 7 1 e f d 9 8
[3,5] fb 2 d 3 e 6 4 a 5 f 9 8 1 7 0 c
  finv e c 1 3 6 8 5 d b a 7 0 f 2 4 9
[3,6] f3 a c 0 2 7 b 5 4 d e 1 6 f 8 9
  finv 3 b 4 0 8 7 c 5 e f 1 6 2 9 a d
[3,7] f9 4 e 6 5 b 7 c f d 1 a 0 2 3 8
  finv c a d e 1 4 3 6 f 0 b 5 7 9 2 8

[seed for encryption encoding]
96:08:0d:d5:6f:25:88:e7:92:da:38:c8:cf:4d:47:1b:
35:41:cc:78:19:9c:b1:9f:ca:22:e9:a3:4c:77:dc:68:

[Whitebox WF-LEA256 encryption]
P06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
A(P) 63:b8:cd:69:05:ea:4c:01:33:0e:cb:c5:60:a4:58:c6:

[ 0] c0c515e2:a90dafe9:e45c1f5e:69cdb863:
[ 1] 2feb93aa:308c94af:803c6dce:c0c515e2:
[ 2] e009c265:9e872434:eed15b8b:2feb93aa:
[ 3] a37298b0:0760fdd1:0798c29d:e009c265:
[ 4] 85ce1a15:7c142bf1:7dda6160:a37298b0:
[ 5] 49191db1:e7e8816c:6f079734:85ce1a15:
[ 6] 3391ea78:b8a22c3f:ae6f0484:49191db1:
[ 7] a7a1f304:957fbf46:265d6b56:3391ea78:
[ 8] d956f1b5:dfae0336:8728f552:a7a1f304:
[ 9] dcdd9b98:e950f3da:591c3baf:d956f1b5:
[10] 96b0cbea:6ea240a9:99273c96:dcdd9b98:
[11] bc89a8ab:6bb2c849:8d55ab52:96b0cbea:
[12] 16b0d9f0:70cc8820:068a39e5:bc89a8ab:
[13] c4efb284:f9357fcb:3bb97703:16b0d9f0:
[14] 15df3c3b:985fdc8e:21096603:c4efb284:
[15] f989946b:6bb1a845:1ca3f4ca:15df3c3b:
[16] 07e9370f:43808fdd:9a5877a8:f989946b:
[17] 5303ba5c:dad32e55:5a8669f8:07e9370f:
[18] 8f30c6d5:7e146695:13205129:5303ba5c:
```

```
[19] 25f535db:00d9175f:1bb9fddc:8f30c6d5:
[20] ac08e2e5:378aa096:57a73d42:25f535db:
[21] 7a77ecc0:5dd1fd0c:dee34d7a:ac08e2e5:
[22] afd6168a:35c7f0a4:9ddb9b95:7a77ecc0:
[23] 60277342:9c238221:3d62f23d:afd6168a:
[24] a4a06e35:90773321:35daf8f2:60277342:
[25] e02345aa:82b5c57c:b23a229a:a4a06e35:
[26] 09fd2119:a497cc8c:7acf60fc:e02345aa:
[27] 8febbe88:19be02ae:a09cfcd5:09fd2119:
[28] 30996dad:64c137e8:c26fc1a1:8febbe88:
[29] f288fe6b:86e42b05:da465a71:30996dad:
[30] 0adde707:eddd0ceb:844e9d49:f288fe6b:
[31] a968bdc4:b778d142:573c495d:0adde707:

Ewb(A(P))c4:bd:68:a9:42:d1:78:b7:5d:49:3c:57:07:e7:dd:0a:
B(Ewb(A(P))) f1:c5:c8:de:d3:ee:80:4f:b2:5e:88:52:42:26:7d:9e:
equal.

[seed for external encoding A]
b7:d1:9a:fc:b9:49:7f:dc:e0:38:1d:b5:7e:6f:53:7c:
dc:dc:f9:5f:ab:9a:f8:9b:da:1d:20:24:d5:28:27:95:
== External encoding A for decryption ==

[0,0] ff 8 6 2 0 4 1 c e a 3 b 9 7 d 5
  finv 4 6 3 a 5 f 2 d 1 c 9 b 7 e 8 0
[0,1] f3 e 1 9 5 4 6 b 0 c 7 a d 2 f 8
  finv 8 2 d 0 5 4 6 a f 3 b 7 9 c 1 e
[0,2] f0 f 5 d a c 7 9 2 1 e 8 6 4 b 3
  finv 0 9 8 f d 2 c 6 b 7 4 e 5 3 a 1
[0,3] f6 8 0 7 4 9 a d 3 2 5 f e 1 c b
  finv 2 d 9 8 4 a 0 3 1 5 6 f e 7 c b
[0,4] f3 2 7 b 9 a 0 d f 6 e c 8 1 4 5
  finv 6 d 1 0 e f 9 2 c 4 5 3 b 7 a 8
[0,5] f5 8 2 7 6 0 4 e d b 9 c 1 f a 3
  finv 5 c 2 f 6 0 4 3 1 a e 9 b 8 7 d
[0,6] fc a 3 7 f 4 9 d 5 2 0 b 1 6 8 e
  finv a c 9 2 5 8 d 3 e 6 1 b 0 7 f 4
[0,7] f3 7 d c a 1 b 6 9 2 f 0 4 5 8 e
  finv b 5 9 0 c d 7 1 e 8 4 6 3 2 f a
[1,0] f6 0 a 4 8 b d c e 2 3 f 5 1 7 9
  finv 1 d 9 a 3 c 0 e 4 f 2 5 7 6 8 b
[1,1] f0 e c d a 7 b f 1 5 6 8 3 9 4 2
  finv 0 8 f c e 9 a 5 b d 4 6 2 3 1 7
[1,2] f5 c a e 2 0 8 3 1 9 d 4 f 7 6 b
  finv 5 8 4 7 b 0 e d 6 9 2 f 1 a 3 c
[1,3] f4 a f e 9 2 d 1 b c 0 5 3 6 7 8
  finv a 7 5 c 0 b d e f 4 1 8 9 6 3 2
[1,4] f3 4 f 2 a b 5 e 9 0 c 7 8 d 1 6
```

```
  finv 9 e 3 0 1 6 f b c 8 4 5 a d 7 2
[1,5] f4 c 0 f 1 2 3 7 5 e b 6 9 8 d a
  finv 2 4 5 6 0 8 b 7 d c f a 1 e 9 3
[1,6] f8 d b 3 1 5 9 4 7 6 a e f c 0 2
  finv e 4 f 3 7 5 9 8 0 6 a 2 d 1 b c
[1,7] fe 6 b 1 a 3 d 9 c 5 0 f 7 4 8 2
  finv a 3 f 5 d 9 1 c e 7 4 2 8 6 0 b
[2,0] ff 7 3 5 c e 2 8 0 a 4 6 9 1 d b
  finv 8 d 6 2 a 3 b 1 7 c 9 f 4 e 5 0
[2,1] f7 8 0 9 e 4 b d 1 2 3 f a c 5 6
  finv 2 8 9 a 5 e f 0 1 3 c 6 d 7 4 b
[2,2] fd 6 e 1 8 7 9 c f 3 5 0 a 2 4 b
  finv b 3 d 9 e a 1 5 4 6 c f 7 0 2 8
[2,3] fb d a 7 f 9 c 2 8 e 6 5 3 0 4 1
  finv d f 7 c e b a 3 8 5 2 0 6 1 9 4
[2,4] f9 8 6 b 3 0 e f 1 a 2 5 c 4 7 d
  finv 5 8 a 4 d b 2 e 1 0 9 3 c f 6 7
[2,5] fd a 7 6 c 9 2 0 f 1 e 4 3 8 b 5
  finv 7 9 6 c b f 3 2 d 5 1 e 4 0 a 8
[2,6] fa 7 c d 8 e 4 b 9 6 2 0 5 f 1 3
  finv b e a f 6 c 9 1 4 8 0 7 2 3 5 d
[2,7] f6 b e 0 5 9 7 2 c a 3 f 8 4 d 1
  finv 3 f 7 a d 4 0 6 c 5 9 1 8 e 2 b
[3,0] ff 5 d e 6 b 0 3 a 7 2 9 1 c 8 4
  finv 6 c a 7 f 1 4 9 e b 8 5 d 2 3 0
[3,1] fa c 9 b 1 7 8 6 0 3 2 e 4 5 f d
  finv 8 4 a 9 c d 7 5 6 2 0 3 1 f b e
[3,2] f7 f 5 e 9 d 3 4 2 1 8 a 6 b 0 c
  finv e 9 8 6 7 2 c 0 a 4 b d f 5 3 1
[3,3] fc f 2 6 3 7 b 9 0 d 8 a e 1 4 5
  finv 8 d 2 4 e f 3 5 a 7 b 6 0 9 c 1
[3,4] f3 1 c a d 0 5 f b 6 8 e 9 7 4 2
  finv 5 1 f 0 e 6 9 d a c 3 8 2 4 b 7
[3,5] f2 7 0 d 5 9 f b 3 1 8 a 6 c 4 e
  finv 2 9 0 8 e 4 c 1 a 5 b 7 d 3 f 6
[3,6] f2 c 9 7 e 4 3 6 5 a 1 8 0 b f d
  finv c a 0 6 5 8 7 3 b 2 9 d 1 f 4 e
[3,7] f4 6 d 3 2 e 8 9 1 c 7 b 0 a f 5
  finv c 8 4 3 0 f 1 a 6 7 d b 9 2 5 e

[seed for external encoding B]
22:0e:b2:8a:c0:f4:1a:a2:65:ee:e9:09:a5:82:e8:72:
01:25:22:44:7c:11:86:c7:a8:29:7e:04:84:6c:98:54:
== External encoding B for decryption ==

[0,0] f5 b a 2 0 8 1 f 9 c e 7 d 4 6 3
  finv 4 6 3 f d 0 e b 5 8 2 1 9 c a 7
```

```
[0,1] f5 c 4 6 a 0 f 9 2 d 8 3 e b 1 7
  finv 5 e 8 b 2 0 3 f a 7 4 d 1 9 c 6
[0,2] f0 1 c e 6 8 5 4 7 a 9 3 2 d f b
  finv 0 1 c b 7 6 4 8 5 a 9 f 2 d 3 e
[0,3] ff a 4 3 e 8 d b 6 7 c 9 2 5 1 0
  finv f e c 3 2 d 8 9 5 b 1 7 a 6 4 0
[0,4] f8 b 3 e 9 f a 1 6 0 7 c 5 d 2 4
  finv 9 7 e 2 f c 8 a 0 4 6 1 b d 3 5
[0,5] f1 d e b 5 0 6 8 2 4 9 a f 3 7 c
  finv 5 0 8 d 9 4 6 e 7 a b 3 f 1 2 c
[0,6] f4 a 8 c 7 6 9 b 1 0 2 d e 5 f 3
  finv 9 8 a f 0 d 5 4 2 6 1 7 3 b c e
[0,7] f9 4 c e 1 7 d 3 6 b a 0 8 5 2 f
  finv b 4 e 7 1 d 8 5 c 0 a 9 2 6 3 f
[1,0] f8 2 d 5 9 6 0 3 c a e 7 f b 4 1
  finv 6 f 1 7 e 3 5 b 0 4 9 d 8 2 a c
[1,1] f1 e 8 b 4 a 6 c 2 0 7 d f 5 9 3
  finv 9 0 8 f 4 d 6 a 2 e 5 3 7 b 1 c
[1,2] ff 8 c 1 a e 7 2 0 b 6 9 4 3 5 d
  finv 8 3 7 d c e a 6 1 b 4 9 2 f 5 0
[1,3] f6 9 d b 5 a 7 0 8 1 4 f c 3 e 2
  finv 7 9 f d a 4 0 6 8 1 5 3 c 2 e b
[1,4] f9 b e 6 3 1 4 5 0 d 8 2 7 c f a
  finv 8 5 b 4 6 7 3 c a 0 f 1 d 9 2 e
[1,5] fd e 5 a c 0 6 9 4 8 1 2 b 3 f 7
  finv 5 a b d 8 2 6 f 9 7 3 c 4 0 1 e
[1,6] f4 a e c b 1 9 5 8 7 6 3 f d 0 2
  finv e 5 f b 0 7 a 9 8 6 1 4 3 d 2 c
[1,7] f4 7 e 8 c 3 1 0 9 a d 2 f 5 6 b
  finv 7 6 b 5 0 d e 1 3 8 9 f 4 a 2 c
[2,0] f1 8 4 c f b 5 2 3 9 e d a 6 7 0
  finv f 0 7 8 2 6 d e 1 9 c 5 3 b a 4
[2,1] f7 0 5 9 1 4 3 a 8 c 6 f d b 2 e
  finv 1 4 e 6 5 2 a 0 8 3 7 d 9 c f b
[2,2] f7 1 2 c e d f 0 5 9 4 3 a b 8 6
  finv 7 1 2 b a 8 f 0 e 9 c d 3 5 4 6
[2,3] fb 0 5 a 9 d e c 3 8 4 6 f 2 7 1
  finv 1 f d 8 a 2 b e 9 4 3 0 7 5 6 c
[2,4] f7 e 8 b 0 4 d 5 f 9 1 3 c 2 6 a
  finv 4 a d b 5 7 e 0 2 9 f 3 c 6 1 8
[2,5] f8 e a b 4 3 0 5 1 6 f 7 d c 9 2
  finv 6 8 f 5 4 7 9 b 0 e 2 3 d c 1 a
[2,6] f3 9 e d b 1 0 f 5 7 a 2 c 8 6 4
  finv 6 5 b 0 f 8 e 9 d 1 a 4 c 3 2 7
[2,7] ff 6 4 d c 9 3 7 e 2 b a 1 0 5 8
  finv d c 9 6 2 e 1 7 f 5 b a 4 3 8 0
[3,0] f4 8 1 e 3 c 2 5 0 6 9 a b f d 7
```

