# 128비트 블록암호 LEA 규격서



# 목 차

1.	개요	1
2.	구성 및 범위	1
3.	용어 정의 및 약어	1
4.	기호와 표기	3
	4.1. 기호	3
	4.2. 丑기	5
5.	블록암호 LEA	7
	5.1. 규격	7
	5.2. 암호화	7
	5.3. 복호화	2
부	록 I. 참조구현값1	6
•	= = =:	-

# 128비트 블록암호 LEA

#### 1. 개요

블록암호 LEA(Lightweight Encryption Algorithm)는 128비트 데이터 블록을 암호화하는 알고리즘으로 128, 192, 256비트 비밀키를 사용할 수 있으며 요구되는 안전성 기준에 따라 용도가 구분될 수 있다. LEA(Lightweight Encryption Algorithm)의 라운드 함수는 32비트 단위의 ARX(Addition, Rotation, XOR) 연산만으로 구성되어 있어, 이들 연산을 지원하는 범용 32비트 소프트웨어 플랫폼에서 고속으로 동작한다. 또한 라운드 함수 내부의 ARX(Addition, Rotation, XOR) 연산 배치는 충분한 안전성을 보장하는 것과 동시에 S-box의 사용을 배제하여 경량 구현이 가능하도록 한다.

#### 2. 구성 및 범위

본 규격서는 128비트 블록암호 LEA의 전체 구조와 내부를 구성하는 라운드 함수, 그리고 암·복호화 과정에서 라운드 함수의 비밀 요소로 작용하는 라운드키 생성 과정에 대해 기술한다.

#### 3. 용어 정의 및 약어

#### 3.1. 용어 정의

#### 3.1.1. 내부상태

암호화, 복호화 또는 키 스케줄 수행 중에 나타나는 중간값

#### 3.1.2. 라운드키

키 스케줄을 통하여 생성되는 값. 암호화 및 복호화 함수에 적용됨

#### 3.1.3. 라운드 함수

암호화 또는 복호화 함수에서 반복 수행되는 함수

#### 3.1.4. 복호화(과정)

비밀키를 이용하여 암호문을 평문으로 바꾸는 일련의 변환으로, 복호화 함수와 복호화 키 스케줄로 구성됨

#### 3.1.5. 복호화 함수

라운드키들과 암호문을 입력받아 평문을 출력하는 함수

#### 3.1.6. 블록

고정된 길이의 비트열

#### 3.1.7. 블록암호

비밀키를 이용하여 고정된 길이의 데이터 블록을 암호화하거나 복호화하는 알고 리즘

#### 3.1.8. 비밀키

암호화 또는 복호화에 사용되는 비밀 정보. 키 스케줄의 입력

#### 3.1.9. 암호문

평문으로부터 암호화된 정보

#### 3.1.10. 암호화 (과정)

비밀키를 이용하여 평문을 암호문으로 바꾸는 일련의 변환으로, 암호화 함수와 암호화 키 스케줄로 구성됨

#### 3.1.11. 암호화 함수

라운드키들과 평문을 입력받아 암호문을 출력하는 함수

#### 3.1.12. 키 스케줄

비밀키를 입력 받아 라운드키들을 생성하는 과정

#### 3.1.13. 평문

암호화되지 않은 원래의 정보

#### 3.1.14. 형(型, Type)

변수의 종류. 비트, 바이트, 또는 워드

#### 3.2. 약어

ARX Addition, Rotation, XOR

LEA Lightweight Encryption Algorithm

#### 4. 기호와 표기

#### 4.1. 기호

정수 a와 정수 n(>0)에 대하여 a를 n으로 나눈 나머지 a mod n 32비트열  $x = x_{31} || x_{30} || \cdots || x_0$  입력에 대하여 BitToInt(x) 정수  $n = x_0 \cdot 2^0 + x_1 \cdot 2^1 + \dots + x_{31} \cdot 2^{31}$ 을 출력하는 함수 암호문. 바이트 배열 C = (C[0], C[1], ..., C[15])로 표기됨 C 복호화 함수. 키 스케줄에 의해 비밀키로부터 생성된 라운드 Decrypt 키를 이용하여 암호문을 평문으로 복호화함 δ 키 스케줄에 사용되는 상수. 워드 배열  $\delta = (\delta[0], \delta[1], \cdots$ δ[7])로 표기됨 암호화 함수. 키 스케줄에 의해 비밀키로부터 생성된 라운드 Encrypt 키를 이용하여 평문을 암호문으로 암호화함 K 비밀키. 바이트 배열 K=(K[0], K[1], ..., K[Nk-1])로 표기 k 비밀키를 구성하는 비트의 개수. LEA에 대하여 k = 128, 192, 또는 256으로 사용됨 Nb 평문 또는 암호문을 구성하는 바이트의 개수. LEA에 대하여 Nb = 16으로 고정됨 비밀키를 구성하는 바이트의 개수. LEA에 대하여 Nk = 16, Nk 24. 또는 32로 사용됨 Nr 라운드 수. Nk에 따라 결정됨. LEA에 대하여 Nr = 24, 28, 또 는 32로 사용됨

IntToBit(n) 0부터  $2^{32} - 1$ 까지의 정수 n에 대하여,  $n = x_0 \cdot 2^0 + x_1 \cdot 2^1 + x_1 \cdot 2^{10} + x_1 \cdot 2$ 

 $\cdots + x_{31} \cdot 2^{31}$ 을 만족시키는 32비트열  $x = x_{31} \| x_{30} \| \cdots \| x_0$ 을

출력하는 함수

**KeySchedule**<sup>enc</sup> k비트 비밀키 K로부터 암호화에 필요한 라운드키 RK<sup>enc</sup> (0 ≤

i≤(Nr-1))를 생성하는 함수

KeySchedule<sub>i</sub>dec k비트 비밀키 K로부터 복호화에 필요한 라운드키 RK<sub>i</sub>dec (0 ≤

i≤(Nr-1))를 생성하는 함수

P 평문. 바이트 배열 P=(P[0], P[1], ···, P[15])로 표기됨

ROL<sub>i</sub>(x) 32비트 비트열 x의 i비트 좌측 순환이동

ROR<sub>i</sub>(x) 32비트 비트열 x의 i비트 우측 순환이동

Round<sup>enc</sup> 암호화 라운드 함수

Round<sup>dec</sup> 복호화 라운드 함수

RK.enc 암호화 과정의 i번째 라운드 함수에 사용되는 192비트 라운

드키. 워드 배열 RK; = (RK; 0], RK; 1], ..., RK; 5])로

포기됨

RK, dec 복호화 과정의 i번째 라운드 함수에 사용되는 192비트 라운

드키. 워드 배열  $RK_i^{dec} = (RK_i^{dec}[0], RK_i^{dec}[1], \dots, RK_i^{dec}[5])$ 로

프기된

T 키 스케줄의 라운드키 생성에 사용되는 내부상태 변수. 비밀

키와 동일한 길이를 가지며 워드 배열 T=(T[0], T[1], …,

T[Nk/4-1])로 표기됨

X; 암호화 또는 복호화 함수의 i번째 라운드 함수에 입력되거나

(i-1)번째 라운드 함수에서 출력되는 128비트 내부상태 변수. 평문과 동일한 길이를 가지며 워드 배열 X = (X[0], X[1]).

