# 해시함수 LSH 규격서



## 목 차

1.	개요 ····································	
2.	구성 및 범위1	
3.	용어 정의 및 약어1	
4.	해시함수 LSH ·······6	;
	4.1. 전체 구조	;
	4.2. 압축함수 8	;
	4.3. 초기화 상수	3

## 해시함수 LSH

#### 1. 개요

해시함수는 임의 길이의 비트열을 입력받아 고정 길이의 비트열을 출력하는 함수이다. 해시함수가 암호학적으로 사용되기 위해 요구되는 세 가지 성질로는 역상 저항성(Preima ge Resistance), 제 2 역상 저항성(2nd Preimage Resistance), 그리도 충돌 저항성(Coll ision Resistance)이 있다. 제 2 역상 저항성과 충돌 저항성은 충돌 쌍을 찾는 문제의 어려움에 기반한다. 여기서 충돌 쌍이란 동일한 해시함수 출력값을 갖는 서로 다른 두 입력값을 의미한다. 이러한 성질들을 만족하는 해시함수는 사용자 및 메시지 정보 변조 방지를 위한 인증 기술의 핵심 요소로 활용될 수 있다. 해시함수는 파일이나 메시지의 무결성을 검증하는 용도로 단독 사용될 뿐만 아니라, 전자 서명, 메시지 인증 코드, 암호키유도, 의사난수 생성 등에 있어서도 핵심 기능을 담당하고 있다.

LSH(Lightweight Secure Hash)는 w 비트 워드 단위로 동작하여 n 비트 출력값을 가지는 해시함수 LSH-8w-n으로 구성된 해시함수 군(Hash Function Family)이다. 여기에서 w는 32 또는 64이고, n은 1과 8w 사이의 정수이다. 해시함수 LSH-8w-n은 상기 암호학적 해시함수의 성질을 모두 제공할 수 있도록 설계되었다.

본 규격서는 해시함수 군 LSH를 구성하는 여섯 개의 주요 해시함수 LSH-224, LSH-256, LSH-512-224, LSH-512-256, LSH-384, LSH-512의 규격을 제시한다. LSH-224와 LSH-256은 각각 LSH-256-224, LSH-256-256을 의미하며, LSH-384, LSH-512는 각각 LSH-512-384, LSH-512를 의미한다.

#### 2. 구성 및 범위

본 규격서는 해시함수 LSH-224, LSH-256, LSH-512-224, LSH-512-256, LSH-384, 그리고 LSH-512의 전체 구조와 동작 방식을 설명한다. 그리고 핵심 구성 요소인 압축함수를 설명하고 세부 논리를 상술한다.