```
  finv 8 2 6 4 0 7 9 f 1 a b c 5 e 3 d
[3,1] f9 6 4 a b f 1 5 d 8 e 7 c 0 2 3
  finv d 6 e f 2 7 1 b 9 0 3 4 c 8 a 5
[3,2] f4 a 9 d 2 8 b c 6 0 7 f 1 e 3 5
  finv 9 c 4 e 0 f 8 a 5 2 1 6 7 3 d b
[3,3] f5 7 a d b 0 2 9 e 6 4 c 8 1 f 3
  finv 5 d 6 f a 0 9 1 c 7 2 4 b 3 8 e
[3,4] f5 4 a 2 b e 8 d 3 7 c f 1 0 6 9
  finv d c 3 8 1 0 e 9 6 f 2 4 a 7 5 b
[3,5] f4 e 6 1 c 9 3 d 8 f 7 0 2 b 5 a
  finv b 3 c 6 0 e 2 a 8 5 f d 4 7 1 9
[3,6] fa 4 d 9 e 7 c 1 6 5 3 f 2 0 8 b
  finv d 7 c a 1 9 8 5 e 3 0 f 6 2 4 b
[3,7] ff 8 e 7 c 2 0 1 9 d 3 5 4 a 6 b
  finv 6 7 5 a c b e 3 1 8 d f 4 9 2 0

[seed for decryption encoding]
c7:62:1a:ef:98:3b:b3:19:5f:0b:6d:d9:67:be:17:58:
d9:88:5f:06:4a:42:9e:18:c4:52:e4:c0:3e:57:3e:fc:

[Whitebox WF-LEA256 decryption]
Cf1:c5:c8:de:d3:ee:80:4f:b2:5e:88:52:42:26:7d:9e:
A(C) 88:ec:1f:58:94:76:53:a2:f3:94:f1:9c:1d:23:b7:cf:

[ 0] cfb7231d:660ad990:f6ea4b4f:3e205ac2:
[ 1] 3e205ac2:92803a92:acaca8a0:7fd9ccef:
[ 2] 7fd9ccef:59106c0e:e56eaefe:58847196:
[ 3] 58847196:0e40562a:26621823:75edbab0:
[ 4] 75edbab0:5cd14817:d09a8f6d:4d125306:
[ 5] 4d125306:8e1d15d2:7e59749a:4c3c3d5f:
[ 6] 4c3c3d5f:d0fc15ac:0f1865a3:fe691bef:
[ 7] fe691bef:00c3eaa6:c5178fc7:ec2f585d:
[ 8] ec2f585d:2467e2ca:ec7031ac:06b0feb1:
[ 9] 06b0feb1:c1a88c16:bb6c251f:3745122f:
[10] 3745122f:f3f8ed6e:aa9de65c:8ee36aca:
[11] 8ee36aca:e35f6566:f4bb6cdc:14d01253:
[12] 14d01253:1d7a7d63:010bc0d8:851f937e:
[13] 851f937e:244a06fa:ba18fcdf:b8441608:
[14] b8441608:7df1bf90:8aff5252:5b71a97c:
[15] 5b71a97c:0d0211e2:ece3e1f4:d207498d:
[16] d207498d:d4097d4c:74fe07d5:f6957a35:
[17] f6957a35:65985291:c6c6f284:69db6e89:
[18] 69db6e89:cd7f811e:ee0cb83e:ad1df5c3:
[19] ad1df5c3:ed4a2357:7f0e4587:83e96df7:
[20] 83e96df7:71d20bab:833baf06:a3ee2fa6:
[21] a3ee2fa6:025b0da7:1c261dd0:4aed12d9:
[22] 4aed12d9:5734541f:c8340833:f97d8fdd:
[23] f97d8fdd:3c4a975e:625706a8:d38719a7:
```

[24] d38719a7:690022c2:4274fdde:eefdc957:
[25] eefdc957:e837193c:a6e76c39:603895c6:
[26] 603895c6:29ff57df:f386d80a:f8b9aa77:
[27] f8b9aa77:02fffb32:5f8180c4:fa12f277:
[28] fa12f277:0cab1ba4:5c4c1037:53a5aac0:
[29] 53a5aac0:7cc3b5a4:1c373cbe:a370cf05:
[30] a370cf05:a66212a6:cf855aac:540b4b5e:
[31] 540b4b5e:5aa9c06a:c6dcf373:c4990011:

Dwb(A(C))5e:4b:0b:54:6a:c0:a9:5a:73:f3:dc:c6:11:00:99:c4:
B(Dwb(A(C))) 06:e3:1c:77:6e:cf:1d:36:ac:1c:cc:10:68:54:f7:4e:
equal.

제 2절 WB-CBC based on LEA

2.1 Length = 79-Bytes

key =
{0x04,0x4d,0x7e,0x5f,0x67,0x39,0xbb,0xfe,0xfb,0x71,0xf9,0x3b,0xd4,0xb0,0x93,0xd4};

IV =
{0x7e,0x76,0x96,0x17,0x0b,0x6d,0x09,0x65,0xa5,0x71,0x6c,0xbf,0xa3,0x2a,0x7e,0xbb};

BeSeed =
{0xc5,0xa0,0x6a,0x5c,0xf7,0x1b,0x9a,0x65,0xc6,0x33,0xa0,0x7c,0xef,0x96,0x24,0x6f,0xd3,0xb8,0x93,0x0d,0x4b,0x46,0x68,0xcb,0x44,0xfa,0xe8,0x6d,0x56,0x2d,0xeb,0x15};

CBCAeSeed =
{0x54,0x9a,0x3e,0x0c,0xb4,0x34,0x42,0x6a,0xfd,0xbd,0x1a,0xb7,0xa4,0x1a,0x34,0x45,0x30,0x79,0x49,0x17,0x8c,0x1a,0x41,0x25,0xfa,0x36,0x30,0x38,0x30,0x2f,0x47,0x88};

CBCBeSeed =
{0x60,0x0c,0x95,0xc7,0x5b,0xb9,0xd1,0x6a,0x5f,0x79,0xe2,0xb0,0xd9,0xff,0xa0,0x86,0xfd,0xab,0xd5,0x9a,0x4a,0x83,0x8f,0x15,0xb2,0xd2,0xf0,0xae,0xf0,0x3d,0x62,0xeb};

msg =
{0x4d,0xa0,0x5f,0x99,0x69,0x28,0x62,0xdc,0xdb,0x54,0x69,0x0c,0x70,0xac,0xbd,0x55,0xa0,0x12,0xc8,0x9d,0x6f,0xa0,0x04,0xe8,0xcf,0x33,0x6b,0x03,0x27,0xb2,0x64,0x13,0xf2,0x4c,0xb1,0x2f,0xab,0x60,0xca,0x68,0x8b,0x54,0xd3,0x3f,0x77,0x6c,0x7f,0xa7,0xa0,0x87,0xac,0x01,0xa7,0xa7,0x92,0x16,0x84,0xe3,0xf4,0x9a,0xeb,0x4f,0x0f,0xd5,0xee,0x69,0xc4,0x28,0x40,0x71,0x6d,0x80,0x13,0x0b,0xd6,0x92,0x54,0x75,0xa6};

Encoded (msg) :
{0x25,0x18,0x70,0x02,0x61,0xe0,0xd8,0xbe,0xf4,0x62,0x3d,0xd9,0x8b,0xbe,0x9

0,0xed,0x48,0x09,0x6c,0x05,0x6a,0xf2,0x4b,0x0f,0x22,0x29,0x33,0xdf,0xf4,0x
d1,0xaa,0x51,0xc2,0x3a,0xc3,0x1d,0xf7,0x12,0x1d,0x8f,0xd4,0x62,0x6a,0x0a,0
x84,0x3e,0xf4,0x77,0x48,0xac,0x55,0x43,0xff,0xfe,0xe8,0x95,0xd7,0x09,0xcc,
0xb5,0x75,0x85,0x04,0x6d,0x97,0x5d,0x6b,0x11,0x80,0x87,0xd7,0xf1,0x6e,0xbf
,0x6e,0xbb,0x3a,0x0f,0x32,0x8e};

dec :
{0x39,0xe1,0xd3,0x18,0xbd,0x2d,0x26,0xa3,0xd7,0x36,0x50,0xb1,0xf4,0xf3,0x4
e,0xe1,0xf4,0xca,0x23,0xf5,0xa2,0xf7,0x90,0x9b,0x29,0xf1,0x1c,0x90,0x37,0x
45,0xa2,0x73,0xf1,0xf2,0x96,0xe1,0x3e,0xc7,0x0f,0x26,0xd2,0x9a,0x46,0x2f,0
x03,0x9b,0xe8,0x63,0x48,0xbe,0x0c,0x69,0x5b,0x8e,0x57,0x4a,0xa4,0xec,0x5a,
0x1b,0x45,0x63,0x5a,0x9c,0xd8,0x83,0xd9,0x50,0x11,0x9f,0x38,0x22,0xa5,0x7a
,0x95,0xa3,0x8e,0x99,0x0c,0x23};

rec :
{0xc1,0x50,0xdf,0xd8,0x05,0xe6,0xdd,0x1b,0xae,0x2f,0x91,0xe5,0xc7,0x20,0x8
1,0x4c,0x0f,0x3f,0xc4,0xd7,0x00,0x89,0x53,0x73,0xca,0xe4,0x9e,0xe8,0xad,0x
4f,0xcd,0x09,0x53,0x6b,0xb2,0x1c,0x5b,0xf9,0x66,0xf3,0x6e,0x2f,0x3b,0x59,0
xcd,0xe0,0x96,0x78,0x0f,0xb9,0x0e,0x6f,0x5d,0x8d,0x1d,0x2c,0x60,0x64,0x57,
0xba,0x4b,0x34,0x16,0xac,0x6b,0x8d,0xc7,0x10,0xe8,0x93,0xdb,0xef,0xf6,0x4a
,0x34,0xb6,0xd3,0x8e,0xd3,0xe0};

DEcode(rec):
{0x4d,0xa0,0x5f,0x99,0x69,0x28,0x62,0xdc,0xdb,0x54,0x69,0x0c,0x70,0xac,0xb
d,0x55,0xa0,0x12,0xc8,0x9d,0x6f,0xa0,0x04,0xe8,0xcf,0x33,0x6b,0x03,0x27,0x
b2,0x64,0x13,0xf2,0x4c,0xb1,0x2f,0xab,0x60,0xca,0x68,0x8b,0x54,0xd3,0x3f,0
x77,0x6c,0x7f,0xa7,0xa0,0x87,0xac,0x01,0xa7,0xa7,0x92,0x16,0x84,0xe3,0xf4,
0x9a,0xeb,0x4f,0x0f,0xd5,0xee,0x69,0xc4,0x28,0x40,0x71,0x6d,0x80,0x13,0x0b
,0xd6,0x92,0x54,0x75,0xa6};

2.2 Length = 80-Byte

key =
{0x03,0xa3,0xee,0x8f,0xac,0x3c,0x52,0x1d,0xe7,0x71,0x48,0x8d,0x2d,0x61,
0x7d,0x67};

IV =
{0x5e,0xef,0xe6,0x89,0xc9,0xd6,0xf0,0xdd,0xa9,0x62,0xcc,0x76,0x61,0x78,
0xaa,0x8b};

BeSeed =
{0xfd,0x7c,0xc9,0xad,0x66,0x3c,0x14,0xcf,0x15,0xf6,0x86,0xee,0x38,
0xe3,0x4b,0xd0,0xc2,0x8e,0xde,0xf9,0xf5,0x85,0xa9,0x60,0xa5,0xac,0x1f,0x47
,0x9d,0x0c,0x4a,0x99};

CBCAeSeed
={0x83,0x73,0x41,0x16,0x9d,0xf5,0xb3,0x18,0x19,0xc4,0x3e,0x34,0xa0,
0x3d,0x87,0x5b,0x46,0xb0,0x0b,0xb9,0xcf,0x22,0x17,0x5b,0xfe,0x4d,0x6f,0xee
,0xd8,0xa6,0x8b,0xc6};