X<sub>i</sub>[2], X<sub>i</sub>[3])로 표기됨

x∥y 두 비트열 x와 y의 연접

x ⊕ y 같은 길이를 갖는 두 비트열 x와 y의 XOR(eXclusive OR)

х田у 두 32비트열 x와 y에 대해 IntToBit(BitToInt(x) + BitToInt(y)

mod 2<sup>32</sup>)로 정의되는 연산

x 日 y 두 32비트열 x와 y에 대해 IntToBit(BitToInt(x) - BitToInt(y)

mod 2<sup>32</sup>)로 정의되는 연산

Y ← X X의 비트열을 Y에 대입시키는 연산

#### 4.2. 표기

#### 4.2.1. 알고리즘 변수 표기

블록암호 LEA의 평문과 암호문은 각각 128비트 길이의 비트열이며, 비밀키는 128, 19 2, 또는 256비트 길이의 비트열이다. LEA의 비밀키, 평문, 암호문은 바이트 배열로 표기되며, 암호화, 복호화 및 키 스케줄 과정에 사용되는 내부상태 변수 및 라운드키는 워드배열로 표기된다.

#### 4.2.2. 입력 비트열의 바이트 배열 변환

본 규격서에서 바이트는 8비트 길이의 비트열을 의미한다. 길이 8n의 입력 비트열 inp ut<sub>0</sub>, input<sub>1</sub>, input<sub>2</sub>, ···, input<sub>8n-1</sub>로부터 바이트 배열 A[0], A[1], A[2], ···, A[n-1]은 다음 과 같이 주어지며, 순서는 <표 4-1>과 같다.

 $A[i] := input_{8i} || input_{8i+1} || \cdots || input_{8i+7} \quad (0 \le i \le (n-1)).$ 

<표 4-1> 입력 비트 수열과 바이트 배열의 관계

입력 비트열	0	1	 7	8	9	 15	 8n-8	8n-7		8n-1
바이트 배열		0			1			n –	1	

#### 4.2.3. 바이트 배열과 워드 배열 간의 변환

본 규격서에서 워드 x는 32비트 길이의 비트열  $x_{31} \| x_{30} \| \cdots \| x_0$ 을 의미하며, 이것은 함수 BitToInt에 의해 0과  $(2^{32}-1)$  사이의 어떤 정수에 대응된다. 반대로, 0과  $(2^{32}-1)$  사이의 정수 n은 함수 IntToBit에 의해 어떤 워드에 대응된다. 본 규격서에서 바이트 또는 워드 값은 16진법을 이용하여 표기되는데, 이 표기는 바이트 또는 워드의 비트들을 4비트씩 묶고 그 4비트를 <표 4-2>와 같이 단일 문자에 대응시켜 표기한다.

<표 4-2> 비트 패턴의 16진수 표현

비트 패턴	문자						
0000	0	0100	4	1000	8	1100	С
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	а	1110	е
0011	3	0111	7	1011	b	1111	f

- 4.1절에 소개된 연산 B←A를 구체적으로 다음과 같이 정의한다.
- ① A와 B가 같은 형의 비트, 바이트, 또는 워드인 경우: B는 A와 동일한 비트열을 가짐
- ② A가 바이트 배열이고 B가 워드인 경우: B에 A = (A[0], A[1], A[2], A[3])의 비트열들이

$$B = A[3] || A[2] || A[1] || A[0]$$

과 같이 대입됨

③ A와 B가 같은 형의 배열인 경우: B=(B[0], B[1], ···, B[n-1])에 A=(A[0], A[1], ···, A[n-1])의 비트열들이

$$B[i] = A[i] \quad (0 \le i \le (n-1))$$

을 만족하도록 대입됨

④ A가 바이트 배열이고 B가 워드 배열인 경우: B = (B[0], B[1], ···, B[n-1])에 A = (A [0], A[1], ···, A[4n-1])의 비트열들이

$$B[i] = A[4i + 3] || A[4i + 2] || A[4i + 1] || A[4i] (0 \le i \le (n - 1))$$

을 만족하도록 대입됨

⑤ A가 워드 배열이고 B가 바이트 배열인 경우: B=(B[0], B[1], ···, B[4n-1])에 A=(A[0], A[1], ···, A[n-1])의 비트열들이

$$A[i] = B[4i + 3] || B[4i + 2] || B[4i + 1] || B[4i] (0 \le i \le (n - 1))$$

을 만족하도록 대입됨

바이트 배열, 워드 배열 및 워드 내 비트 색인의 관계를 정리하면 <표 4-3>과 같다.

<표 4-3> 바이트 배열, 워드 배열, 워드 내 비트 색인의 관계

바이트 배열	3	2	1	0	7	6	5	4	•••	4n-1	4n-2	4n-3	4n-4
워드 배열		(	)				1				n -	- 1	
워드 내 비트 색인	31	•	••	0	31	•	••	0		31			0

#### 5. 블록암호 LEA

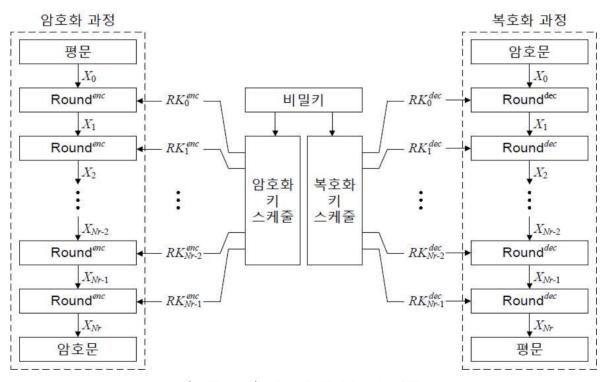
#### 5.1. 규격

본 규격서에서는 LEA를 각 키 길이에 따라 LEA-128, LEA-192, LEA-256으로 구분한다. 블록암호 LEA의 규격은 <표 5-1>과 같이 정리할 수 있다. 블록 길이를 Nb 바이트, 비밀키 길이를 Nk 바이트, 라운드 수를 Nr로 나타내었다.

구분	Nb	Nk	Nr
LEA-128	16	16	24
LEA-192	16	24	28
LEA-256	16	32	32

<표 5-1> LEA 규격

LEA의 전체적인 동작 과정을 도시하면 (그림 5-1)과 같다.



(그림 5-1) 암호화 및 복호화 과정

#### 5.2. 암호화

LEA의 암호화 과정은 k비트 키 K로부터 Nr개의 192비트 암호화용 라운드키 RK $_{i}^{enc}$  (0  $\leq$   $i \leq$  (Nr - 1))를 생성하는 키 스케줄 함수 KeySchedule $_{k}^{enc}$ 와, 라운드키 RK $_{i}^{enc}$  및 라운드

함수 Round<sup>enc</sup>를 이용하여 128비트 평문 P를 128비트 암호문 C로 변환하는 암호화 함수 Encrypt로 구성된다.

#### 5.2.1. 암호화 함수

LEA의 암호화 함수 Encrypt는 k비트 키 K에 대해 키 스케줄 함수 KeySchedule $_k^{enc}$ 을 수행하여 생성된 Nr개의 192비트 라운드키

$$RK_{i}^{enc} = (RK_{i}^{enc}[0], RK_{i}^{enc}[1], \dots, RK_{i}^{enc}[5]) \quad (0 \le i \le (Nr - 1))$$

와 128비트 평문 P=(P[0], P[1], ···, P[15])를 입력받아 알고리즘 1을 수행하여 128비트 암호문 C=(C[0], C[1], ···, C[15])를 출력한다.