- 3. 용어 정의 및 약어
- 3.1. 용어 정의
- 3.1.1. 일반 용어
- 3.1.1.1 비트
  - 0 또는 1의 값을 가지는 이진수

#### 3.1.1.2. 순서열(Ordered Sequence)

어떤 집합에서 원소들의 중복을 허용하고 하나씩 선택하여 나오는 순서대로 나열한 것

#### 3.1.1.3. 비트열

비트들의 순서열

#### 3.1.1.4. 바이트

8개의 비트로 이루어진 비트열

#### 3.1.1.5. 워드

w 비트 길이의 비트열

#### 3.1.1.6. 메시지 블록

32개 워드의 배열로 구성된 해시 압축 함수 입력 메시지 처리 단위

#### 3.1.1.7. 상수

어떤 알고리즘에서 고정되어 사용되는 값

#### 3.1.1.8. 연접

동일 유형의 순서열 두 개를 이어붙이는 연산

#### 3.1.1.9. 배열(Array)

색인과 색인에 대응하는 데이터들로 이루어진 자료 구조

#### 3.1.2. 해시함수 용어

#### 3.1.2.1. 역상 공격(Preimage Attack)

주어진 해시함수 출력값을 가지는 입력값을 찾는 공격

#### 3.1.2.2. 제 2 역상 공격(2nd Preimage Attack)

주어진 해시함수 입력값과 출력값의 쌍에 대해, 동일한 출력값을 가지는 새로운 해 시함수 입력값을 찾는 공격

#### 3.1.2.3. 충돌 쌍 공격(Collision Attack)

같은 출력값을 가지는 서로 다른 두 개의 해시함수 입력값을 찾는 공격

#### 3.1.2.4. 패딩(Padding)

입력 데이터의 길이를 해시함수 내부 처리 단위 길이의 배수가 되도록 만들기 위해 특정 형태의 비트열을 덧붙이는 것

#### 3.1.2.5. 연결 변수(Chaining Variable)

해시 압축 함수의 출력값을 저장하는 변수

#### 3.1.2.6. 초기화 상수(IV: Initialization Vector)

해시함수 초기값

#### 3.2. 약어

XOR eXclusive OR mod modular

#### 3.3. 기호

 $\mathbf{W}^{\mathsf{t}}$  t개 워드 배열의 전체 집합

**Z** 모든 정수의 집합

**Z**<sub>16</sub> 정수 환, **Z**<sub>16</sub> = {0, 1, ···, 15}

**LSH-8w-n** w 비트 워드 기반 n 비트 출력 LSH

**n** 해시함수 출력값의 비트 길이 (1 ≤ n ≤ 8w)

 $N_s$  압축 함수 내 단계 함수의 반복 회수

 $\mathbf{M}^{(i)}$  메시지 M을 블록 단위로 분할할 때 i 번째 메시지 블록

 $\mathbf{M_{i}}^{(i)}$  메시지 확장 함수을 이용하여 메시지 블록  $\mathbf{M}^{(i)}$ 로부터 유도된

16 워드 배열 (Ns+1)개 중에서 i 번째 배열

 ${
m CV}^{(i)}$  압축 단계의 i 번째 압축 함수 입력값을 저장하는 16 워드 배

열 크기의 중간 변수

SC; j 번째 단계 함수 계산에 사용되는 상수

T 단계 함수에서 사용되는 16 워드 배열 크기의 임시 변수

m 해시함수 입력 메시지

x || y 두 비트열 x와 y의 연접

X의 비트열을 Y에 대입시키는 연산, 또는 X가 함수일 경우, X

의 출력값을 Y에 대입시키는 연산

|x| 비트열 X의 비트 길이

x ⊕ y 같은 길이를 갖는 두 비트열 x와 y의 XOR

**X**[r:s] 비트열 x의 부분 비트열 x<sub>[r:s]</sub> := x<sub>r</sub>||x<sub>r+1</sub>||···||x<sub>s</sub> (r ≤ s)

{0.1}<sup>k</sup> 길이 k인 비트열 전체의 집합

X<sup>≪ r</sup> 워드 X를 왼쪽으로 r 비트만큼 순환 이동(Rotation)시킨 결과값

₩ 기 위드 X를 오른쪽으로 r 비트만큼 순환 이동(Rotation)시킨 결과

값 (X<sup>≫ r</sup> := X<sup>≪ (w-r)</sup>)

X[r:s] 워드 X의 부분 비트열 X[r:s] := x<sub>r</sub>||x<sub>r-1</sub>||···||x<sub>s</sub> (r ≥ s)

[x] 실수 x보다 크거나 같은 최소의 정수

[x] 실수 x보다 작거나 같은 최대의 정수

w 워드의 비트 길이 (w = 32 또는 64)

#### 3.4. 데이터 표기와 연산

#### 3.4.1. 비트열

비트열은 최상위 비트부터 표기하며, 비트 색인은 최상위 비트부터 0을 붙인다. 예를 들어 s 비트 비트열 x는  $x=x_0\|x_1\|\cdots\|x_{(s-1)}$   $(x_k\in\{0,1\},\ 0\le k\le (s-1))$ 과 같이 표기한다. 단, 예외적으로 워드를 비트열로 표현하는데 있어서는 최상위 비트에 비트 색인 (w-1)을 붙여  $x_{w-1}\|x_{w-2}\|\cdots\|x_0$   $(x_k\in\{0,1\},\ 0\le k\le (w-1))$ 과 같이 표기한다. 그리고 16진수(Hexadecimal) 표기를 4 비트열을 표현하기 위해 사용하며, 그 대응 관계는 <표 3-1>과 같다.