```
CBCBeSeed
={0x6b,0xee,0x50,0x88,0xa8,0x7c,0x7b,0x7f,0xb7,0x2d,0x47,0xc0,0x40,
0x3c,0x33,0x71,0x7d,0xda,0x04,0x05,0x0a,0x60,0xe8,0x75,0x9f,0xb0,0xd2,0x92
,0xfe,0x54,0xd7,0xa9};

msg =
{0xcf,0x09,0x80,0xba,0x85,0xeb,0x26,0x40,0x4c,0xee,0x91,0x03,0x16,0xf3,
0x9a,0x84,0x6c,0x44,0x97,0xfb,0x8d,0x11,0x30,0x56,0xbf,0xf3,0x40,0xc1,0x4c
,0x80,0xc6,0x94,0xec,0x8e,0x04,0x5d,0xf9,0x7b,0x59,0xcc,0x98,0x54,0xa8,0x1
a,0x31,0x92,0xf7,0x23,0x98,0xb2,0x60,0x1e,0x70,0xc4,0x02,0x89,0x40,0x58,0x
3a,0x0e,0x9c,0xf6,0x21,0x20,0xd0,0x1f,0x34,0x77,0x69,0xb0,0x7e,0xd0,0x99,0
x4c,0x85,0xd6,0x6d,0x39,0xa1,0x43};

Encoded msg :
{0xc6,0x03,0x43,0xc7,0x96,0xcb,0x5c,0x5d,0x01,0x9b,0xd0,0x8a,
0xf3,0xed,0xd6,0x5f,0x09,0xc9,0xa8,0x46,0x97,0x07,0x4d,0xe3,0x30,0x11,0x08
,0x72,0xbd,0xaf,0xbc,0xcf,0xa9,0xdb,0xc9,0x98,0xac,0xeb,0x31,0xfb,0x77,0x5
6,0xba,0xfb,0x6c,0x70,0x40,0x8e,0xd5,0x8c,0x03,0x2d,0x04,0x5d,0xc6,0xc7,0x
0f,0x5d,0x5d,0x8d,0xed,0xe4,0x83,0x8a,0xfb,0x90,0x29,0xee,0x2c,0xa4,0xfe,0
xbd,0x76,0x3e,0x46,0xc4,0x99,0x5e,0x63,0x2e,0x4b,0x9e,0x33,0x25,0x64,0x04,
0x1d,0x0d,0xff,0xe8,0x68,0xf8,0xfb,0xbf,0xe8,0x3a};

dec :
{0xbe,0xc7,0x6d,0xf9,0xa9,0xa0,0xff,0x29,0x5c,0x30,0x24,0xa0,0x7e,0x7c,
0x17,0x55,0x5f,0xad,0x73,0xd0,0xdc,0xb8,0x22,0xec,0x19,0x4a,0xc5,0x48,0xf9
,0xdb,0xe9,0x42,0xa1,0x53,0xb0,0xaa,0x6d,0xb2,0x49,0xe4,0x3f,0xda,0x2a,0x2
9,0x06,0xa7,0x94,0xc7,0x1e,0x6a,0xdf,0x14,0x51,0x75,0x61,0x57,0x8b,0xbc,0x
82,0x1e,0xf4,0xff,0x28,0x97,0x86,0x2a,0x76,0x31,0xae,0x53,0xda,0x49,0x4f,0
x00,0xe1,0xee,0xe4,0xd0,0x4a,0xca,0xcb,0x19,0x8a,0x82,0x35,0x1a,0x4c,0x16,
0xa4,0x97,0xeb,0xc6,0xd7,0x49,0x76,0x5a};

rec :
{0x49,0xb0,0xc6,0x50,0x26,0x7a,0xf9,0x1d,0x13,0xac,0xf6,0x06,0x6a,0x56,
0x4c,0xda,0x8d,0x87,0xdb,0x6c,0x20,0xd7,0x77,0xc7,0xbe,0xb3,0x50,0x71,0xd7
,0x72,0x7a,0x9a,0xad,0x2b,0x7e,0x44,0xc8,0xfa,0x9c,0xff,0x58,0x17,0x18,0x3
2,0x16,0x15,0x97,0x68,0x2a,0xd4,0x46,0x37,0xa7,0xe8,0x21,0x5c,0x11,0x10,0x
cd,0x0b,0xc7,0x59,0x56,0x6b,0x0b,0x36,0x3e,0x95,0x98,0x50,0xd0,0x8d,0x52,0
x39,0x3e,0x2a,0xb8,0x40,0x26,0x88,0x3b,0x31,0x26,0x36,0x87,0xd0,0x37,0x2d,
0xe1,0xda,0xd0,0x3e,0x6c,0xe2,0x09,0x7b};

DEcode(rec ):
{0xcf,0x09,0x80,0xba,0x85,0xeb,0x26,0x40,0x4c,0xee,0x91,0x03,0x16,0xf3,0x9
a,0x84,0x6c,0x44,0x97,0xfb,0x8d,0x11,0x30,0x56,0xbf,0xf3,0x40,0xc1,0x4c,0x
80,0xc6,0x94,0xec,0x8e,0x04,0x5d,0xf9,0x7b,0x59,0xcc,0x98,0x54,0xa8,0x1a,0
x31,0x92,0xf7,0x23,0x98,0xb2,0x60,0x1e,0x70,0xc4,0x02,0x89,0x40,0x58,0x3a,
0x0e,0x9c,0xf6,0x21,0x20,0xd0,0x1f,0x34,0x77,0x69,0xb0,0x7e,0xd0,0x99,0x4c
,0x85,0xd6,0x6d,0x39,0xa1,0x43};
```

2.3 Length = 81-Byte

```
key =
{0x2e,0xab,0x28,0xc1,0xd4,0xdd,0xc8,0x4b,0xa6,0xc9,0x73,0xdb,0xec,0x7a,
0x02,0x9f};

IV =
{0x21,0x9c,0x10,0x0b,0x4d,0x6b,0xb5,0x0d,0x84,0xd0,0x28,0x1d,0xbb,0x6b,
0x9f,0x7b};

BeSeed =
{0xd0,0x0a,0xa2,0x46,0x28,0xbb,0xc6,0xc6,0x4e,0xd8,0x87,0xd6,0xdb,
0xaf,0x71,0x61,0x2d,0x1d,0x24,0x0a,0x09,0xce,0x1b,0xd7,0x0f,0xf5,0x53,0x5f
,0xae,0x21,0x51,0x61};

CBCAeSeed
={0xfb,0xd4,0x5d,0xef,0xfa,0x87,0x91,0x20,0x58,0x59,0x69,0x99,0x1e,
0xd7,0xd5,0xcd,0x79,0x7f,0x4c,0x3c,0xb1,0xb4,0x8a,0x93,0xdc,0xac,0x89,0x1c
,0x2a,0xdc,0x47,0x23};

CBCBeSeed
={0x32,0xf3,0x53,0x6b,0xac,0xc2,0x57,0xf4,0x46,0x0a,0xa3,0x07,0xe7,
0xca,0xbe,0x91,0x0d,0x99,0x4f,0xaa,0xed,0x29,0x0d,0x04,0x7b,0xdc,0x73,0x9f
,0x35,0x08,0xab,0x82};

msg =
{0xee,0x0f,0x00,0xcc,0x36,0x0a,0x3b,0xe5,0x46,0x3b,0xe5,0x90,0x98,0xae,
0xe0,0x24,0x1c,0xf6,0xb1,0x48,0x79,0xe0,0x67,0x2a,0x5a,0x37,0xd2,0xaa,0x6f
,0xd1,0x64,0xb1,0xee,0x8d,0x07,0x4a,0x0c,0x99,0x69,0x70,0x5d,0xad,0xca,0x6
8,0x4b,0xd9,0xd9,0x59,0xa2,0xf4,0xf9,0x77,0x33,0x75,0x3d,0x44,0xac,0x0b,0x
6e,0x5a,0x7c,0xfe,0xac,0x38,0x1e,0x3e,0x59,0xa0,0x6f,0x5c,0x97,0xae,0x90,0
x1e,0xfe,0x3e,0xf9,0x38,0xa5,0x24,0x46};

Encoded msg :
{0x88,0x63,0xad,0xae,0xb3,0x79,0x7e,0x13,0xda,0x6f,0xa2,0x0a,
0xc0,0x92,0xac,0x11,0xa2,0xd7,0xf9,0x78,0xde,0x84,0x8d,0x91,0xf0,0x60,0x31
,0xf7,0x48,0xc8,0x89,0xf7,0x88,0xf8,0xa3,0x77,0xf5,0xca,0x8b,0xe0,0xfc,0xb
8,0xb3,0x8f,0x5c,0xcc,0x5b,0x36,0x11,0xd5,0x7f,0x29,0xb1,0x16,0x75,0xc5,0x
b2,0x4f,0xc5,0x47,0xa1,0xd2,0xc6,0x2f,0xa8,0x0f,0x3f,0x5f,0x0d,0x47,0xdd,0
x67,0xe9,0x15,0xe5,0x11,0x0a,0xb0,0xc3,0x11,0x99,0x63,0xa5,0xe1,0xfd,0x7f,
0xf6,0x08,0x2d,0x47,0x06,0x2d,0xf8,0x6e,0x95,0x49};