알고리즘 1 암호화 함수:  $C \leftarrow Encrypt(P, RK_0^{enc}, RK_1^{enc}, \cdots, RK_{Nr-1}^{enc})$ 

입력: 128비트 평문 P, Nr개의 192비트 라운드키 RK<sup>enc</sup>, RK<sup>enc</sup>, ···, RK<sup>enc</sup>

출력: 128비트 암호문 C

1:  $X_0 \leftarrow P$ 

2: **for** i = 0 to (Nr - 1) **do** 

3:  $X_{i+1} \leftarrow Round^{enc}(X_i, RK_i^{enc})$ 

4: end for

5:  $C \leftarrow X_{Nr}$ 

알고리즘 1에서 i (0 ≤ i ≤ (Nr - 1))번째 라운드의 라운드 함수 Round<sup>enc</sup>는 128비트 내 부상태 변수

$$X_i = (X_i[0], X_i[1], X_i[2], X_i[3])$$

와 192비트 라운드키

$$RK_i^{enc} = (RK_i^{enc}[0], RK_i^{enc}[1], \dots, RK_i^{enc}[5])$$

로부터 알고리즘 2를 수행하여 새로운 128비트 내부상태 변수

$$X_{i+1} = (X_{i+1}[0], X_{i+1}[1], X_{i+1}[2], X_{i+1}[3])$$

을 생성한다.

알고리즘 2 암호화 과정의 i번째 라운드 함수: X<sub>i+1</sub> ← Round<sup>enc</sup>(X<sub>i</sub>, RK<sup>enc</sup>)

입력: 128비트 내부상태 변수 Xi, 192비트 라운드키 RKienc

출력: 128비트 내부상태 변수 Xi+1

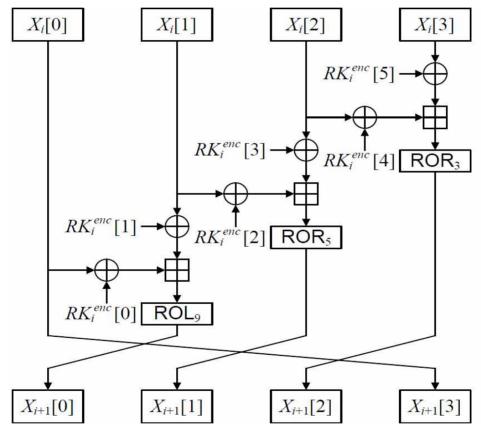
1:  $X_{i+1}[0] \leftarrow ROL_9((X_i[0] \oplus RK_i^{enc}[0]) \boxplus (X_i[1] \oplus RK_i^{enc}[1]))$ 

 $2 \colon \ X_{i+1}[1] \leftarrow ROR_5((X_i[1] \oplus RK_i^{enc}[2]) \boxplus (X_i[2] \oplus RK_i^{enc}[3]))$ 

3:  $X_{i+1}[2] \leftarrow ROR_3((X_i[2] \oplus RK_i^{enc}[4]) \boxplus (X_i[3] \oplus RK_i^{enc}[5]))$ 

4:  $X_{i+1}[3] \leftarrow X_{i}[0]$ 

(그림 5-2)는 암호화 과정의 i번째 라운드 함수를 도식화한 것이다.



(그림 5-2) 암호화 과정의 i번째 라운드 함수 (0≤i≤(Nr-1))

#### 5.2.2. 암호화 키 스케줄

키 K로부터 암호화 과정에 필요한 Nr개의 192비트 암호화 라운드키 RK $_i^{enc}$   $(0 \le i \le (N r-1))$ 들을 생성하는 키 스케줄 과정을 설명한다.

키 스케줄 함수에서 사용되는 32비트 상수들  $\delta[i]$  ( $0 \le i \le 7$ )는 다음과 같다.

 $\delta[0] = c3efe9db,$   $\delta[1] = 44626b02,$   $\delta[2] = 79e27c8a,$   $\delta[3] = 78df30ec,$   $\delta[4] = 715ea49e,$   $\delta[5] = c785da0a,$   $\delta[6] = e04ef22a,$   $\delta[7] = e5c40957.$ 

128비트 키 K=(K[0], K[1], ···, K[15])에 대해 LEA-128의 암호화를 위해 사용되는 키 스케줄 함수 KeySchedule<sup>enc</sup>는 24개의 192비트 암호화 라운드키

$$RK_{i}^{enc} = (RK_{i}^{enc}[0], RK_{i}^{enc}[1], \dots, RK_{i}^{enc}[5]) (0 \le i \le 23)$$

를 알고리즘 3과 같이 생성하며, 이 과정에서 128비트 내부상태 변수 T=(T[0],T[1],T[2],T[3])가 사용된다.

알고리즘 3 LEA-128 암호화 키 스케줄 함수: (RK<sup>enc</sup>, ···, RK<sup>enc</sup>) ← KeySchedule<sup>enc</sup>(K)

입력: 128비트 비밀키 K

출력: 24개의 192비트 암호화 라운드키 RK<sub>i</sub><sup>enc</sup> (0 ≤ i ≤ 23)

1: T ← K

2: for i = 0 to 23 do

3:  $T[0] \leftarrow ROL_1(T[0] \boxplus ROL_1(\delta[i \mod 4]))$ 

4:  $T[1] \leftarrow ROL_3(T[1] \boxplus ROL_{i+1}(\delta[i \mod 4]))$ 

5:  $T[2] \leftarrow ROL_6(T[2] \boxplus ROL_{i+2}(\delta[i \mod 4]))$ 

6:  $T[3] \leftarrow ROL_{11}(T[3] \boxplus ROL_{i+3}(\delta[i \mod 4]))$ 

7:  $RK_i^{enc} \leftarrow (T[0], T[1], T[2], T[1], T[3], T[1])$ 

8: end for

192비트 키 K=(K[0], K[1], ..., K[23])에 대해 LEA-192의 암호화를 위해 사용되는 키 스케줄 함수 KeySchedule<sup>enc</sup>는 28개의 192비트 암호화 라운드키

$$RK_{i}^{enc} = (RK_{i}^{enc}[0], RK_{i}^{enc}[1], \dots, RK_{i}^{enc}[5]) (0 \le i \le 27)$$

를 알고리즘 4와 같이 생성하며, 이 과정에서 192비트 내부상태 변수  $T = (T[0], T[1], \cdots, T[5])$ 가 사용된다.

```
알고리즘 4 LEA-192 암호화 키 스케줄 함수: (RK_0^{enc}, \cdots, RK_{27}^{enc}) \leftarrow KeySchedule_{192}^{enc}(K)
입력: 192비트 비밀키 K
출력: 28개의 192비트 암호화 라운드키 RK; (0 ≤ i ≤ 27)
 1: T ← K
 2: for i = 0 to 27 do
 3:
            T[0] \leftarrow ROL_1(T[0] \boxplus ROL_i(\delta[i \mod 6]))
            T[1] \leftarrow ROL_3(T[1] \boxplus ROL_{i+1}(\delta[i \mod 6]))
 4:
 5:
            T[2] \leftarrow ROL_6(T[2] \boxplus ROL_{i+2}(\delta[i \mod 6]))
 6:
            T[3] \leftarrow ROL_{11}(T[3] \boxplus ROL_{i+3}(\delta[i \mod 6]))
 7:
            T[4] \leftarrow ROL_{13}(T[4] \boxplus ROL_{i+4}(\delta[i \mod 6]))
            T[5] \leftarrow ROL_{17}(T[5] \boxplus ROL_{i+5}(\delta[i \mod 6]))
 8:
            RK_i^{enc} \leftarrow (T[0], T[1], T[2], T[3], T[4], T[5])
 9:
```