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

<표 3-1> 4 비트 비트열의 16진수 표현

#### 3.4.2. 바이트 배열

바이트 배열은 영어 알파벳 소문자로 표기하며, 배열의 각 요소는 '배열명[색인]'의 형태로 표현한다. 예를 들어 s 바이트 배열 z는  $z=(z[0],z[1],\cdots,z[s-1])와 같이 표기한다.$ 

#### 3.4.3. 워드 배열

워드 배열을 영어 알파벳 대문자로 표기하며, 배열의 각 요소는 '배열명[색인]'의 형태로 표현한다. 예를 들어 s 워드 배열  $X \leftarrow X = (X[0], X[1], \cdots, X[s-1])$ 와 같이 표기한다.

#### 3.4.4. 워드 연산

두 워드  $X=x_{(w-1)}\|\cdots\|x_0$   $(x_k\in\{0,1\},\ 0\le k\le (w-1))$ 와  $Y,\ 그리고 정수 <math>Z=z_{w-1}\cdot 2^{w-1}+z_{w-2}\cdot 2^{w-2}+\cdots+z_1\cdot 2+z_0$   $(z_k\in\{0,1\},\ 0\le k\le (w-1))$ 에 대해, 워드와 정수간의 변환 및 이를 이용한 워드 간의 연산을 다음과 같이 정의한다.

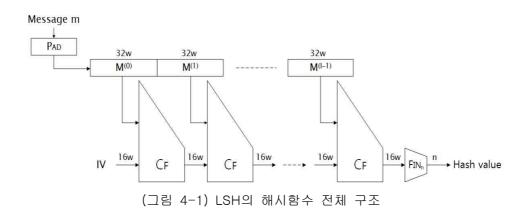
- $\bigcirc$  WordToInt(X) :=  $x_{w-1} \cdot 2^{w-1} + x_{w-2} \cdot 2^{w-2} + \cdots + x_1 \cdot 2 + x_0$
- $\bigcirc$  IntToWord(Z) :=  $z_{(w-1)} || \cdots || z_0$

 $\bigcirc$  X  $\boxplus$  Y := IntToWord((WordToInt(X) + WordToInt(Y)) mod 2<sup>w</sup>)

#### 4. 해시함수 LSH

#### 4.1. 전체 구조

LSH를 구성하는 각 해시함수는 (그림 4-1)과 같은 전체 구조를 가지며, 입력 메시지에 대해 다음의 세 가지 단계를 거쳐 해시값을 출력한다.



- 초기화(Initialization): 입력 메시지를 메시지 블록 비트 길이의 배수가 되도록 패딩을 한 후, 이를 메시지 블록 단위로 분할한다. 그리고 연결 변수를 IV로 초기화한다.
- 압축(Compression): 32 워드 배열 메시지 블록을 압축 함수의 입력으로 하여 얻은 출력값으로 연결 변수를 갱신하며, 이를 마지막 메시지 블록을 처리할 때까지 반복하여 메시지를 압축한다.
- 완료(Finalization): 압축 과정을 통해 연결 변수에 최종 저장된 값으로부터 n 비트 길이의 해시함수 출력값을 생성한다.

```
1 입력: 메시지 m
```

#### 2 처리 과정

3 출력 : h ∈ {0,1}<sup>n</sup>

해시함수 LSH-224, LSH-256, LSH-512-224, LSH-512-256, LSH-384, 그리고 LSH -512의 규격을 정리하면 <표 4-1>과 같다. LSH-224와 LSH-256은 기반 워드 길이 w 가 32이고, 함수 LSH-512-224, LSH-512-256, LSH-384, 그리고 LSH-512는 w가 64이다. (그림 4-1)에서 볼 수 있듯이 기반 워드 길이에 의해 연쇄 변수와 메시지 블록 비트 길이가 동일하게 설정되며, 압축 함수의 단계 수 N<sub>s</sub> 또한 w에 따라 다르게 설정된다.

구분	n	N <sub>s</sub>	연쇄 변수 비트 길이	메시지 블록 비트 길이	w
LSH-224	224	26	512	1024	32
LSH-256	256	26	312		32
LSH-512-224	224		1024	2048	
LSH-512-256	256	20			6.4
LSH-384	384	28			64
LSH-512	512				

<표 4-1> 해시함수 LSH 규격

아래에서는 입력 메시지를 처리하는 단계 별 과정을 구체적으로 정리한다.