dec :
{0xa6,0xf6,0x46,0x4a,0x1d,0xf6,0x21,0x47,0xb4,0x47,0x22,0xad,0x00,0x6c,0xc
2,0xc1,0x89,0x64,0x74,0x3b,0x77,0x66,0x19,0xa4,0x0a,0xac,0xda,0xaf,0x50,0x
c3,0xd5,0xb5,0x47,0x92,0xf8,0x70,0x3a,0x79,0x9a,0xbd,0x61,0x32,0x83,0x76,0
x19,0x85,0x5a,0x2b,0x14,0x73,0x5c,0x5a,0x0f,0xe9,0x6b,0x2a,0xfe,0xe4,0x9d,
0xf1,0x5c,0xa9,0xbc,0x28,0xde,0x7b,0xaf,0xed,0xb5,0x05,0xc9,0x8a,0x41,0x9e
,0x53,0xbb,0xb5,0xf6,0x4c,0xa6,0xe1,0x8c,0xae,0x09,0x9e,0xbe,0xa1,0xd9,0x5
a,0xd1,0xf0,0x93,0xc4,0x4f,0xfb,0xd2};
```

rec :
{0x31,0x15,0xe2,0x50,0x5b,0xc5,0x74,0xd3,0xb6,0x58,0xc8,0xca,0xae,0xad,0x3b,0xe7,0x03,0xda,0x94,0xa5,0xbe,0x33,0xbc,0x95,0x08,0x5d,0x87,0x22,0x57,0x56,0x92,0x5f,0x31,0x01,0xe8,0xa2,0x34,0xb6,0xbf,0x66,0x05,0xdb,0xd5,0xe0,0x6a,0x53,0x65,0x7c,0x28,0xdb,0x17,0x4e,0x5f,0xa9,0x73,0x3e,0x1a,0x68,0xea,0x02,0xeb,0x3d,0xcc,0x33,0x01,0xed,0xf7,0xcb,0xad,0x92,0x9c,0x58,0x20,0xb9,0x7a,0xd3,0xd3,0x07,0xc8,0xe7,0x1b,0x15,0xed,0x06,0x3d,0xca,0xad,0xe7,0xe9,0x6c,0xf4,0xad,0xb7,0xdf,0x57,0xba};

DEcode(rec):
{0xee,0x0f,0x00,0xcc,0x36,0x0a,0x3b,0xe5,0x46,0x3b,0xe5,0x90,0x98,0xae,0xe0,0x24,0x1c,0xf6,0xb1,0x48,0x79,0xe0,0x67,0x2a,0x5a,0x37,0xd2,0xaa,0x6f,0xd1,0x64,0xb1,0xee,0x8d,0x07,0x4a,0x0c,0x99,0x69,0x70,0x5d,0xad,0xca,0x68,0x4b,0xd9,0xd9,0x59,0xa2,0xf4,0xf9,0x77,0x33,0x75,0x3d,0x44,0xac,0x0b,0x6e,0x5a,0x7c,0xfe,0xac,0x38,0x1e,0x3e,0x59,0xa0,0x6f,0x5c,0x97,0xae,0x90,0x1e,0xfe,0x3e,0xf9,0x38,0xa5,0x24,0x46};

제 3절 WB-CTR based on LEA

3.1 Length = 79-Byte

key =
{0x06,0xa1,0x9f,0x01,0xdb,0x34,0x8d,0xc1,0x25,0x72,0x5b,0x99,0x23,0x4e,0xc0,0xae};

IV =
{0xbf,0x84,0xf8,0x32,0x90,0x9a,0x3a,0x76,0x9c,0x8f,0xaa,0x51,0x26,0x8c,0x26,0x1a};

AeSeed =
{0xca,0xe8,0x45,0x83,0x8a,0x87,0x56,0x06,0xb4,0x6f,0xf2,0x68,0xf2,0x51,0xa4,0xb8,0xda,0xa4,0x9e,0x6a,0xef,0xe9,0x30,0x0f,0x0c,0xa3,0x5e,0x0a,0x73,0x4c,0xa9,0xaa};

CTRGSeed =
{0xc9,0xc7,0xd5,0xa3,0xf0,0x62,0xdd,0xd7,0xc5,0x05,0xc5,0xaa,0x83,0x0e,0x22,0xbf,0xd8,0xdf,0x7a,0x2f,0x31,0xaf,0x7e,0x41,0x39,0xa6,0xf7,0xca,0xb5,0xdd,0x8b,0x84};

CTRHSeed =
{0xdd,0xae,0xd7,0x6c,0x69,0x91,0x87,0x0f,0x87,0xa7,0xd1,0xfc,0x1a,0x7e,0xdc,0x7f,0xc0,0xa3,0xde,0x29,0xa4,0x4b,0xb0,0xab,0xb6,0x18,0x54,0xa4,0xb8,0x12,0xa9,0xe5};

CTRHdSeed =
{0x28,0xcc,0xd7,0x40,0xcb,0xd5,0x9f,0x1e,0x3f,0x7a,0xf3,0xd2,0xb8,0xf4,0x54,0x4d,0x7d,0x89,0xe6,0x42,0x02,0xe0,0x85,0x9d,0x8e,0x16,0xf2,0xf2,0x57,0xab,0x1e,0x53};

```
msg =
{0x49,0xcc,0x1c,0x56,0x30,0xa1,0xda,0x13,0xf8,0x22,0x18,0x1f,0x24,0x1d,0x0
2,0xf8,0x09,0xac,0x29,0xe1,0x32,0xbc,0xad,0x0b,0xed,0xb5,0xc1,0x88,0xdd,0x
15,0xa2,0x12,0xfd,0xc9,0x0b,0xd3,0x0f,0x29,0xad,0xa0,0x72,0x54,0x29,0x88,0
x05,0x1e,0x90,0xb1,0xb0,0x31,0x42,0xc7,0x15,0x6e,0xb2,0x2f,0x0b,0xf7,0x66,
0xb2,0x89,0x03,0xea,0x40,0x29,0xfe,0xe5,0x8b,0xef,0xf5,0x4d,0xe0,0x08,0x89
,0x79,0x09,0x20,0xec,0xb0};

Encoded msg :
{0x7e,0x71,0xe7,0x6f,0xa6,0xf4,0xa6,0xa4,0x56,0xf2,0xf3,0x00,0x29,0x83,0x1
6,0xbe,0x2e,0x81,0xab,0xd4,0xa5,0xbd,0xcd,0x9c,0x94,0x2f,0x12,0x4e,0xc7,0x
89,0x66,0x2d,0xa9,0x7e,0xcc,0xfb,0xcc,0x0a,0xcd,0xfa,0x3f,0x06,0x09,0x4e,0
x0b,0x86,0x08,0x91,0xbc,0x1a,0x3e,0x53,0x33,0xde,0xf8,0xd0,0x45,0x55,0xab,
0x68,0xdf,0x15,0xcb,0x06,0xce,0xd5,0x73,0x7a,0x7c,0x53,0xbd,0xea,0x46,0xe9
,0xc9,0xaf,0x22,0xb7,0xb8,0xd1};

dec :
{0x4d,0x8b,0x7e,0x7d,0x3e,0xd6,0xb4,0x40,0x63,0xf8,0xf8,0x73,0x59,0x6a,0x3
6,0xf0,0x96,0x33,0x7a,0x2c,0x04,0x9f,0xc4,0xe8,0x7e,0xa8,0x44,0x91,0x54,0x
ed,0x69,0x21,0x98,0x30,0xcf,0xf3,0x3b,0x14,0x22,0xfe,0x4e,0xa3,0x36,0x79,0
x75,0x17,0x7c,0xe0,0x3a,0x8a,0xf7,0xcc,0x1d,0xec,0x60,0x67,0xf4,0x69,0xcf,
0xbe,0xe9,0x42,0xe3,0x6f,0x87,0x9d,0x8d,0xbd,0x3d,0xca,0xa9,0x65,0xc6,0x22
,0x2c,0xbd,0xc4,0x90,0x87,0x94};

rec :
{0x00,0x29,0x86,0xec,0xc9,0x10,0xca,0x54,0xc2,0x2b,0x1f,0x29,0x1f,0x7d,0x7
4,0x92,0x20,0x39,0x30,0x82,0xc0,0x58,0x22,0x48,0xe9,0x6e,0x07,0x4c,0x93,0x
74,0x94,0xdf,0x14,0x24,0x04,0x3b,0x65,0x02,0x22,0x7e,0x6e,0xf1,0x31,0x4c,0
x88,0x7f,0xd3,0x86,0x82,0xef,0xf2,0xf4,0x93,0xf7,0xae,0xc9,0xb6,0x17,0xbb,
0xf4,0xc5,0x35,0xe0,0x04,0xa0,0xc8,0x95,0x47,0x15,0xca,0x42,0x6e,0xb2,0xaa
,0xc1,0x73,0x17,0x4a,0x83,0x46};

DEcode(rec ):
{0x49,0xcc,0x1c,0x56,0x30,0xa1,0xda,0x13,0xf8,0x22,0x18,0x1f,0x24,0x1d,0x0
2,0xf8,0x09,0xac,0x29,0xe1,0x32,0xbc,0xad,0x0b,0xed,0xb5,0xc1,0x88,0xdd,0x
15,0xa2,0x12,0xfd,0xc9,0x0b,0xd3,0x0f,0x29,0xad,0xa0,0x72,0x54,0x29,0x88,0
x05,0x1e,0x90,0xb1,0xb0,0x31,0x42,0xc7,0x15,0x6e,0xb2,0x2f,0x0b,0xf7,0x66,
0xb2,0x89,0x03,0xea,0x40,0x29,0xfe,0xe5,0x8b,0xef,0xf5,0x4d,0xe0,0x08,0x89
,0x79,0x09,0x20,0xec,0xb0};
```

3.2 Length = 80-Byte

```
key =
{0xdb,0x98,0x65,0xcf,0xb4,0x93,0x17,0x93,0x65,0x1a,0x30,0x4b,0x64,0x35,0x3
b,0x76};

IV =
{0xfc,0xd8,0xcd,0xb0,0x0c,0x05,0x75,0x46,0xc1,0x21,0x4e,0xaa,0xcd,0x99,0x3
1,0x2b};
```

```
AeSeed =
{0x9e,0xf1,0x42,0x95,0xa6,0x6a,0x5f,0xbc,0xbf,0x9b,0x16,0x59,0xb3,0xad,0x5
4,0x38,0x04,0xff,0x2b,0xc5,0xd4,0xac,0xe9,0x50,0x71,0xfc,0x34,0x67,0x8b,0x
08,0xd4,0x11};

CTRGSeed =
{0xe9,0x4d,0x10,0xa8,0x0f,0xd8,0x03,0x4c,0xfe,0x31,0x8f,0x4e,0x15,0x41,0x7
0,0x5f,0x58,0xb1,0xbc,0x9f,0x42,0xf0,0xa4,0x0a,0x11,0x67,0x8d,0x1a,0x9a,0x
06,0x32,0x5c};

CTRHSeed =
{0x89,0x51,0x34,0xd1,0xd8,0x2a,0x43,0x00,0x8a,0xeb,0xfc,0x09,0x13,0x96,0x6
0,0x51,0xbd,0xbc,0xae,0x3e,0x7a,0xe7,0x1f,0x1b,0x14,0x39,0xed,0xcd,0x54,0x
38,0x74,0x9e};

CTRHdSeed =
{0x54,0xc6,0x7b,0x22,0x73,0x79,0x32,0x08,0xbb,0xa7,0xc7,0x72,0xd7,0xd2,0x7
4,0xcb,0x5c,0x2e,0x8d,0xec,0x30,0x75,0x7b,0xb5,0x0a,0x9b,0x95,0x92,0xff,0x
df,0xdd,0xe3};

msg =
{0x2a,0xc6,0x9c,0x44,0x7f,0x82,0xc5,0x6e,0xfe,0xd4,0xc4,0x91,0xa1,0x63,0xb
c,0x58,0x58,0xfa,0x0f,0x95,0x46,0xee,0x75,0x37,0x52,0x70,0x2f,0x9f,0xba,0x
c5,0x03,0xf5,0xfb,0xc9,0x07,0xe6,0xfc,0x0b,0x9d,0xfc,0xac,0xfb,0x07,0x3b,0
xeb,0xd8,0xae,0x7b,0xa6,0xef,0xaa,0x6e,0x53,0xbd,0x77,0x75,0x9f,0x45,0x32,
0x66,0xa8,0xfb,0x5e,0x28,0xdb,0xde,0xc0,0x62,0xe9,0x48,0x34,0x02,0x11,0xb7
,0x00,0xa1,0x95,0xed,0xad,0xe9};