256비트 키 K=(K[0], K[1], ..., K[31])에 대해 LEA-256의 암호화를 위해 사용되는 키 스케줄 함수 KeySchedule<sup>enc</sup>는 32개의 192비트 암호화 라운드키

10: end for

$$RK_i^{enc} = (RK_i^{enc}[0], RK_i^{enc}[1], \dots, RK_i^{enc}[5]) (0 \le i \le 31)$$

를 알고리즘 5와 같이 생성하며, 이 과정에서 256비트 내부상태 변수 T = (T[0], T[1], …, T[7])가 사용된다.

```
알고리즘 5 LEA-256 암호화 키 스케줄 함수: (RK<sub>0</sub><sup>enc</sup>, ···, RK<sub>31</sub><sup>enc</sup>) ← KeySchedule<sup>enc</sup>(K)
입력: 256비트 비밀키 K
출력: 32개의 192비트 암호화 라운드키 RK; (0 ≤ i ≤ 31)
 1: T ← K
 2: for i = 0 to 31 do
 3:
           T[6i \mod 8] \leftarrow ROL_1(T[6i \mod 8] \boxplus ROL_1(\delta[i \mod 8]))
 4:
           T[(6i+1) \mod 8] \leftarrow ROL_3(T[(6i+1) \mod 8] \boxplus ROL_{i+1}(\delta[i \mod 8]))
 5:
           T[(6i+2) \mod 8] \leftarrow ROL_6(T[(6i+2) \mod 8] \boxplus ROL_{i+2}(\delta[i \mod 8]))
           T[(6i+3) \mod 8] \leftarrow ROL_{11}(T[(6i+3) \mod 8] \boxplus ROL_{i+3}(\delta[i \mod 8]))
 6:
 7:
           T[(6i+4) \mod 8] \leftarrow ROL_{13}(T[(6i+4) \mod 8] \boxplus ROL_{i+4}(\delta[i \mod 8]))
 8:
           T[(6i+5) \mod 8] \leftarrow ROL_{17}(T[(6i+5) \mod 8] \boxplus ROL_{i+5}(\delta[i \mod 8]))
 9:
           RK_i^{enc} \leftarrow (T[6i \mod 8], T[(6i + 1) \mod 8], T[(6i + 2) \mod 8], T[(6i + 3) \mod 8],
                      T[(6i+4) \mod 8], T[(6i+5) \mod 8])
10: end for
```

#### 5.3. 복호화

LEA의 복호화 과정은 k비트 키 K로부터 Nr개의 192비트 복호화용 라운드키  $RK_i^{dec}$  (0  $\leq i \leq (Nr-1)$ )를 생성하는 키 스케줄 함수 KeySchedule $_k^{dec}$ 와, 라운드키  $RK_i^{dec}$  및 라운드함수 Round $_k^{dec}$ 를 이용하여 128비트 암호문 C를 128비트 평문 P로 변환하는 복호화 함수 Decrypt로 구성된다.

#### 5.3.1. 복호화 함수

LEA의 복호화 함수 Decrypt는 k비트 키 K에 대해 키 스케줄 함수 KeySchedule<sup>dec</sup>을 수행하여 생성된 Nr개의 192비트 라운드키

$$RK_i^{dec} = (RK_i^{dec}[0], RK_i^{dec}[1], \dots, RK_i^{dec}[5])$$
  $(0 \le i \le (Nr - 1))$ 

와 128비트 암호문 C = (C[0], C[1], ···, C[15])를 입력받아 알고리즘 6을 수행하여 128 비트 평문 P = (P[0], P[1], ···, P[15])를 출력한다.

알고리즘 6 복호화 함수:  $P \leftarrow Decrypt(C, RK_0^{dec}, RK_1^{dec}, \cdots, RK_{Nr-1}^{dec})$ 

입력: 128비트 암호문 C, Nr개의 192비트 라운드키 RK<sup>dec</sup>, RK<sup>dec</sup>, ···, RK<sup>dec</sup>

출력: 128비트 평문 P

1:  $X_0 \leftarrow C$ 

2: **for** i = 0 to (Nr - 1) **do** 

3:  $X_{i+1} \leftarrow Round^{dec}(X_i, RK_i^{dec})$ 

4: end for

5:  $P \leftarrow X_{Nr}$ 

알고리즘 6에서 i (0 ≤ i ≤ (Nr - 1))번째 라운드의 라운드 함수 Round<sup>dec</sup>는 128비트 내 부상태 변수

$$X_i = (X_i[0], X_i[1], X_i[2], X_i[3])$$

와 192비트 라운드키

$$RK_i^{dec} = (RK_i^{dec}[0], RK_i^{dec}[1], \dots, RK_i^{dec}[5])$$

로부터 알고리즘 7을 수행하여 새로운 128비트 내부상태 변수

$$X_{i+1} = (X_{i+1}[0], X_{i+1}[1], X_{i+1}[2], X_{i+1}[3])$$

을 생성한다.

```
알고리즘 7 복호화 과정의 i번째 라운드 함수: X_{i+1} \leftarrow Round^{dec}(X_i, RK_i^{dec})
```

입력: 128비트 내부상태 변수 X<sub>i</sub>, 192비트 라운드키 RK<sup>dec</sup>

출력: 128비트 내부상태 변수 Xi+1

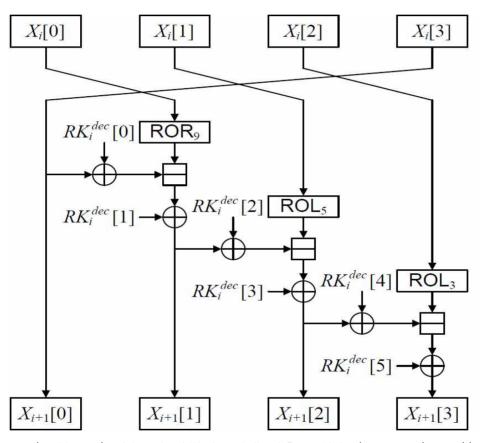
1:  $X_{i+1}[0] \leftarrow X_i[3]$ 

 $2 \colon \ X_{i+1}[1] \leftarrow (ROR_9(X_i[0]) \boxminus (X_{i+1}[0] \oplus RK_i^{dec}[0])) \oplus RK_i^{dec}[1]$ 

 $3 \colon \ X_{i+1}[2] \leftarrow (ROL_5(X_i[1]) \boxminus (X_{i+1}[1] \oplus RK_i^{dec}[2])) \oplus RK_i^{dec}[3]$ 

4:  $X_{i+1}[3] \leftarrow (ROL_3(X_i[2]) \boxminus (X_{i+1}[2] \oplus RK_i^{dec}[4])) \oplus RK_i^{dec}[5]$ 

(그림 5-3)은 복호화 과정의 i번째 라운드 함수를 도식화한 것이다.