#### 4.1.1. 초기화

해시함수의 입력 메시지를 m이라 하자. 먼저 m은 메시지 패딩 과정을 거친다. 메시지 패딩 과정은 m의 끝에 비트 '1'을 덧붙인 후, 길이가 32wt 비트가 될 때까지 비트 '0'을 덧붙인다. 여기에서  $t = \lceil (|m|+1)/32w \rceil$  이다.

메시지 패딩을 거친 메시지를  $m_p = m_0 \|m_1\| \cdots \|m_{32wt-1}$ 이라고 하자. 그러면  $m_p$ 는 4wt 바이트 배열  $m_a = (m[0], \cdots, m[4wt-1])$ 로 볼 수 있다. 여기에서  $m[k] = m_{8k} \|m_{8k+1}\| \cdots \|m_{8k+7}$  (0  $\leq k \leq (4wt-1)$ )이다. 바이트 배열  $m_a$ 는 식 (4.1)에 의해 32t 워드 배열  $M = (M[0], \cdots, M[32t-1])$ 으로 변환할 수 있다.

$$M[s] \leftarrow m[ws/8 + (w/8 - 1)] || \cdots || m[ws/8 + 1] || m[ws/8]$$
 (0 \le s \le (32t - 1)). (4.1)

이어서 워드 배열 M은 식 (4.2)와 같은 규칙에 따라 t개의 메시지 블록  $M^{(0)}, M^{(1)}, \cdots, M^{(t-1)}$ 로 분할할 수 있다.

$$M^{(i)} \leftarrow (M[32i], M[32i+1], \dots, M[32i+31])$$
  $(0 \le i \le (t-1)).$  (4.2)

연결 변수  $CV^{(0)}$  ( $\in W^{16}$ )는 4.3절에 제시된 초기화 상수(IV)를 이용하여 식 (4.3)과 같이 배열 색인별로 값을 할당하는 방식으로 초기화 한다.

$$CV^{(0)}[I] \leftarrow IV[I] \qquad (0 \le I \le 15).$$
 (4.3)

#### 4.1.2. 압축

초기화 단계에서 생성된 t개의 메시지 블록은 압축 단계에서 순차적으로 압축 함수 CF :  $W^{16} \times W^{32} \to W^{16}$ 의 입력값으로 사용된다. 압축 함수 CF는 연결 변수값  $CV^{(i)}$ 와 메시지 블록  $M^{(i)}$ 을 입력으로 받아 연결 변수값  $CV^{(i+1)}$ 을 반환한다. 압축 함수 CF는 4.2절에서 구체적으로 다룬다.

#### 4.1.3. 완료

완료 함수  $FIN_n: W^{16} \to \{0,1\}^n$ 은 압축 과정의 결과로 생성된 마지막 연결 변수값  $CV^{(t)}$  =  $(CV^{(t)}[0], \cdots, CV^{(t)}[15])$ 에 적용되어 n 비트 길이의 해시값 h를 생성한다. H =  $(H[0], \cdots, H[7])$ 를 8 워드 배열,  $h_b$  =  $(h_b[0], \cdots, h_b[w-1])$ 는 w 바이트 배열이라고 하면, 출력 함수  $FIN_n$ 은 식 (4.4)의 절차를 수행한다.

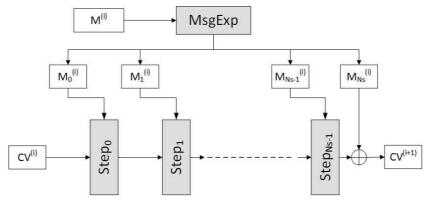
$$\begin{split} H[I] &\leftarrow CV^{(t)}[I] \oplus CV^{(t)}[I+8] & (0 \leq I \leq 7), \\ h_b[s] &\leftarrow H[\ [\ 8s/w\ ]\ ]^{\gg (8s \bmod w)}_{[7:0]} & (0 \leq s \leq (w-1)), \\ h &\leftarrow (h_b[0]||\cdots||h_b[w-1])_{[0:n-1]}. \end{split} \tag{4.4}$$

#### 4.2. 압축 함수

해시 연산에서 수행되는 i 번째 압축 함수  $CF:W^{16}\times W^{32}\to W^{16}$ 는 연결 변수  $CV^{(i)}$  ( $\in W^{16}$ )와 메시지 블록  $M^{(i)}$  ( $\in W^{32}$ )를 입력으로 받는다. 압축 함수 CF는 다음 네 가지 함수로 구성된다.