Encoded msg :
{0x0c,0xdf,0x67,0xeb,0x3b,0x4d,0x4b,0xde,0x09,0x53,0x2f,0xbe,0x0d,0xa6,0x6
f,0x27,0x69,0xe3,0x84,0xd4,0x20,0xe7,0xdb,0xb4,0xf0,0x20,0xbe,0xbb,0xae,0x
3f,0x9e,0x9f,0x21,0xde,0x82,0x86,0xde,0xcc,0xa3,0x0c,0xe7,0xf8,0x53,0x32,0
x98,0xfa,0x31,0x08,0x9a,0x8d,0xbc,0xa0,0xc1,0x9f,0xd6,0x1f,0xb6,0x4d,0xe2,
0xd9,0x05,0x8d,0xa1,0x67,0x71,0xcb,0x98,0xaa,0x67,0xd4,0x84,0x90,0x42,0xb9
,0x5a,0x6e,0xc0,0xd9,0x3b,0xb5,0xe5,0x64,0x48,0x53,0x96,0xb8,0x9a,0x52,0x4
a,0x00,0x1a,0x01,0xd1,0x21,0xb2,0xdb};

dec :
{0x1c,0xf0,0xbb,0x2b,0x1e,0x8e,0x9a,0x7c,0xd3,0x1c,0x0c,0x64,0xff,0xaa,0x1
d,0x0a,0x57,0x13,0x32,0xf0,0x12,0x1e,0xe1,0xdd,0x22,0xdd,0x4f,0x93,0x40,0x
ae,0x72,0x3b,0xa7,0x7a,0xda,0xa7,0x95,0x54,0xd8,0x0e,0x1f,0xe6,0x2b,0x54,0
xd5,0xda,0x6f,0x03,0x1f,0xe6,0xc2,0xd8,0xe3,0xc4,0x3d,0x8d,0x13,0x7f,0x38,
0x23,0xe4,0x33,0x16,0x92,0x47,0xc9,0xb4,0x41,0x29,0x5b,0x6e,0xf4,0x8d,0x5b
,0x44,0x22,0xc8,0x15,0xa7,0xa6,0xcf,0xc4,0x01,0x3c,0x68,0xf6,0xa5,0xdf,0x4
6,0x99,0x1d,0x58,0xfe,0x69,0x3a,0xe3};

rec :
{0x10,0x85,0x80,0xa2,0xcd,0x2c,0x64,0x2f,0xe4,0x7d,0xe3,0x65,0x11,0x99,0xe
c,0x22,0xf4,0x09,0x7e,0x19,0xe9,0x3f,0xa4,0x8b,0xd0,0xc7,0x32,0x6c,0xb3,0x
```

```
dc,0xfe,0x7d,0x2d,0x87,0x7a,0x6d,0x32,0x50,0x89,0xd6,0x9d,0xe2,0x5d,0x83,0
x6b,0x6b,0xc5,0x84,0x77,0xa2,0x63,0xcc,0x57,0xd8,0xad,0x02,0xfe,0xd4,0x71,
0x4b,0x16,0x5e,0x25,0x02,0x3d,0x31,0x5d,0xcb,0xb8,0x1b,0xd2,0x61,0x12,0x89
,0x5e,0x55,0x54,0x7a,0xca,0xa5,0xd5,0xcb,0xfd,0x27,0x01,0xb7,0x4f,0x70,0x1
6,0x47,0x1e,0x34,0x7c,0xf1,0x76,0xd8};

DEcode(rec ):
{0x2a,0xc6,0x9c,0x44,0x7f,0x82,0xc5,0x6e,0xfe,0xd4,0xc4,0x91,0xa1,0x63,0xb
c,0x58,0x58,0xfa,0x0f,0x95,0x46,0xee,0x75,0x37,0x52,0x70,0x2f,0x9f,0xba,0x
c5,0x03,0xf5,0xfb,0xc9,0x07,0xe6,0xfc,0x0b,0x9d,0xfc,0xac,0xfb,0x07,0x3b,0
xeb,0xd8,0xae,0x7b,0xa6,0xef,0xaa,0x6e,0x53,0xbd,0x77,0x75,0x9f,0x45,0x32,
0x66,0xa8,0xfb,0x5e,0x28,0xdb,0xde,0xc0,0x62,0xe9,0x48,0x34,0x02,0x11,0xb7
,0x00,0xa1,0x95,0xed,0xad,0xe9};
```

3.3 Length = 81-Byte

```
key =
{0x35,0x51,0x1b,0x75,0xeb,0xc8,0xa8,0x76,0x36,0xcd,0x49,0xa4,0x7e,0xa2,
0x9e,0x9b};

IV =
{0x04,0x4e,0xe6,0xea,0x1f,0x8a,0x60,0xc8,0x64,0x3a,0x82,0x1d,0x86,0x43,
0x6b,0xc7};

AeSeed =
{0x4f,0x12,0xa4,0x3e,0xb9,0xee,0x87,0x99,0xef,0x93,0x92,0x66,0x9d,0xaf,0x8
a,0x1c,0x52,0x62,0x62,0x44,0x43,0xbc,0x57,0x26,0xaf,0x1b,0xb8,0x4c,0xc7,0x
ac,0x6a,0xff};

CTRGSeed =
{0x8a,0x63,0xaf,0x0f,0x17,0x86,0xce,0x9b,0xaf,0xaa,0x76,0xee,0xdf,0x04,0xa
4,0x30,0xa8,0xd2,0x0f,0x47,0x88,0x15,0xe5,0x1a,0x15,0x2b,0x5a,0x7e,0x96,0x
cc,0x28,0x12};

CTRHSeed =
{0xee,0xcf,0xfd,0xa6,0x2a,0x05,0x37,0x63,0xe6,0x5b,0xcb,0x63,0x21,0x0a,0x5
c,0x84,0x40,0xe5,0xbc,0xcb,0x27,0xa3,0x3d,0xab,0x13,0xef,0x8a,0x6f,0xf8,0x
aa,0x3f,0xa7};

CTRHdSeed =
{0x13,0x8b,0x43,0xde,0xec,0xc9,0x18,0x93,0x0a,0x86,0x9d,0xa7,0x95,0x59,0xb
a,0xb5,0x1a,0x98,0x20,0x9d,0xad,0x74,0x34,0xa3,0x22,0xdf,0x25,0x51,0xd5,0x
17,0x96,0xb6};

msg =
{0x60,0x29,0x18,0xe2,0x6f,0xb1,0xdd,0x24,0x2d,0x0a,0xca,0xd2,0x0e,0xba,0xd
2,0xde,0x8a,0x91,0x06,0xb6,0xa3,0xc4,0x35,0x27,0xc5,0xfc,0xff,0x7a,0x6a,0x
2c,0xbd,0x2c,0x16,0xc2,0xc3,0x08,0xe9,0xda,0x84,0xb3,0x04,0xad,0xf9,0x69,0
x39,0xc9,0x94,0xfa,0xda,0xc6,0x08,0xab,0xb4,0xad,0x2d,0x1d,0x87,0xd4,0x80,
```

0x2f,0xa6,0x72,0x2b,0x2e,0xec,0x47,0x4a,0x7b,0x53,0xa9,0x24,0x7e,0xea,0x58
,0x38,0x60,0x45,0xd8,0xc8,0xb3,0xc0};

Encoded msg :
{0x9e,0x83,0x4c,0x75,0x66,0x5a,0x37,0x1a,0xce,0x7f,0xef,0x4f,0x9c,0xae,0x7
e,0x71,0xf5,0xc4,0x26,0xb6,0x0b,0x7b,0x12,0x15,0x94,0x54,0x91,0xd0,0xa9,0x
89,0xc7,0x6a,0x12,0x7a,0x73,0x2a,0x7a,0xfd,0xa9,0xb1,0xd9,0xa0,0x9b,0xc7,0
x6f,0xcd,0xa9,0x58,0xc5,0x7b,0x2c,0x54,0xad,0xd9,0x57,0x9f,0xa2,0x2e,0x38,
0x74,0xb0,0x3f,0x9d,0x61,0x76,0x02,0xa9,0x14,0x3b,0xd4,0x59,0x52,0x3a,0x82
,0x56,0xca,0x53,0x08,0x6f,0x13,0xee,0x1f,0x25,0x2f,0x46,0x28,0x4b,0xd7,0xd
f,0x7d,0x21,0x24,0x9e,0xea,0x08,0xae};

dec :
{0xe3,0x17,0x0b,0x47,0x22,0xd6,0x8b,0xe2,0x82,0xa7,0xea,0xe8,0x45,0x03,0x8
2,0xc0,0xee,0x69,0x19,0x10,0x21,0x83,0xec,0x12,0x99,0xb4,0x9d,0x12,0x5c,0x
cc,0x22,0x4e,0x7d,0xed,0x6e,0x76,0x40,0x04,0x7c,0x22,0xe4,0xd1,0x88,0x04,0
x83,0x92,0x66,0xd0,0x66,0x04,0xb8,0x19,0x12,0xff,0x45,0x41,0x47,0x86,0x70,
0x72,0x07,0x66,0x8c,0x0a,0x8b,0xac,0x60,0x1e,0xe2,0xc8,0x3b,0x1a,0x47,0xdb
,0xb4,0x0b,0x87,0xc5,0xbf,0x0e,0xfb,0xc2,0xa6,0x40,0xf7,0x8e,0x73,0xa6,0x7
0,0xe5,0xfb,0x23,0x61,0x5b,0xbf,0xca};

rec :
{0xe1,0xae,0xd5,0xb4,0x2d,0x11,0x54,0x37,0x6c,0x38,0x25,0x98,0xd9,0x21,0xa
1,0x46,0x84,0xec,0xac,0xd6,0x7e,0x8a,0xa6,0x3b,0x20,0x7e,0x8f,0xce,0x06,0x
a3,0xe7,0x63,0xc8,0x12,0x13,0x08,0x40,0x52,0x73,0xfd,0x08,0x83,0x88,0x35,0
x58,0xc7,0xc8,0xe1,0xf4,0x1b,0xa5,0x7d,0xe1,0x26,0x64,0x85,0x7d,0x9d,0x77,
0x72,0x3b,0xf0,0xb0,0x66,0xbb,0x0a,0x57,0x9d,0xbe,0x25,0x63,0x9a,0x39,0x21
,0x64,0x3b,0x6c,0x58,0x63,0xae,0x21,0x67,0xa2,0x0f,0x3d,0x6e,0x30,0x40,0x0
e,0x37,0x3f,0xe2,0xd2,0x8f,0x5d,0x84};

DEcode(rec):
{0x60,0x29,0x18,0xe2,0x6f,0xb1,0xdd,0x24,0x2d,0x0a,0xca,0xd2,0x0e,0xba,0xd
2,0xde,0x8a,0x91,0x06,0xb6,0xa3,0xc4,0x35,0x27,0xc5,0xfc,0xff,0x7a,0x6a,0x
2c,0xbd,0x2c,0x16,0xc2,0xc3,0x08,0xe9,0xda,0x84,0xb3,0x04,0xad,0xf9,0x69,0
x39,0xc9,0x94,0xfa,0xda,0xc6,0x08,0xab,0xb4,0xad,0x2d,0x1d,0x87,0xd4,0x80,
0x2f,0xa6,0x72,0x2b,0x2e,0xec,0x47,0x4a,0x7b,0x53,0xa9,0x24,0x7e,0xea,0x58
,0x38,0x60,0x45,0xd8,0xc8,0xb3,0xc0};

제 4절 VP-MAC based on WF – LEA

4.1 Length = 79-Byte

Key
={0xdf,0x41,0xa6,0x11,0x9d,0x87,0xbb,0x1a,0xb8,0x1c,0xf3,0x06,0x00,0x70,0x
94,0x2a};

IV
={0x7d,0xf4,0x91,0xe6,0x16,0x60,0xd8,0x67,0xaf,0x5d,0xca,0xce,0xd3,0x5e,0x
80,0xe9};

```
BeSeed =
{0xf2,0x2d,0xa7,0x8f,0xba,0x7d,0xe2,0xa0,0x98,0x37,0x40,0xef,0x5f,0x29,0xa
f,0xe6,0x47,0xdd,0x14,0xe4,0x0d,0x1a,0x3a,0x71,0xe0,0xc3,0xe9,0x6e,0xa1,0x
c9,0x84,0xff};

AinSeed =
{0xfa,0x44,0x7b,0x3d,0x20,0x6b,0x0d,0xde,0xdd,0x8e,0x87,0x74,0x0c,0xfd,0x1
d,0x65,0xb0,0x23,0x15,0xec,0xe9,0x6d,0x6b,0xdb,0x87,0xf0,0xf5,0xb1,0x24,0x
f6,0x8d,0x29};