(그림 5-3) 복호화 과정의 i번째 라운드 함수 (0≤i≤(Nr-1))

#### 5.3.2. 복호화 키 스케줄

키 K로부터 복호화 과정에 필요한 Nr개의 복호화 라운드키  $RK_i^{dec}$   $(0 \le i \le (Nr-1))들을 생성하는 키 스케줄 과정을 설명한다. 암호화 라운드키와 복호화 라운드키는$ 

$$RK_i^{dec} = RK_{Nr-i-1}^{enc} \quad (0 \le i \le (Nr-1))$$

인 관계를 제외하면 동일한 방법으로 생성되며, 동일한 상수가 사용된다.

128비트 키 K=(K[0], K[1], ..., K[15])에 대해 LEA-128의 복호화를 위해 사용되는 키 스케줄 함수 KeySchedule<sup>dec</sup>는 24개의 192비트 복호화 라운드키

$$RK_i^{dec} = (RK_i^{dec}[0], RK_i^{dec}[1], \dots, RK_i^{dec}[5]) \ (0 \le i \le 23)$$

를 알고리즘 8과 같이 생성하며, 이 과정에서 128비트 내부상태 변수 T = (T[0], T[1], T[2], T[3])가 사용된다.

알고리즘 8 LEA-128 복호화 키 스케줄 함수: (RK<sup>dec</sup><sub>0</sub>, ···, RK<sup>dec</sup><sub>23</sub>) ← KeySchedule<sup>dec</sup><sub>128</sub>(K) 입력: 128비트 비밀키 K 출력: 24개의 192비트 복호화 라운드키 RK<sup>dec</sup><sub>i</sub> (0 ≤ i ≤ 23) 1: T ← K 2: for i = 0 to 23 do 3: T[0] ← ROL<sub>1</sub>(T[0] ⊞ ROL<sub>i</sub>(δ[i mod 4])) 4: T[1] ← ROL<sub>3</sub>(T[1] ⊞ ROL<sub>i+1</sub>(δ[i mod 4])) 5: T[2] ← ROL<sub>6</sub>(T[2] ⊞ ROL<sub>i+2</sub>(δ[i mod 4])) 6: T[3] ← ROL<sub>11</sub>(T[3] ⊞ ROL<sub>i+3</sub>(δ[i mod 4])) 7: RK<sup>dec</sup><sub>23-i</sub> ← (T[0], T[1], T[2], T[1], T[3], T[1])

192비트 키 K=(K[0], K[1], ..., K[23])에 대해 LEA-192의 복호화를 위해 사용되는 키 스케줄 함수 KeySchedule<sup>dec</sup>는 28개의 192비트 복호화 라운드키

$$RK_i^{dec} = (RK_i^{dec}[0], RK_i^{dec}[1], \dots, RK_i^{dec}[5]) \ (0 \le i \le 27)$$

를 알고리즘 9와 같이 생성하며, 이 과정에서 192비트 내부상태 변수 T = (T[0], T[1], ..., T[5])가 사용된다.

```
알고리즘 9 LEA-192 복호화 키 스케줄 함수: (RK<sup>dec</sup>, ···, RK<sup>dec</sup>) ← KeySchedule<sup>dec</sup>(K)
```

입력: 192비트 비밀키 K

8: end for

출력: 28개의 192비트 복호화 라운드키  $\mathsf{RK}_{\mathsf{i}}^{\mathsf{dec}}$  (0  $\leq$   $\mathsf{i}$   $\leq$  27)

```
1: T ← K
2: for i = 0 to 27 do
3:
              T[0] \leftarrow ROL_1(T[0] \boxplus ROL_i(\delta[i \mod 6]))
4:
              T[1] \leftarrow ROL_3(T[1] \boxplus ROL_{i+1}(\delta[i \mod 6]))
5:
              T[2] \leftarrow ROL_6(T[2] \boxplus ROL_{i+2}(\delta[i \mod 6]))
6:
              T[3] \leftarrow ROL_{11}(T[3] \boxplus ROL_{i+3}(\delta[i \mod 6]))
7:
              T[4] \leftarrow ROL_{13}(T[4] \boxplus ROL_{i+4}(\delta[i \mod 6]))
              T[5] \leftarrow ROL_{17}(T[5] \boxplus ROL_{i+5}(\delta[i \mod 6]))
8:
             \mathsf{RK}^{\mathsf{dec}}_{27-\mathsf{i}} \leftarrow (\mathsf{T[0]}, \mathsf{T[1]}, \mathsf{T[2]}, \mathsf{T[3]}, \mathsf{T[4]}, \mathsf{T[5]})
9:
```

256비트 키 K=(K[0], K[1], ···, K[31])에 대해 LEA-256의 복호화를 위해 사용되는 키 스케줄 함수 KeySchedule<sup>dec</sup>는 32개의 192비트 복호화 라운드키

$$\mathsf{RK}_{i}^{\mathsf{dec}} = (\mathsf{RK}_{i}^{\mathsf{dec}}[0], \, \mathsf{RK}_{i}^{\mathsf{dec}}[1], \, \cdots, \, \mathsf{RK}_{i}^{\mathsf{dec}}[5]) \, (0 \le i \le 31)$$

를 알고리즘 10과 같이 생성하며, 이 과정에서 256비트 내부상태 변수 T = (T[0], T[1], ···, T[7])가 사용된다.

```
알고리즘 10 LEA-256 복호화 키 스케줄 함수: (RK<sup>dec</sup>, ···, RK<sup>dec</sup>) ← KeySchedule<sup>dec</sup>(K)
입력: 256비트 비밀키 K
출력: 32개의 192비트 복호화 라운드키 RK<sup>dec</sup> (0 ≤ i ≤ 31)
1: T ← K
2: for i = 0 to 31 do
3: T[6i mod 8] ← ROL<sub>1</sub>(T[6i mod 8] ⊞ ROL<sub>i</sub>(δ[i mod 8]))
4: T[(6i+1) mod 8] ← ROL<sub>3</sub>(T[(6i+1) mod 8] ⊞ ROL<sub>i+1</sub>(δ[i mod 8]))
5: T[(6i+2) mod 8] ← ROL<sub>6</sub>(T[(6i+2) mod 8] ⊞ ROL<sub>i+2</sub>(δ[i mod 8]))
6: T[(6i+3) mod 8] ← ROL<sub>11</sub>(T[(6i+3) mod 8] ⊞ ROL<sub>i+3</sub>(δ[i mod 8]))
7: T[(6i+4) mod 8] ← ROL<sub>13</sub>(T[(6i+4) mod 8] ⊞ ROL<sub>i+4</sub>(δ[i mod 8]))
```

9:  $RK_{31-i}^{dec} \leftarrow (T[6i \mod 8], T[(6i + 1) \mod 8], T[(6i + 2) \mod 8], T[(6i + 3) \mod 8],$ 

 $T[(6i+5) \mod 8] \leftarrow ROL_{17}(T[(6i+5) \mod 8] \boxplus ROL_{i+5}(\delta[i \mod 8]))$ 

T[(6i + 4) mod 8], T[(6i + 5) mod 8])

10: end for

8:

# 부 록 I 참조구현값

#### 1.1. 규약

모든 참조구현값은 다음의 규약을 따른다.