- 메시지 확장 함수 MsgExp: W<sup>32</sup> → W<sup>16(Ns+1)</sup>
- 메시지 덧셈 함수 MsgAdd: W<sup>16</sup> × W<sup>16</sup> → W<sup>16</sup>
- 섞음 함수 Mix<sub>i</sub>: W<sup>16</sup> → W<sup>16</sup>
- $\bigcirc$  워드 단위 순환(Permutation) 함수 WordPerm :  $W^{16} \rightarrow W^{16}$

압축 함수의 전체 구조를 도시하면 (그림 4-2)와 같다.



(그림 4-2) LSH의 압축 함수 전체 구조

압축 함수의 입력값 중 메시지 블록  $M^{(i)}$ 는 메시지 확장 함수 MsgExp를 거쳐  $(N_s+1)$ 개의 16 워드 배열  $M_j^{(i)}$   $(0 \le j \le N_s)$ 로 확장된다. 이어서 16 워드 배열 크기의 임시 변수  $T=(T[0],\cdots,T[15])$ 에 초기값을  $CV^{(i)}$ 로 할당한 후, 순차적으로 단계 함수  $Step_j$ 를 통해  $M_j^{(i)}$ 를 처리하면서 T를 갱신한다. 단계 함수를 거쳐 T에 저장된 값은 MsgAdd 함수를 통해 처리된 후 (i+1) 번째 단계 변수  $CV^{(i+1)}$ 에 입력된다. 이러한 압축 함수 처리 절차를 정리하면 다음과 같다.

```
1 입력 : 단계 변수값 CV<sup>(i)</sup> (∈ W<sup>32</sup>)와 메시지 블록 M<sup>(i)</sup> (∈ W<sup>32</sup>)
```

2 처리 과정

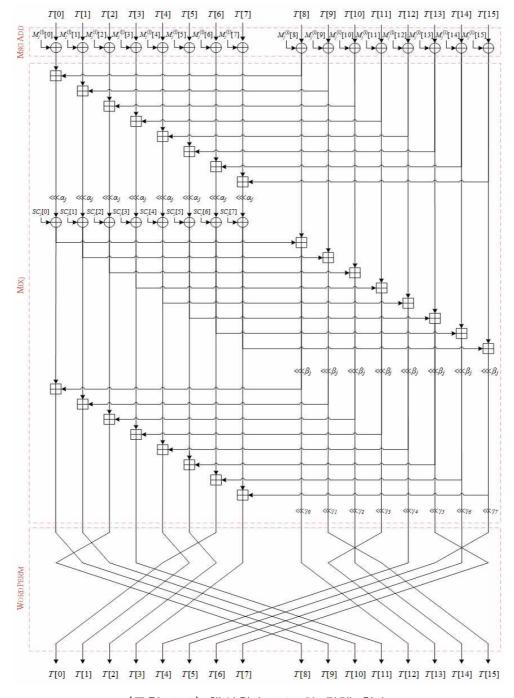
$$\begin{split} \{M_j^{(i)}\}_{0 \leq j \leq N_S} &\leftarrow \mathsf{MsgExp}(\mathsf{M}^{(i)}); \\ \mathsf{T} &\leftarrow \mathsf{CV}^{(i)}; \\ \mathsf{for} \ j = 0 \ \mathsf{to} \ (\mathsf{N}_S - 1) \ \mathsf{do} \\ &\quad \mathsf{T} &\leftarrow \mathsf{Step}_j(\mathsf{T}, \mathsf{M}_j^{(i)}); \ / / \ \mathsf{T} &\leftarrow \mathsf{MsgAdd}(\mathsf{T}, \mathsf{M}_j^{(i)}); \ \mathsf{T} &\leftarrow \mathsf{Mix}_j(\mathsf{T}); \ \mathsf{T} &\leftarrow \mathsf{WordPerm}(\mathsf{T}); \\ \mathsf{end} \ \mathsf{for} \\ &\quad \mathsf{CV}^{(i+1)} &\leftarrow \mathsf{MsgAdd}(\mathsf{T}, \ \mathsf{M}_{\mathsf{N}_S}^{\ (i)}); \end{split}$$

3 출력 : 단계 변수 CV<sup>(i+1)</sup> (∈ W<sup>32</sup>)

압축 함수에서 순차적으로  $M_j^{(i)}$ 를 처리하는 단계 함수  $Step_j: W^{16} \times W^{16} \to W^{16}$ 은 식 (4.5)와 같다.