BinSeed =
{0x25,0xc8,0xe1,0x60,0xcf,0x34,0x70,0x6d,0x2a,0x2a,0x1b,0x5a,0xa8,0x8e,0x9
5,0xcb,0x80,0xb8,0xfb,0xf0,0xf0,0x2a,0x95,0x8b,0x07,0xa3,0xde,0x80,0xb5,0x
f2,0x17,0xca};

wbe_dat =
{0x22,0x1f,0x18,0xbf,0x0d,0x74,0xb5,0xdc,0x38,0x6e,0x23,0xb7,0x09,0x45,0x4
6,0x9e,0x29,0x2e,0xd1,0x1d,0xcc,0x27,0xc6,0x7f,0x8f,0x73,0xda,0xa9,0x26,0x
8b,0x7f,0xf2,0x11,0xc3,0xbc,0x2e,0xc3,0x9d,0x63,0x6c,0x7a,0xfb,0xb4,0xcd,0
x05,0x3c,0xd0,0x8e,0xc6,0x43,0xd7,0xf9,0x2f,0x4b,0xb6,0xa7,0xaf,0x6e,0x18,
0x96,0xe4,0x62,0x15,0xfd,0x51,0x08,0x01,0x28,0x46,0x4f,0xf2,0xc3,0xfb,0xb4
,0x46,0x90,0x2e,0xdb,0xc1};

tag =
{0xd3,0x62,0xc6,0x22,0x03,0x9c,0xca,0x77,0xaf,0x6f,0xfb,0x56,0x79,0x59,0xf
6,0x4a};
```

4.2 Length = 80-Byte

```
key =
{0xd5,0xbe,0xc4,0x61,0xdf,0xdd,0xac,0xf7,0x98,0x06,0x6d,0xb6,0x4e,0x65,0xc
3,0x6e};

IV =
{0xe5,0xaf,0x0b,0x70,0xe7,0x2c,0x79,0x02,0xb5,0x4d,0xab,0x9b,0xae,0x26,0x2
2,0xd1};

BeSeed =
{0x56,0x44,0x6e,0xb4,0x8d,0x6d,0x33,0x14,0x21,0xe9,0x53,0xba,0x88,0xe1,0x6
9,0x61,0x6e,0x30,0xf7,0x80,0x12,0x04,0x81,0x5c,0xf6,0xbd,0xf3,0xe9,0xc4,0x
c3,0xa1,0xd8};

AinSeed =
{0x5d,0xf2,0x71,0x85,0xc7,0x86,0x32,0xea,0x47,0x62,0x09,0x52,0x5a,0x1e,0x9
f,0x31,0x67,0x95,0xd8,0xe2,0x89,0x1e,0xbe,0x73,0xb5,0x7c,0xf8,0x6f,0xa7,0x
66,0x53,0x8b};

BinSeed =
{0x6a,0x9b,0xbb,0x49,0x27,0xb6,0x52,0xb0,0x41,0x5a,0x9c,0x02,0x1b,0xf3,0x3
7,0x8d,0x5e,0xaf,0x8e,0x28,0xe6,0x83,0x6d,0x8c,0x6d,0xf6,0x0d,0x66,0x40,0x
b4,0x7e,0xd1};
```

```
wbe_dat =
{0x79,0x4e,0x1f,0x61,0xcb,0xd9,0x5c,0x27,0xe5,0x68,0x70,0x1a,0x2c,0x21,0x8
e,0x53,0x65,0x9b,0xaf,0x43,0x3a,0xde,0xd8,0xf7,0x73,0xd2,0x16,0x20,0x81,0x
1c,0xce,0x4b,0xd5,0xd1,0x3d,0x11,0xc4,0xc1,0x74,0xb8,0x3f,0xe5,0x8c,0xd5,0
x34,0xcd,0x3e,0x24,0x49,0x12,0xea,0xcd,0xe7,0x09,0xd3,0xe2,0x07,0xe9,0x16,
0x6c,0x28,0x64,0x64,0xbf,0x94,0xf7,0xa4,0xa3,0x66,0x36,0xe0,0x10,0x59,0xcf
,0xe5,0x43,0xb8,0x08,0x84,0x9b};

tag =
{0x84,0xe4,0x53,0x63,0x4e,0xb5,0xb1,0x78,0x0a,0x1b,0x2e,0x21,0xc1,0x45,0x0
d,0xe1};
```

4.3 Length = 81-Byte

```
key =
{0x45,0xd0,0x60,0x08,0x07,0x61,0x2d,0xe2,0xd2,0xe5,0xf3,0x0e,0x1b,0x86,0xf
4,0xe7};

IV =
{0x60,0xbf,0x91,0xb1,0xde,0x47,0x52,0x60,0x43,0xa5,0x5c,0x06,0x34,0xa1,0xc
1,0xfd};

BeSeed =
{0x42,0xa8,0x62,0x70,0x61,0x3e,0xd6,0x63,0x3e,0xf3,0x40,0x39,0xec,0x08,0x8
5,0x51,0x56,0x65,0xdb,0x35,0xcf,0x77,0x32,0xee,0x3f,0x6c,0xc6,0xae,0xc2,0x
ca,0xeb,0x6d};

AinSeed =
{0xd8,0xf3,0xfe,0x3b,0x6d,0x7a,0x19,0xb9,0xd6,0xee,0xcc,0x6c,0xa8,0x39,0x1
e,0x1b,0xbd,0x2e,0x06,0xea,0x8d,0x85,0xb5,0xa4,0xe2,0x9a,0x9d,0xd8,0x88,0x
94,0xf1,0xcf};

BinSeed =
{0x0d,0xb7,0x80,0xbd,0xcd,0xdf,0xef,0x3c,0x6e,0x27,0x60,0x0f,0x41,0x3f,0x9
5,0xd7,0x6c,0x2a,0x10,0x96,0x3e,0xe4,0x19,0x38,0xf2,0x3d,0x35,0x0e,0x29,0x
76,0xe9,0x63};

wbe_dat =
{0xfd,0x7f,0x4b,0x76,0x06,0xb5,0x9c,0x7d,0xd9,0x78,0x8b,0xa6,0x07,0x32,0x5
8,0x11,0x89,0xf2,0x4c,0x5b,0x7e,0x63,0x1d,0x1a,0x3b,0x20,0xc5,0x91,0x30,0x
c4,0x26,0xd0,0x75,0x29,0x51,0x12,0x13,0x12,0x1c,0x0e,0xf3,0xc4,0x24,0x3d,0
xb3,0x4e,0x5a,0x80,0x87,0xf4,0xba,0x28,0x45,0x44,0x6f,0x2e,0xc7,0x98,0xda,
0xa1,0x3d,0x8c,0x6e,0xad,0x5a,0x15,0x08,0x2a,0x3f,0x4d,0xd4,0x52,0x6b,0xb8
,0x82,0x14,0x21,0x10,0x23,0x17,0x6f};

tag =
{0xf1,0x68,0x71,0x7e,0x09,0xe0,0x88,0xe1,0xc7,0xe5,0xdc,0x74,0x9a,0x81,0x2
4,0xbb};
```

4.4 Length = 161-Byte

```
key =
{0x30,0x21,0x0b,0xd7,0xd1,0x0f,0xa6,0xbc,0x7d,0xb9,0x37,0xbe,0x0f,0x21,0x3
c,0x01};

IV =
{0x0e,0xac,0xd4,0x37,0x3d,0x48,0x7c,0x0c,0x53,0x75,0x7e,0x57,0xa8,0x81,0x3
0,0x9d};

BeSeed =
{0x82,0x15,0x13,0x23,0xef,0x4c,0xe9,0x68,0xf9,0xb4,0x5d,0x03,0x6c,0xc9,0xd
5,0xff,0xa9,0x1d,0x76,0xab,0x35,0x89,0xb6,0x79,0xaa,0xc3,0xd0,0x7a,0x4b,0x
b2,0x4b,0xef};

AinSeed =
{0x84,0xc4,0x0a,0x5c,0x0d,0x52,0xa3,0xf3,0x13,0x0d,0x41,0xfa,0x4a,0x84,0xf
d,0x62,0xf5,0xcc,0x0c,0x0c,0xe9,0xc6,0x46,0xbd,0xa1,0x0f,0xb5,0x20,0x73,0x
dd,0x99,0x5f};

BinSeed =
{0x8b,0x00,0x86,0x4a,0xcc,0x0f,0xff,0x53,0x58,0x6d,0x43,0xdd,0x47,0x5e,0xc
c,0x5e,0x9d,0xc1,0x70,0x8b,0xf7,0xd5,0x51,0xbb,0xec,0x45,0x1b,0x9b,0x11,0x
49,0x42,0xee};

wbe_dat =
{0x2d,0x47,0x7a,0xdd,0x9f,0x44,0xb0,0x79,0xa3,0x04,0x4d,0x64,0xf3,0x32,0xc
7,0xa9,0xcb,0x00,0xd7,0x06,0x79,0x43,0x6c,0x78,0xf5,0x9e,0x11,0x3e,0x0c,0x
b5,0x26,0x01,0xf6,0x89,0xa6,0x05,0x62,0x76,0xcd,0x0a,0x8b,0x97,0xa8,0x28,0
xc9,0x98,0xad,0x27,0x86,0xbe,0x93,0xed,0x7f,0xdd,0x1a,0x17,0x33,0x03,0x1c,
0xc1,0x74,0x35,0x1f,0x7c,0xc2,0xaa,0xbe,0x6e,0xa7,0x58,0xbf,0x3c,0xad,0x2e
,0x6e,0xbe,0x4e,0x2e,0xa4,0xe8,0x1c,0x8f,0xd1,0x0c,0xda,0x1a,0x13,0xd6,0x1
9,0xe7,0xc5,0xf4,0x98,0x11,0xe8,0xb6,0x7c,0x89,0x5d,0x6f,0xd3,0x7f,0x75,0x
46,0xff,0xac,0x28,0x48,0x15,0x61,0x45,0xec,0x8a,0xfa,0x8f,0x2a,0x5e,0x12,0
x6b,0x4f,0x88,0x36,0x9d,0xea,0xad,0xb3,0xf0,0x4b,0x72,0x25,0xfa,0x26,0xd9,
0xd5,0x3d,0xed,0xf5,0x0c,0x60,0xf1,0xe2,0x97,0xea,0x09,0x84,0x5c,0xe1,0xf5
,0xcc,0x87,0xdf,0x71,0x42,0xa1,0x52,0xec,0x49,0xbe,0x7c,0x24,0xa4};

tag =
{0xc0,0x01,0x47,0x98,0x94,0xd4,0xab,0xff,0x34,0xdf,0x65,0xff,0x90,0xd5,0xb
6,0xc9};
```

4.5 32 바이트 시드로 4 비트 단위 16 바이트 치환 생성하기

```
[Seed]
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:

[random 512-byte derived from seed]
8b:50:ca:ed:e1:fe:2e:d4:d8:ff:29:d3:d6:80:9a:1f:
```