- ① 모든 값은 16진수로 표기
- ② 바이트 배열 A가 다음과 같이 주어졌을 때, 좌측부터 배열 색인 0, 1,…에 해당 Of 1e 2d 3c 4b 5a 69 78 87 96 a5 b4 c3 d2 e1 f0

즉, A[0] = 0f, A[1] = 1e,  $\cdots$  , A[15] = f0임

③ 워드 배열 A가 다음과 같이 주어졌을 때, 좌측부터 배열 색인 0, 1,…에 해당 003a0fd4 02497010 194f7db1 02497010 090d0883 02497010

즉, A[0] = 003a0fd4, A[1] = 02497010, ··· , A[5] = 02497010임

- ④ 비밀키 K, 평문 P, 암호문 C는 바이트 배열로 표기
- ⑤ 라운드키 RK; enc (0≤i≤(Nr-1)), 내부상태 변수 X; (0≤i≤Nr)는 워드 배열로 표기
- ⑥ 바이트 배열과 워드 배열은 변환 규약(4.2.3절)을 따름

복호화 과정의 내부상태 변수 및 복호화 라운드키는 암호화 과정의 내부상태 변수 및 암호화 라운드키의 역순이라는 관계를 이용하여 확인 가능하다.

# I.2. LEA-128 암호화

# I.2.1. 암호화 함수

K	Of 1e 2d 3c 4b 5a 69 78 87 96 a5 b4 c3 d2 e1 f0
Р	10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
X <sub>0</sub>	13121110 17161514 1b1a1918 1f1e1d1c
X <sub>1</sub>	0f079051 693d668d e5edcfd4 13121110
X <sub>2</sub>	3fc44a2d f767ea2a a0b67cf0 0f079051
X <sub>3</sub>	99e912cd 906fd05d 4d293e55 3fc44a2d
$\chi_4$	43048c71 5faa8d15 dfc687fb 99e912cd
X <sub>5</sub>	862a337d 419f623d 4b97dd8a 43048c71
X <sub>6</sub>	055b3a34 a2eb0f67 af9873ba 862a337d
X <sub>7</sub>	38875cb8 19f1c052 02e13d1c 055b3a34
X <sub>8</sub>	f1ddbcca 2c031302 8a5f86d6 38875cb8
<b>X</b> 9	f770a17e c47d9365 2df8cda7 f1ddbcca
X <sub>10</sub>	58a898f4 db57aa1e 20d820a4 f770a17e
X <sub>11</sub>	11c9f487 bf079d6e 28c10b82 58a898f4
X <sub>12</sub>	a7e4a0e4 e8e97f62 47727e5f 11c9f487
X <sub>13</sub>	d1ea924a 2298587f f2afc1d0 a7e4a0e4
X <sub>14</sub>	7e91cf8c fcca259f 86ab69cf d1ea924a
X <sub>15</sub>	809fd3e9 ef492067 536df05e 7e91cf8c
X <sub>16</sub>	2b54eee2 98b175f9 d9c14ac4 809fd3e9
X <sub>17</sub>	63eb48a2 7ad2716d 783a355e 2b54eee2
X <sub>18</sub>	4b34e264 101d5f00 7fee2017 63eb48a2
X <sub>19</sub>	ba42cf9e d156295c b88c1f9d 4b34e264
X <sub>20</sub>	970433ea a0d420cb 4b96b2c1 ba42cf9e
X <sub>21</sub>	49facf18 6f1fe3c2 3744e7b8 970433ea
X <sub>22</sub>	d1527e90 6ce66afe 1d55c7f1 49facf18
X <sub>23</sub>	fd8b6404 8675df3b e4b9d73f d1527e90
X <sub>24</sub>	354ec89f 18c6c628 a7c73255 fd8b6404
С	9f c8 4e 35 28 c6 c6 18 55 32 c7 a7 04 64 8b fd

# 1.2.2. 암호화 키 스케줄

K	Of 1e 2d	3c 4b 5a	69 78 87	96 a5 b4	c3 d2 e1	fO
RK <sub>0</sub> enc	003a0fd4	02497010	194f7db1	02497010	090d0883	02497010
RK <sub>1</sub> <sup>enc</sup>	11fdcbb1	9e98e0c8	18b570cf	9e98e0c8	9dc53a79	9e98e0c8
RK <sub>2</sub> <sup>enc</sup>	f30f7bb5	6d6628db	b74e5dad	6d6628db	a65e46d0	6d6628db
RK <sub>3</sub> <sup>enc</sup>	74120631	dac9bd17	cd1ecf34	dac9bd17	540f76f1	dac9bd17
RK <sub>4</sub> <sup>enc</sup>	662147db	c637c47a	46518932	c637c47a	23269260	c637c47a
RK <sub>5</sub> <sup>enc</sup>	e4dd5047	f694285e	e1c2951d	f694285e	8ca5242c	f694285e
RK <sub>6</sub> <sup>enc</sup>	baf8e5ca	3e936cd7	Ofc7e5b1	3e936cd7	f1c8fa8c	3e936cd7
RK <sub>7</sub> <sup>enc</sup>	5522b80c	ee22ca78	8a6fa8b3	ee22ca78	65637b74	ee22ca78
RK <sub>8</sub> <sup>enc</sup>	8a19279e	6fb40ffe	85c5f092	6fb40ffe	92cc9f25	6fb40ffe
RK <sub>9</sub> <sup>enc</sup>	9dde584c	cb00c87f	4780ad66	cb00c87f	e61b5dcb	cb00c87f
RK <sup>enc</sup>	4fa10466	f728e276	d255411b	f728e276	656839ad	f728e276
RK <sub>11</sub> <sup>enc</sup>	9250d058	51bd501f	1cb40dae	51bd501f	1abf218d	51bd501f
RK <sup>enc</sup> <sub>12</sub>	21dd192d	77c644e2	cabfaa45	77c644e2	681c207d	77c644e2
RK <sup>enc</sup>	de7ac372	9436afd0	10331d80	9436afd0	f326fe98	9436afd0
RK <sub>14</sub> <sup>enc</sup>	fb3ac3d4	93df660e	2f65d8a3	93df660e	df92e761	93df660e
RK <sup>enc</sup> <sub>15</sub>	27620087	265ef76e	4fb29864	265ef76e	2656ed1a	265ef76e
RK <sup>enc</sup> <sub>16</sub>	227b88ec	d0b3fa6f	c86a08fd	d0b3fa6f	a864cba9	d0b3fa6f
RK <sup>enc</sup>	f 1002361	e5e85fc3	1f0b0408	e5e85fc3	488e7ac4	e5e85fc3
RK <sup>enc</sup> <sub>18</sub>	c65415d5	51e176b6	eca88bf9	51e176b6	edb89ece	51e176b6
RK <sub>19</sub> <sup>enc</sup>	9b6fb99c	0548254b	8de9f7c2	0548254b	b6b4d146	0548254b
RK <sub>20</sub> <sup>enc</sup>	7257f 134	06051a42	36bcef01	06051a42	b649d524	06051a42
RK <sub>21</sub> <sup>enc</sup>	a540fb03	34b196e6	f7c80dad	34b196e6	71bc7dc4	34b196e6
RK <sub>22</sub> <sup>enc</sup>	8fbee745	cf744123	907c0a60	cf744123	8215ec35	cf744123
RK <sub>23</sub> <sup>enc</sup>	Obf6adba	df69029d	5b72305a	df69029d	cb47c19f	df69029d