Step<sub>i</sub> := WordPerm 
$$\circ$$
 Mix<sub>i</sub>  $\circ$  MsgAdd  $(0 \le j \le (N_s - 1))$ . (4.5)

단계 함수의 전체 구조는 (그림 4-3)과 같다.



(그림 4-3) 해시함수 LSH의 단계 함수

아래에서는 압축 함수를 구성하는 세부 함수를 규정한다.

#### 4.2.1. 메시지 확장 함수 MsgExp

압축 함수에 입력된 i 번째 메시지 블록  $M^{(i)}=(M^{(i)}[0],\cdots,M^{(i)}[31])$ 에 대해, 메시지 확장 함수 MsgExp는  $(N_s+1)$ 개의 16 워드 배열  $M_j^{(i)}$   $(0 \le j \le N_s)$ 를 생성한다. 생성 방법은 식 (4.6)과 같다.

$$\begin{split} M_0^{(i)} &\leftarrow (M^{(i)}[0], M^{(i)}[1], \cdots, M^{(i)}[15]), \\ M_1^{(i)} &\leftarrow (M^{(i)}[16], M^{(i)}[17], \cdots, M^{(i)}[31]), \\ M_i^{(i)}[I] &\leftarrow M_{i-1}^{(i)}[I] \boxplus M_{i-2}^{(i)}[\tau(I)] \qquad (0 \leq I \leq 15, \ 2 \leq j \leq N_s). \end{split}$$

식 (4.6)에 사용된 함수 τ는 <표 4-2>와 같이 정의된 ℤ₁6 상의 치환(Permutation)이다.

<표 4-2> ℤ<sub>16</sub> 상의 치환 τ

- 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
τ(Ι)	3	2	0	1	7	4	5	6	11	10	8	9	15	12	13	14

#### 4.2.2. 메시지 덧셈 함수 MsgAdd

두 개의 16 워드 배열 X = (X[0], ···, X[15])와 Y = (Y[0], ···, Y[15])에 대해, 메시지 덧셈 함수 MsqAdd: W<sup>16</sup> × W<sup>16</sup> → W<sup>16</sup>는 식 (4.7)과 같이 정의한다.

$$MsgAdd(X, Y) := (X[0] \oplus Y[0], \dots, X[15] \oplus Y[15]).$$
 (4.7)

#### 4.2.3. 섞음 함수 Mix<sub>i</sub>

섞음 함수  $Mix_j: W^{16} \to W^{16}$ 는 입력된 16 워드 배열  $T=(T[0], \cdots, T[15])$ 에 대해 두 개의 워드 T[I], T[I+8] (0  $\leq I \leq 7$ )을 쌍으로 구성한 후 식 (4.8)과 같이 각각 섞어주는 방식으로 T를 갱신한다.

$$(T[I], T[I+8]) \leftarrow Mix_{j,l}(T[I], T[I+8]) \qquad (0 \le I \le 7)$$
 (4.8)

여기에서  $\mathrm{Mix_{j,l}}\colon W^2\to W^2$ 는 두 개 워드 X, Y를 처리하는 섞음 함수로 다음의 절차를 따르며, 이를 도시하면 (그림 4-4)와 같다.

#### 1 입력: 워드 X, Y

#### 2 처리 과정

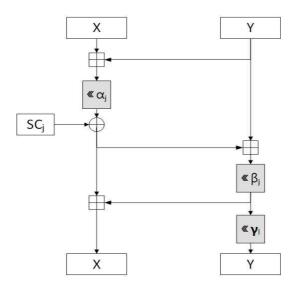
 $X \leftarrow X \boxplus Y; \qquad X \leftarrow X^{\ll \alpha_j};$ 

 $X \leftarrow X \oplus SC_{i}[1];$ 

 $Y \leftarrow X \boxplus Y; \qquad Y \leftarrow Y^{\otimes \beta_j};$ 

 $X \leftarrow X \boxplus Y; \qquad Y \leftarrow Y^{\ll y_{|}};$ 

3 출력: 워드 X, Y



(그림 4-4) 해시함수 LSH의 섞음 함수

섞음 함수  $Mix_{j,l}$ 에 사용되는 비트 순환량  $\alpha_j$ ,  $\beta_j$ ,  $\gamma_l$ 은 <표 4-3>과 같다. 비트 순환량  $\alpha_j$ ,  $\beta_i$ 는 짝수 단계와 홀수 단계에서 다른 값이 적용된다.