```
9c:32:ec:3a:cc:f9:37:97:4a:0f:32:5c:cc:7e:71:7a:
27:4a:f5:86:6e:15:c2:92:89:31:d0:08:5f:2e:89:c8:
e0:47:73:cd:ec:c2:d1:d5:62:9e:36:76:fc:96:3d:b9:
4e:fe:ea:93:da:f1:5d:ca:0b:a6:88:df:4c:4b:03:14:
6c:4f:7b:c0:04:7b:56:9e:73:4b:fc:11:86:b9:3f:c6:
78:95:ea:7e:1d:07:68:4b:e8:ee:32:8b:70:d8:6a:c4:
a6:0e:7d:66:70:79:21:e3:ff:9d:3f:13:8a:35:cb:7e:
88:cb:18:87:3a:ec:18:f3:8e:9a:13:20:e7:d9:61:b7:
12:b5:c8:59:36:a8:45:60:35:13:74:3e:40:d8:f2:65:
2e:af:e7:db:7a:dd:72:12:f5:2f:b3:03:21:8c:89:4b:
8a:35:9a:75:d9:c3:73:af:db:db:f7:af:f8:8a:b8:7e:
12:bd:ef:e4:93:15:7f:c2:2c:84:01:98:a0:80:a4:60:
f7:d3:6e:d0:92:8f:c1:4e:a8:ed:cd:95:52:44:5c:22:
6d:9c:9f:19:5f:c7:7a:51:14:f7:14:b8:17:96:f6:a7:
85:4a:d6:e9:c7:26:08:88:ed:ae:e0:d6:e1:a1:b6:23:
41:62:67:a8:25:ee:c4:02:61:60:6b:42:f1:5d:a3:27:
f9:54:dc:ce:ec:f0:bc:71:a2:e3:9f:d1:0d:7c:15:da:
a3:87:6c:88:65:86:87:be:a2:61:14:89:c4:1c:06:64:
81:2f:4e:56:22:63:f8:f7:82:0d:d0:7f:bf:86:d1:85:
43:e9:b8:8d:7c:d7:fc:d1:9c:fb:b4:e4:6e:79:37:ef:
4d:50:46:05:01:e5:ed:f3:d2:8d:e7:06:8c:3a:f0:2c:
41:c7:cc:aa:db:10:f8:88:2b:82:2f:dd:34:a1:6a:f2:
a0:c1:4d:df:62:a2:bf:4d:ec:73:bb:e9:44:63:ac:77:
c5:d5:47:d0:1d:e6:f9:a6:eb:50:43:d9:05:99:18:3e:
8a:28:6d:33:b8:26:85:f3:0f:26:65:f3:b2:ce:4c:16:
dc:46:fd:fe:82:ec:2b:84:5f:e4:3a:23:91:b0:e2:f3:
5b:6c:c1:f7:b7:78:4e:84:5d:32:97:02:86:6b:35:a8:
77:21:27:c6:e6:b9:af:77:8f:45:b3:43:7c:c4:56:05:
dd:6f:e3:fc:27:1b:2a:5f:4c:b1:d0:b9:60:f7:66:ae:
71:1a:3b:6d:dd:fe:e9:93:a5:45:38:4d:94:20:ae:ab:
af:8c:16:ab:e8:a6:24:19:3a:02:2d:60:ca:8a:7f:0b:

[Permutations]
( 0) 11:13: 2: 7:14: 1:12: 4: 8: 5: 9: 3: 6: 0:10:15:
( 1)  8: 6: 0:15: 4:14: 3: 9: 5:11: 2:13:12: 7: 1:10:
( 2) 11:10:14: 6: 3: 5:12: 2: 4: 1: 0:15:13: 7: 9: 8:
( 3)  0:15: 3:10: 4: 8: 1: 5: 2: 7:11:14:12: 6:13: 9:
( 4)  9: 0:15:14:13: 1:10: 2: 5: 6: 8: 7:12:11: 3: 4:
( 5) 13: 2:11: 0:10: 8:14:12: 3: 5: 4: 1:15: 9: 7: 6:
( 6)  6: 1:15:14: 5: 9:12: 3:13: 7: 2:11: 0: 8:10: 4:
( 7)  9: 4:13: 2: 0: 8: 1:15: 6:12: 7: 3:10: 5:11:14:
( 8)  5:14:13: 8: 2: 4:11:10:12: 6: 3: 0:15: 9: 1: 7:
( 9) 14:10: 6: 1: 9:11:13:12:15: 3: 4: 7: 0: 8: 2: 5:
(10)  0: 4:13:10: 6: 8:14: 2: 5: 7:15: 3: 1:12: 9:11:
(11)  0: 5: 2: 4: 1:12: 3:13: 9: 6:11: 7:15:10: 8:14:
(12)  9: 7:10:12:11: 5: 3: 2: 6:14: 1: 8:13:15: 4: 0:
(13) 11: 7:13: 0:14: 3: 1: 9: 8:10: 6: 5:15: 4:12: 2:
```

```
(14) 13: 0:11: 9: 5: 3: 2: 1:10:12: 4: 8:15:14: 6: 7:
(15)  4: 8: 2:12: 5:15: 9:10:11: 7: 0:14:13: 1: 6: 3:
(16)  6:10: 8: 9:15:14: 4:11:12: 0: 5: 2: 1:13: 3: 7:
(17) 15: 9:14: 8:13: 0: 4:11: 2: 3: 7: 1: 6:12: 5:10:
(18) 10:11: 5: 0: 8: 7: 3:14: 2: 1:13: 9:15:12: 6: 4:
(19)  9:14:15: 4: 8: 3:10:11: 2:13: 0:12: 7: 6: 1: 5:
(20)  2:13: 0: 8: 6: 3:10: 1:12: 5:11: 4:14: 9: 7:15:
(21) 13:14: 8: 4: 1: 9: 5: 3: 2:11: 7: 6:15:10: 0:12:
(22) 12:13: 8: 9: 3: 6:14: 0: 5:15:11: 7: 4: 1:10: 2:
(23)  0:14: 1: 8:10: 2:11: 5: 6:13:15: 9: 4: 3:12: 7:
(24)  4: 2: 7:11:15:10: 1: 6:12: 0: 3:13: 5: 9: 8:14:
(25) 14: 4: 1: 9: 0: 8:10:11:13:15: 5: 3: 2: 7:12: 6:
(26)  6:13:12: 5:14: 8:11: 9: 7: 4:10:15: 1: 0: 2: 3:
(27)  9:14: 1:15: 3: 0:12: 4:10:13: 7: 2: 6:11: 5: 8:
(28)  0:13: 9: 2:14: 1:10: 8: 7:15:11: 3:12: 4: 6: 5:
(29)  4: 8:11:10: 5: 3: 2:13:15: 1:12: 9: 0: 7: 6:14:
(30)  2:13:12:10: 7:15: 1: 3: 9: 5: 8: 6: 4: 0:14:11:
(31)  8:12: 9: 5:15: 3: 4: 1:14: 2: 6: 0:13:10: 7:11:
```

부록 B 블록암호 LEA 키스케줄

블록암호 키스케줄은 다음 8개의 32비트 상수 $\delta[0], \delta[1], \ldots, \delta[7] \in \{0,1\}^{32}$을 사용한다.

$\delta[0]$ = 0xc3efe9db, $\delta[1]$ = 0x44626b02, $\delta[2]$ = 0x79e27c8a, $\delta[3]$ = 0x78df30ec,

$\delta[4]$ = 0x715ea49e, $\delta[5]$ = 0xc785da0a, $\delta[6]$ = 0xe04ef22a, $\delta[7]$ = 0xe5c40957

Memo 8.1. 'L', 'E', 'A' 의 ASCII 코드에 대응하는 10진수 정수 76, 69, 65이다. 8개의 32비트열 $\delta[0], \delta[1], \ldots, \delta[7] \in \{0,1\}^{32}$는 $\sqrt{766965} - \lfloor\sqrt{766965}\rfloor$의 상위 256비트 값이다.

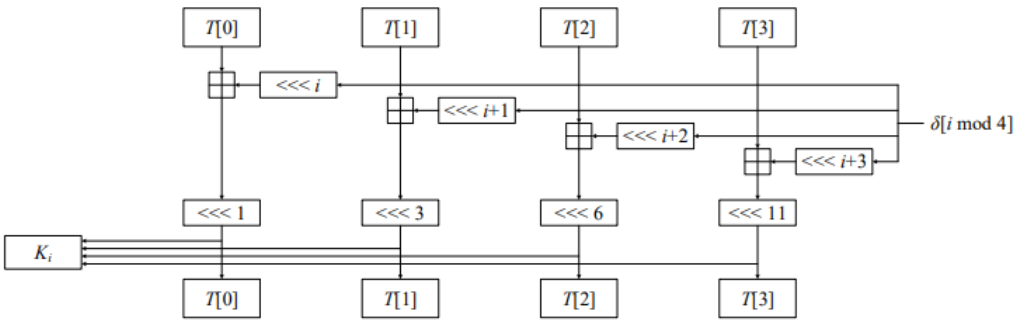
제 1절 LEA128

키스케줄 함수는 128비트 키를 입력받아 24개의 192비트 라운드키를 반환한다.
알고리듬 34는 키스케줄 함수 유사부호이다.

알고리듬 34 LEA128: 키스케줄

Input: 128-bit Key $K = K[0]\|K[1]\|K[2]\|K[3]$
Output: 24 192-bit RoundKeys $K_i = K_i[0]\|\cdots\|K_i[5]$ for $i = 0, 1, \ldots, 23$

procedure (KeySchedule(K))
  $T[0]\|T[1]\|T[2]\|T[3] \leftarrow K[0]\|K[1]\|K[2]\|K[3]$
  for $i = 0$ to 23 **do**
    $T[0] \leftarrow (T[0] \boxplus (\delta[i \bmod 4] \lll (i+0))) \lll 1$
    $T[1] \leftarrow (T[1] \boxplus (\delta[i \bmod 4] \lll (i+1))) \lll 3$
    $T[2] \leftarrow (T[2] \boxplus (\delta[i \bmod 4] \lll (i+2))) \lll 6$
    $T[3] \leftarrow (T[3] \boxplus (\delta[i \bmod 4] \lll (i+3))) \lll 11$
    $K_i \leftarrow T[0]\|T[1]\|T[2]\|T[1]\|T[3]\|T[1]$
  end for
  return $K_0, K_1, \ldots, K_{23}$
end procedure



테스트벡터

```
[test: LEA128 encryption]
block bit length = 128
key bit length = 128

[key]
0f:1e:2d:3c:4b:5a:69:78:87:96:a5:b4:c3:d2:e1:f0:

[roundkeys]
[00] 003a0fd4:02497010:194f7db1:02497010:090d0883:02497010:
[01] 11fdcbb1:9e98e0c8:18b570cf:9e98e0c8:9dc53a79:9e98e0c8:
[02] f30f7bb5:6d6628db:b74e5dad:6d6628db:a65e46d0:6d6628db:
[03] 74120631:dac9bd17:cd1ecf34:dac9bd17:540f76f1:dac9bd17:
[04] 662147db:c637c47a:46518932:c637c47a:23269260:c637c47a:
[05] e4dd5047:f694285e:e1c2951d:f694285e:8ca5242c:f694285e:
[06] baf8e5ca:3e936cd7:0fc7e5b1:3e936cd7:f1c8fa8c:3e936cd7:
[07] 5522b80c:ee22ca78:8a6fa8b3:ee22ca78:65637b74:ee22ca78:
[08] 8a19279e:6fb40ffe:85c5f092:6fb40ffe:92cc9f25:6fb40ffe:
[09] 9dde584c:cb00c87f:4780ad66:cb00c87f:e61b5dcb:cb00c87f:
[10] 4fa10466:f728e276:d255411b:f728e276:656839ad:f728e276:
[11] 9250d058:51bd501f:1cb40dae:51bd501f:1abf218d:51bd501f:
[12] 21dd192d:77c644e2:cabfaa45:77c644e2:681c207d:77c644e2:
[13] de7ac372:9436afd0:10331d80:9436afd0:f326fe98:9436afd0:
[14] fb3ac3d4:93df660e:2f65d8a3:93df660e:df92e761:93df660e:
[15] 27620087:265ef76e:4fb29864:265ef76e:2656ed1a:265ef76e:
[16] 227b88ec:d0b3fa6f:c86a08fd:d0b3fa6f:a864cba9:d0b3fa6f:
[17] f1002361:e5e85fc3:1f0b0408:e5e85fc3:488e7ac4:e5e85fc3:
[18] c65415d5:51e176b6:eca88bf9:51e176b6:edb89ece:51e176b6:
[19] 9b6fb99c:0548254b:8de9f7c2:0548254b:b6b4d146:0548254b:
[20] 7257f134:06051a42:36bcef01:06051a42:b649d524:06051a42:
[21] a540fb03:34b196e6:f7c80dad:34b196e6:71bc7dc4:34b196e6:
[22] 8fbee745:cf744123:907c0a60:cf744123:8215ec35:cf744123:
[23] 0bf6adba:df69029d:5b72305a:df69029d:cb47c19f:df69029d:
```

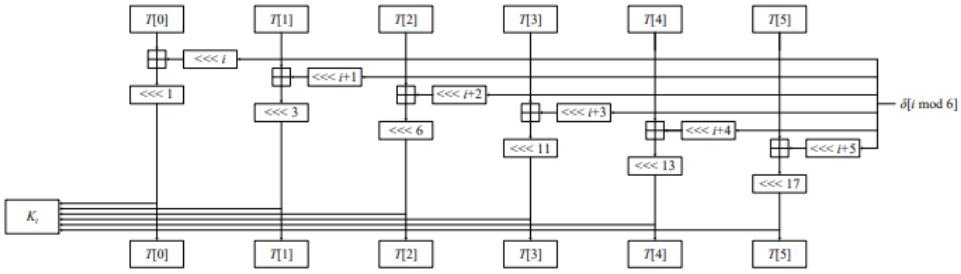
제 2절 LEA192

키스케줄 함수는 192비트 키를 입력받아 28개의 192비트 라운드키를 반환한다.
알고리듬 35는 키스케줄 함수 유사부호이다.

알고리듬 35 LEA192: 키스케줄

Input: 192-bit Key $K = K[0]\|K[1]\|K[2]\|K[3]\|K[4]\|K[5]$
Output: 28 192-bit RoundKeys $K_i = K_i[0]\|\cdots\|K_i[5]$ for $i = 0, 1, \ldots, 27$

procedure (KeySchedule(K))
  $T[0]\|T[1]\|T[2]\|T[3]\|T[4]\|T[5] \leftarrow K[0]\|K[1]\|K[2]\|K[3]\|K[4]\|K[5]$
  for $i = 0$ to 27 **do**
    $T[0] \leftarrow (T[0] \boxplus (\delta[i \bmod 6] \lll (i+0))) \lll 1$
    $T[1] \leftarrow (T[1] \boxplus (\delta[i \bmod 6] \lll (i+1))) \lll 3$
    $T[2] \leftarrow (T[2] \boxplus (\delta[i \bmod 6] \lll (i+2))) \lll 6$
    $T[3] \leftarrow (T[3] \boxplus (\delta[i \bmod 6] \lll (i+3))) \lll 11$
    $T[4] \leftarrow (T[4] \boxplus (\delta[i \bmod 6] \lll (i+4))) \lll 13$
    $T[5] \leftarrow (T[5] \boxplus (\delta[i \bmod 6] \lll (i+5))) \lll 17$
    $K_i \leftarrow T[0]\|T[1]\|T[2]\|T[3]\|T[4]\|T[5]$
  end for
  return $K_0, K_1, \ldots, K_{27}$
end procedure



테스트벡터