# I.3. LEA-192 암호화

# Ⅰ.3.1. 암호화 함수

	Of 1° 09	3c 4b 5a	60 70 07	7 OG aF I	h 1 a 2	40	_ 1	ŧΟ	tΟ	_ 1	<b>ا</b> ل	-0	h 1	۰E	06	07
									10	ет	uZ	CO	υ4	ao	90	0/
Р	20 21 22	23 24 25	26 27 28	3 29 2a 3	2b 2c	2d	2e	2f								
X <sub>0</sub>	23222120	27262524	2b2a2928	3 2f2e2d	2c											
X <sub>1</sub>	0f085091	030483d2	be4ab9e1	232221	20											
X <sub>2</sub>	23fc7579	99fa0bb5	92d65065	0f0850	91											
X <sub>3</sub>	1e64758b	6b19e366	3edbb998	3 23fc75	79											
X <sub>4</sub>	8da45638	d886dfdd	2da2b83d	1e6475	8b											
X <sub>5</sub>	a29dfd15	d8687adf	b89c47b4	8da456	38											
X <sub>6</sub>	34e43c2e	b6268ba7	584086d7	a29dfd	15											
X <sub>7</sub>	df6e285b	88f26855	198fbb0d	34e43c	2e											
X <sub>8</sub>	e62dde25	dd1cdf46	b8049544	l df6e28	5b											
X <sub>9</sub>	d715e465	0e4d136e	35bc93a5	e62dde	25											
X <sub>10</sub>	1ab97de4	a5c19c64	cfd627b0	) d715e4	65											
X <sub>11</sub>	1a155930	4ccf6ee2	8c5136f7	' 1ab97d	e4											
X <sub>12</sub>	0eef4d0d	9f3065e1	405fc8b9	) 1a1559	30											
X <sub>13</sub>	a0dd5c61	fcefa1a4	48e2adc3	0eef4d	0d											
X <sub>14</sub>	dcf21fcc	f9ca084f	0b02cdd8	a0dd5c	61											
X <sub>15</sub>	dfd53021	2eee92c5	eedfe48b	dcf21f	СС											
X <sub>16</sub>	81dce833	4e0af1c2	3abac76k	dfd530	21											
X <sub>17</sub>	9646ef75	607a7771	9fbc48ec	81dce8	33											
X <sub>18</sub>	7f40d072	ac609c27	5de1c810	) 9646ef	75											
X <sub>19</sub>	fd0bae5f	35429a83	11f5e49f	7f40d0	72											
X <sub>20</sub>	2feb9af2	0799218a	e5374a51	fd0bae	5f											
X <sub>21</sub>	fd86cfee	a7d97a82	7c97ed23	3 2feb9a	f2											
X <sub>22</sub>	add0adf1	e09057e1	ccd95f65	fd86cf	ee											
X <sub>23</sub>	06c45cfe	392aaa1d	ae9fd56d	add0ad	f1											
X <sub>24</sub>	f3032315	b1df9d75	16279280	06c45c	fe											
X <sub>25</sub>	f4eec840	6a15d699	2392c666	f30323	15											
X <sub>26</sub>	b353eb6a	ad286c22	5a9d146d	f4eec8	40											
X <sub>27</sub>	f2c67476	44333e44	6d5a1045	b353eb	6a											
X <sub>28</sub>	325eb96f	871bad5a	35f5dc8d	f2c674	76											
C	6f b9 5e	32 5a ad	1b 87 8d	dc f5	35 76	74	с6	f2								

# 1.3.2. 암호화 키 스케줄

K	0f 1e 2d	3c 4b 5a	69 78 87	96 a5 b4	c3 d2 e1	f0 f0 e1	d2 c3	b4	a5	96	87
RK <sub>0</sub> <sup>enc</sup>	003a0fd4	02497010	194f7db1	090d0883	2ff5805a	c2580b27					
RK <sub>1</sub> <sup>enc</sup>	11fdcbb1	9e98e0c8	18b570cf	9dc53a79	5c145788	9771b5e5					
RK <sub>2</sub> <sup>enc</sup>	f30f7bb5	6d6628db	b74e5dad	a65e46d0	6f44da96	f643115f					
RK <sub>3</sub> <sup>enc</sup>	74120631	dac9bd17	cd1ecf34	540f76f1	aa1a5bdb	fbafaae7					
RK <sub>4</sub> <sup>enc</sup>	13f8a031	34f28728	31fdb409	0e31481b	df498117	cf9371f1					
RK <sub>5</sub> <sup>enc</sup>	0967c312	b3484ec8	3aae5b3d	5a9714a0	b2d4dd5f	3a1fcdf7					
RK <sub>6</sub> <sup>enc</sup>	0ac47404	59e9e54d	a60dc00a	566139d3	898dce4f	582d72dd					
RK <sub>7</sub> <sup>enc</sup>	77f3ea4c	e2a73c8d	b8f1249a	6a172700	bc0e539c	2e46fdbb					
RK <sub>8</sub> <sup>enc</sup>	b4e0e98a	3d028c05	b8d3a050	dbd67bef	df675c7a	99eefbb0					
RK <sub>9</sub> <sup>enc</sup>	e68584f6	ce31ef45	96c105ac	2a1be677	9d72b8b0	33cecc54					
RK <sup>enc</sup>	c22ffd76	1ab7167e	42bb3060	7da517f5	4aa0e8d3	0a070c3c					
RK <sup>enc</sup>	e200a765	c2be17b3	7f22543f	3e4eb7a1	c992a6f4	a783c823					
RK <sup>enc</sup> <sub>12</sub>	c13cc747	ffcc8185	66514e9e	e4ccc199	cd5c766d	a004f676					
RK <sup>enc</sup>	1d3a1fa6	d46894ec	f49c33e6	782fda7e	1fe6346c	Offe981c					
RK <sup>enc</sup> <sub>14</sub>	78b97c3d	956e8ee8	49ab721c	2672138a	037ea242	ce5fe8a4					
RK <sup>enc</sup> <sub>15</sub>	225f7158	32d83e3e	e118f6aa	1fb83751	4d27715c	ed2fba4e					
RK <sub>16</sub> <sup>enc</sup>	8dfbc56d	e0a907db	e4af091c	5e123225	d0e8d2e1	cc4501fb					
RK <sup>enc</sup>	8422a8f0	46a12f92	415152ad	f55417f5	38738248	c6e29ded					
RK <sup>enc</sup> <sub>18</sub>	5723715e	abfa788c	c3646af7	64af9186	8fc855ec	2bc36989					
RK <sup>enc</sup> <sub>19</sub>	5e6b28e3	e0f5f592	eb3dd108	0551012a	50e4221d	97e85c0f					
RK <sub>20</sub> <sup>enc</sup>	4e258e14	92298f0b	771269c3	6f934254	c0933b6b	421159b8					
RK <sub>21</sub> <sup>enc</sup>	d76953f4	6a3e36be	53b656fb	610c22e0	9f399330	acf7e7e9					
RK <sub>22</sub> <sup>enc</sup>	fe0b573b	cbb73085	89ed67fc	77014cef	e1b8431f	ba1b4105					
RK <sub>23</sub> <sup>enc</sup>	06de3450	b3f5b2fe	df1cec27	fb22bd10	8e3de6fe	3d4acd27					
RK <sub>24</sub> <sup>enc</sup>	c5444873	5bec968b	8b2af393	11e2f6ca	9cb3694f	94c56b91					
RK <sub>25</sub> <sup>enc</sup>	939a1a93	27f 101bb	5381bae7	48ebd1b1	f6d5fca7	0ca24bbc					
RK <sub>26</sub> <sup>enc</sup>	7b03490b	de00acfb	c7f8abfe	410a14c1	d37932a9	14029327					
RK <sub>27</sub>	bd948525	2c75004d	c52486d5	Of07e2fa	1963e1fd	882719c3					