W j  $\alpha_{j} \\$ Уo **V**1 **y**2 ۷з **V**4 **Y**5 86 **V**7 짝수 29 1 0 8 24 8 0 32 16 24 16 홀수 5 17 짝수 23 59 0 24 40 64 16 32 48 8 56 7 홀수

<표 4-3> 비트 순환량 αj, βj, γl

그리고 8 워드 배열의 단계 상수  $SC_j = (SC_j[0], \cdots, SC_j[7])$ 는 먼저  $SC_0$ 를 <표 4-4>와 같이 정의한 후, 식 (4.9)에 따라 나머지 (N<sub>s</sub>-1)개의 상수  $SC_j$ 를 유도하여 사용한다.

<표 4-4> 초기 단계 상수

	w = 32	w = 64
SC <sub>0</sub> [0]	917caf90	97884283c938982a
SC <sub>0</sub> [1]	6c1b10a2	ba1fca93533e2355
SC <sub>0</sub> [2]	6f352943	c519a2e87aeb1c03
SC <sub>0</sub> [3]	cf778243	9a0fc95462af17b1
SC <sub>0</sub> [4]	2ceb7472	fc3dda8ab019a82b
SC <sub>0</sub> [5]	29e96ff2	02825d079a895407
SC <sub>0</sub> [6]	8a9ba428	79f2d0a7ee06a6f7
SC <sub>0</sub> [7]	2eeb2642	d76d15eed9fdf5fe

$$SC_{i}[I] \leftarrow SC_{i-1}[I] \boxplus SC_{i-1}[I]^{\otimes 8} \quad (0 \le I \le 7, 1 \le j \le (N_{s}-1)) \quad (4.9)$$

#### 4.2.4. 워드 단위 순환 함수 WordPerm

워드 단위 순환 함수 WordPerm: W<sup>16</sup> → W<sup>16</sup>은 주어진 16 워드 배열 X = (X[0], ···, X[15])에 대해 식 (4.10)과 같이 정의한다.

$$WordPerm(X) := (X[\sigma(0)], \dots, X[\sigma(15)]). \tag{4.10}$$

식 (4.10)에서 사용된 함수 σ는 <표 4-5>와 같이 정의된 ℤ<sub>16</sub> 상의 치환이다.

<표 4-5> ℤ<sub>16</sub> 상의 치환 σ

I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ(I)	6	4	5	7	12	15	14	13	2	0	1	3	8	11	10	9

#### 4.3. 초기화 상수

해시함수의 초기화 상수는 해시값의 길이 n과 단위 워드 길이 w에 의존하여 정의한다. 16 워드 배열 X = (X[0], ···, X[15])와 32 워드 배열 Y = (Y[0], ···, Y[31])가 다음과 같이 정의되어 있다고 하자.

- $\bigcirc X[0] = IntToWord(w)$
- $\bigcirc$  X[1] = IntToWord(n)
- $\bigcirc X[I] = 0^{w} (2 \le I \le 15)$
- $\bigcirc Y[k] = 0^{w} (0 \le k \le 31)$

그러면 LSH의 각 해시함수에 대한 초기화 상수는 식 (4.11)에 의해 결정된다.

$$IV \leftarrow CF(X, Y). \tag{4.11}$$

아래에서는 식 (4.11)에 의해 계산된 LSH-224, LSH-256, LSH-512-224, LSH-512-256, LSH-384, LSH-512의 초기화 상수를 명시한다.

## 4.3.1. LSH-256-224 (LSH-224)

LSH-224의 초기화 상수 512 비트는 다음과 같다.

IV[0]	IV[1]	IV[2]	IV[3]	IV[4]	IV[5]	IV[6]	IV[7]
068608D3	62D8F7A7	D76652AB	4C600A43	BDC40AA8	1ECA0B68	DA1A89BE	3147D354
IV[8]	IV[9]	IV[10]	IV[11]	IV[12]	IV[13]	IV[14]	IV[15]
707EB4F9	F65B3862	6B0B2ABE	56B8EC0A	CF237286	EE0D1727	33636595	8BB8D05F

### 4.3.2. LSH-256-256 (LSH-256)

LSH-256의 초기화 상수 512 비트는 다음과 같다.