```
[test: LEA192 encryption]
block bit length = 128
key bit length = 192

[key]
0f:1e:2d:3c:4b:5a:69:78:87:96:a5:b4:c3:d2:e1:f0:f0:e1:d2:c3:b4:a5:96:87:

[roundkeys]
[00] 003a0fd4:02497010:194f7db1:090d0883:2ff5805a:c2580b27:
[01] 11fdcbb1:9e98e0c8:18b570cf:9dc53a79:5c145788:9771b5e5:
[02] f30f7bb5:6d6628db:b74e5dad:a65e46d0:6f44da96:f643115f:
[03] 74120631:dac9bd17:cd1ecf34:540f76f1:aa1a5bdb:fbafaae7:
[04] 13f8a031:34f28728:31fdb409:0e31481b:df498117:cf9371f1:
[05] 0967c312:b3484ec8:3aae5b3d:5a9714a0:b2d4dd5f:3a1fcdf7:
[06] 0ac47404:59e9e54d:a60dc00a:566139d3:898dce4f:582d72dd:
[07] 77f3ea4c:e2a73c8d:b8f1249a:6a172700:bc0e539c:2e46fdbb:
[08] b4e0e98a:3d028c05:b8d3a050:dbd67bef:df675c7a:99eefbb0:
[09] e68584f6:ce31ef45:96c105ac:2a1be677:9d72b8b0:33cecc54:
[10] c22ffd76:1ab7167e:42bb3060:7da517f5:4aa0e8d3:0a070c3c:
[11] e200a765:c2be17b3:7f22543f:3e4eb7a1:c992a6f4:a783c823:
[12] c13cc747:ffcc8185:66514e9e:e4ccc199:cd5c766d:a004f676:
[13] 1d3a1fa6:d46894ec:f49c33e6:782fda7e:1fe6346c:0ffe981c:
```

```
[14] 78b97c3d:956e8ee8:49ab721c:2672138a:037ea242:ce5fe8a4:
[15] 225f7158:32d83e3e:e118f6aa:1fb83751:4d27715c:ed2fba4e:
[16] 8dfbc56d:e0a907db:e4af091c:5e123225:d0e8d2e1:cc4501fb:
[17] 8422a8f0:46a12f92:415152ad:f55417f5:38738248:c6e29ded:
[18] 5723715e:abfa788c:c3646af7:64af9186:8fc855ec:2bc36989:
[19] 5e6b28e3:e0f5f592:eb3dd108:0551012a:50e4221d:97e85c0f:
[20] 4e258e14:92298f0b:771269c3:6f934254:c0933b6b:421159b8:
[21] d76953f4:6a3e36be:53b656fb:610c22e0:9f399330:acf7e7e9:
[22] fe0b573b:cbb73085:89ed67fc:77014cef:e1b8431f:ba1b4105:
[23] 06de3450:b3f5b2fe:df1cec27:fb22bd10:8e3de6fe:3d4acd27:
[24] c5444873:5bec968b:8b2af393:11e2f6ca:9cb3694f:94c56b91:
[25] 939a1a93:27f101bb:5381bae7:48ebd1b1:f6d5fca7:0ca24bbc:
[26] 7b03490b:de00acfb:c7f8abfe:410a14c1:d37932a9:14029327:
[27] bd948525:2c75004d:c52486d5:0f07e2fa:1963e1fd:882719c3:
```

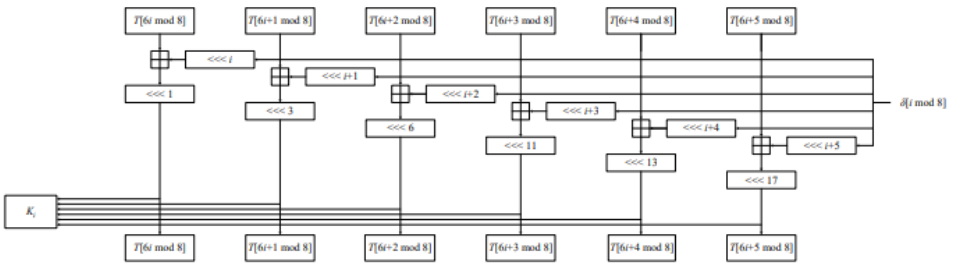
제 3절 LEA256

키스케줄 함수는 256 비트 키를 입력받아 32 개의 192 비트 라운드키를 반환한다. 알고리듬 36 은 키스케줄 함수 유사부호이다.

알고리듬 36 LEA256: 키스케줄

Input: 256-bit Key $K = K[0]\|K[1]\|K[2]\|K[3]\|K[4]\|K[5]\|K[6]\|K[7]$
Output: 32 192-bit RoundKeys $K_i = K_i[0]\|\cdots\|K_i[5]$ for $i = 0, 1, \ldots, 31$

procedure (KeySchedule(K))
  $T[0]\|T[1]\|T[2]\|T[3]\|T[4]\|T[5]\|T[6]\|T[7] \leftarrow K[0]\|K[1]\|K[2]\|K[3]\|K[4]\|K[5]\|K[6]\|K[7]$
  for $i = 0$ to 31 **do**
    $T[6i+0 \bmod 8] \leftarrow (T[6i+0 \bmod 8] \boxplus (\delta[i \bmod 8] \lll (i+0))) \lll 1$
    $T[6i+1 \bmod 8] \leftarrow (T[6i+1 \bmod 8] \boxplus (\delta[i \bmod 8] \lll (i+1))) \lll 3$
    $T[6i+2 \bmod 8] \leftarrow (T[6i+2 \bmod 8] \boxplus (\delta[i \bmod 8] \lll (i+2))) \lll 6$
    $T[6i+3 \bmod 8] \leftarrow (T[6i+3 \bmod 8] \boxplus (\delta[i \bmod 8] \lll (i+3))) \lll 11$
    $T[6i+4 \bmod 8] \leftarrow (T[6i+4 \bmod 8] \boxplus (\delta[i \bmod 8] \lll (i+4))) \lll 13$
    $T[6i+5 \bmod 8] \leftarrow (T[6i+5 \bmod 8] \boxplus (\delta[i \bmod 8] \lll (i+5))) \lll 17$
    $K_i \leftarrow T[6i+0 \bmod 8]\|T[6i+1 \bmod 8]\|T[6i+2 \bmod 8]\|T[6i+3 \bmod 8]\|T[6i+4] \bmod 8]\|T[6i+5 \bmod 8]$
  end for
  return $K_0, K_1, \ldots, K_{31}$
end procedure



테스트벡터

```
[test: LEA256 encryption]
block bit length = 128
key bit length = 256

[key]
0f:1e:2d:3c:4b:5a:69:78:87:96:a5:b4:c3:d2:e1:f0:f0:e1:d2:c3:b4:a5:96:87:78
:69:5a:4b:3c:2d:1e:0f:

[roundkeys]
[00] 003a0fd4:02497010:194f7db1:090d0883:2ff5805a:c2580b27:
[01] a83e7ef9:053eca29:d359f988:8101a243:9bbf34b3:9228434f:
[02] 2efee506:8b5f7bd4:9991e811:72dbc20c:2384c97f:cefee47f:
[03] c571782c:00da90b1:b940a552:5db79619:4bc9a125:5d08a419:
[04] 72de26cc:d69bc26f:46a7f207:66ff4d81:a87862fc:a5f63601:
[05] 7909c4fa:f3f93651:72cb0bcd:ae69b2e3:80f2ca4b:f13efcce:
[06] 7869db69:6b7a5b8e:fefbf6b1:ec608c8e:76e9d5d2:13ca4bf6:
[07] c5eeec7a:aa42a59d:1f22cd00:fdd92bdc:d6bbe3e8:15d459ec:
[08] cda7632a:9cf01bef:6596e261:8c1de14c:1127c3b8:48b3f629:
[09] 3723d0e1:fc0317ec:3fdd5378:0201ae1d:e55db65e:e4c84dbc:
[10] 3633db3f:e4c24fc2:bb1e1fd7:a339425c:fe3e1bdf:d61c808d:
[11] bdca3449:beb8aa4e:145a9687:eb6fcd87:8b88ca72:7677a84b:
[12] d11005e9:558275c5:bc742819:3f17e888:20fcb71f:60886959:
[13] 8d9446c4:67d2d167:855a6aef:69ea517c:36e48e11:0d3f4e86:
[14] bb0ede65:cceecc06:efc9c49f:44902261:bd8549c0:a7e7f682:
[15] 772101e6:b4b9a250:6faa7b73:7318b792:1e57e751:fd43b41c:
[16] 4ec21b5f:dcfbf30b:a4046947:be0e781c:d74e21ac:6b1f5d22:
[17] e8b8e02b:4a662d2d:b50f9ca9:01c98c69:9eb28089:216cfd3f:
[18] 92f0126b:7b9961aa:581f94ac:ab4be6dd:c2a91af5:fb4e8e0c:
[19] 4c2c8f04:81a45991:1fcb946c:bccbb5b5:808899cb:8c1b2f89:
[20] 192061be:78e5cf04:f239ab5c:e8471e86:9e6217c7:e5fdf35c:
[21] 83c3150d:766887f8:a1092ac7:6aa6f41d:16e200f9:6bdc26ca:
[22] 52345706:db70d6af:a8d8ffeb:492ee661:4cd1e991:d75d8352:
[23] 85a9c5fb:1e0f569e:7ff7c600:3f36a1d8:e406ad00:4ded8f16:
[24] 512bb2f4:772b192c:2e6168bd:76af67e1:d893a786:3e276f69:
[25] d11ee3ad:b7f8c612:d3b19318:89fee4db:b6c3aedd:05420f90:
[26] 04f662f0:8fb41a6c:2f42dd5e:a8ad1839:46474e45:46418de0:
[27] 351550c8:668014f6:04924365:5f353d6f:4eba8d76:924a4318:
[28] 5aba711c:a36b1398:5b3e7bf4:7b3a2cf9:1d006ebe:0d5683e5:
[29] 4f56916f:215dccd2:9f57886f:876d1357:46013d49:2a4932a3:
[30] aa285691:ebefe7d3:e960e64b:dd893f0f:6a234412:495d13c9:
[31] 71c683e8:8069dfd0:6c1a501d:00699418:262142f0:a91a7393:
```