# I.4. LEA-256 암호화

# Ⅰ.4.1. 암호화 함수

K	Of 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
Р	30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
X <sub>0</sub>	33323130 37363534 3b3a3938 3f3e3d3c
X <sub>1</sub>	0f0810d1 030583d2 a246bdef 33323130
X <sub>2</sub>	e370475a 379d1cd0 7b627f7b 0f0810d1
Х3	a212c114 c5be3591 435bb556 e370475a
$\chi_4$	90bcb059 94df55a0 d8e15ef6 a212c114
X <sub>5</sub>	4e5cc849 f484b5d8 ef0fc663 90bcb059
X <sub>6</sub>	a520787d ae3db194 fa2feb17 4e5cc849
X <sub>7</sub>	231a5d45 f338ad75 9d4b9850 a520787d
X <sub>8</sub>	dd744e80 0a6568a0 3f9c93a9 231a5d45
X <sub>9</sub>	d141f34e 311ba7ed b34c9f6f dd744e80
X <sub>10</sub>	f5a76166 3e00a130 b1f9a58d d141f34e
X <sub>11</sub>	af52973c c4befd35 aae4a642 f5a76166
X <sub>12</sub>	3df5e119 b8937ebb b4a7a6ab af52973c
X <sub>13</sub>	ede0ddb3 2c84bd26 2c86c203 3df5e119
X <sub>14</sub>	960f7157 477a5b5a 29659f76 ede0ddb3
X <sub>15</sub>	2c8d1d71 e0b54ae6 fbdd003c 960f7157
X <sub>16</sub>	720a9b5f 18bf274a 0a1af597 2c8d1d71
X <sub>17</sub>	1aa88202 c3867edc c49ce291 720a9b5f
X <sub>18</sub>	e16c34f7 69defa8b 15b2990f 1aa88202
X <sub>19</sub>	c7837b0b cf85d76f 17203201 e16c34f7
X <sub>20</sub>	a3061bb3 bbe1ce55 00a3f8e9 c7837b0b
X <sub>21</sub>	54f6bcfa c195ea5b b8280ef0 a3061bb3
X <sub>22</sub>	662f351e 49995ddc 4ef48970 54f6bcfa
X <sub>23</sub>	09db178e 4748e08a 30ba1411 662f351e
$X_{24}$	751113cb 9a425ee2 200fee63 09db178e
X <sub>25</sub>	47d21a23 08561dff 86131859 751113cb
X <sub>26</sub>	f7aaf6ac 4f5eac5b f4247a5b 47d21a23
X <sub>27</sub>	8e952768 3de52e9b 367ed97c f7aaf6ac
X <sub>28</sub>	cb641a2d 8d161a90 dbd4a137 8e952768
X <sub>29</sub>	b6e87380 93b8b779 c9530e82 cb641a2d
X <sub>30</sub>	48bd3559 5ad96ae7 2e0feb8b b6e87380
X <sub>31</sub>	97e1f927 853a0309 487c41fc 48bd3559
X <sub>32</sub>	f6af51d6 c189b147 ca00893a 97e1f927
С	d6 51 af f6 47 b1 89 c1 3a 89 00 ca 27 f9 e1 97

# I.4.2. 암호화 키 스케줄

	Of 1a 2d	3c 4b 5a	60 78 87	96 a5 h1	c3 d2 <u>a</u> 1	fn
K						
DIZenc		c3 b4 a5				
RK <sub>0</sub> <sup>enc</sup>		02497010				
RK <sub>1</sub> <sup>enc</sup>		053eca29				
RK <sub>2</sub> <sup>enc</sup>		8b5f7bd4				
RK <sub>3</sub> <sup>enc</sup>		00da90b1				
RK <sub>4</sub> <sup>enc</sup>		d69bc26f				
RK <sub>5</sub> <sup>enc</sup>		f3f93651				<del></del>
RK <sub>6</sub> <sup>enc</sup>	7869db69	6b7a5b8e	fefbf6b1	ec608c8e	76e9d5d2	13ca4bf6
RK <sub>7</sub> <sup>enc</sup>	c5eeec7a	aa42a59d	1f22cd00	fdd92bdc	d6bbe3e8	15d459ec
RK <sub>8</sub> <sup>enc</sup>	cda7632a	9cf01bef	6596e261	8c1de14c	1127c3b8	48b3f629
RK <sub>9</sub> enc	3723d0e1	fc0317ec	3fdd5378	0201ae1d	e55db65e	e4c84dbc
RK <sub>10</sub> <sup>enc</sup>	3633db3f	e4c24fc2	bb1e1fd7	a339425c	fe3e1bdf	d61c808d
RK <sub>11</sub>	bdca3449	beb8aa4e	145a9687	eb6fcd87	8b88ca72	7677a84b
RK <sup>enc</sup>	d11005e9	558275c5	bc742819	3f 17e888	20fcb71f	60886959
RK <sup>enc</sup>	8d9446c4	67d2d167	855a6aef	69ea517c	36e48e11	0d3f4e86
RK <sub>14</sub> <sup>enc</sup>	bb0ede65	cceecc06	efc9c49f	44902261	bd8549c0	a7e7f682
RK <sup>enc</sup> <sub>15</sub>	772101e6	b4b9a250	6faa7b73	7318b792	1e57e751	fd43b41c
RK <sub>16</sub> <sup>enc</sup>	4ec21b5f	dcfbf30b	a4046947	be0e781c	d74e21ac	6b1f5d22
RK <sup>enc</sup>	e8b8e02b	4a662d2d	b50f9ca9	01c98c69	9eb28089	216cfd3f
RK <sub>18</sub> <sup>enc</sup>	92f0126b	7b9961aa	581f94ac	ab4be6dd	c2a91af5	fb4e8e0c
RK <sup>enc</sup> <sub>19</sub>	4c2c8f04	81a45991	1fcb946c	bccbb5b5	808899cb	8c1b2f89
RK <sub>20</sub> <sup>enc</sup>	192061be	78e5cf04	f239ab5c	e8471e86	9e6217c7	e5fdf35c
RK <sub>21</sub>	83c3150d	766887f8	a1092ac7	6aa6f41d	16e200f9	6bdc26ca
RK <sub>22</sub>	52345706	db70d6af	a8d8ffeb	492ee661	4cd1e991	d75d8352
RK <sub>23</sub> <sup>enc</sup>	85a9c5fb	1e0f569e	7ff7c600	3f36a1d8	e406ad00	4ded8f16
RK <sub>24</sub>	512bb2f4	772b192c	2e6168bd	76af67e1	d893a786	3e276f69
RK <sub>25</sub>	d11ee3ad	b7f8c612	d3b19318	89fee4db	b6c3aedd	05420f90
RK <sub>26</sub>	04f662f0	8fb41a6c	2f42dd5e	a8ad1839	46474e45	46418de0
RK <sub>27</sub>	351550c8	668014f6	04924365	5f353d6f	4eba8d76	924a4318
RK <sub>28</sub>	5aba711c	a36b1398	5b3e7bf4	7b3a2cf9	1d006ebe	0d5683e5
RK <sub>29</sub>	4f56916f	215dccd2	9f57886f	876d1357	46013d49	2a4932a3
RK <sub>30</sub>		ebefe7d3				
RK <sub>31</sub>		8069dfd0				