IV[0]	IV[1]	IV[2]	IV[3]	IV[4]	IV[5]	IV[6]	IV[7]
46A10F1F	FDDCE486	B41443A8	198E6B9D	3304388D	B0F5A3C7	B36061C4	7ADBD553
IV[8]	IV[9]	IV[10]	IV[11]	IV[12]	IV[13]	IV[14]	IV[15]
105D5378	2F74DE54	5C2F2D95	F2553FBE	8051357A	138668C8	47AA4484	E01AFB41

#### 4.3.3. LSH-512-224

LSH-512-224의 초기화 상수 1,024 비트는 다음과 같다.

IV[0]	IV[1]	IV[2]	IV[3]	
0C401E9FE8813A55	4A5F446268FD3D35	FF13E452334F612A	F8227661037E354A	
IV[4]	IV[5]	IV[6]	IV[7]	
A5F223723C9CA29D	95D965A11AED3979	01E23835B9AB02CC	52D49CBAD5B30616	
IV[8]	IV[9]	IV[10]	IV[11]	
9E5C2027773F4ED3	66A5C8801925B701	22BBC85B4C6779D9	C13171A42C559C23	
IV[12]	IV[13]	IV[14]	IV[15]	
31E2B67D25BE3813	D522C4DEED8E4D83	A79F5509B43FBAFE	E00D2CD88B4B6C6A	

## 4.3.4. LSH-512-256

LSH-512-256의 초기화 상수 1,024 비트는 다음과 같다.

IV[0]	IV[1]	IV[2]	IV[3]	
6DC57C33DF989423	D8EA7F6E8342C199	76DF8356F8603AC4	40F1B44DE838223A	
IV[4]	IV[5]	IV[6]	IV[7]	
39FFE7CFC31484CD	39C4326CC5281548	8A2FF85A346045D8	FF202AA46DBDD61E	
IV[8]	IV[9]	IV[10]	IV[11]	
CF785B3CD5FCDB8B	1F0323B64A8150BF	FF75D972F29EA355	2E567F30BF1CA9E1	
IV[12]	IV[13]	IV[14]	IV[15]	
B596875BF8FF6DBA	FCCA39B089EF4615	ECFF4017D020B4B6	7E77384C772ED802	

## 4.3.5. LSH-384 (LSH-512-384)

LSH-384의 초기화 상수 1,024 비트는 다음과 같다.

IV[0]	IV[1]	IV[2]	IV[3]	
53156A66292808F6	B2C4F362B204C2BC	B84B7213BFA05C4E	976CEB7C1B299F73	
IV[4]	IV[5]	IV[6]	IV[7]	
DF0CC63C0570AE97	DA4441BAA486CE3F	6559F5D9B5F2ACC2	22DACF19B4B52A16	
IV[8]	IV[9]	IV[10]	IV[11]	
BBCDACEFDE80953A	C9891A2879725B3E	7C9FE6330237E440	A30BA550553F7431	
IV[12]	IV[13]	IV[14]	IV[15]	
BB08043FB34E3E30	AODEC48D54618EAD	150317267464BC57	32D1501FDE63DC93	

## 4.3.6. LSH-512 (LSH-512-512)

LSH-512의 초기화 상수 1,024 비트는 다음과 같다.

IV[0]	IV[1]	IV[2]	IV[3]	
ADD50F3C7F07094E	E3F3CEE8F9418A4F	B527ECDE5B3D0AE9	2EF6DEC68076F501	
IV[4]	IV[5]	IV[6]	IV[7]	
8CB994CAE5ACA216	FBB9EAE4BBA48CC7	650A526174725FEA	1F9A61A73F8D8085	
IV[8]	IV[9]	IV[10]	IV[11]	
B6607378173B539B	1BC99853B0C0B9ED	DF727FC19B182D47	DBEF360CF893A457	
IV[12]	IV[13]	IV[14]	IV[15]	
4981F5E570147E80	D00C4490CA7D3E30	5D73940C0E4AE1EC	894085E2EDB2D819	