해시함수 LSH

https://youtu.be/YO764pePJjo

IT융합공학부 송경주

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

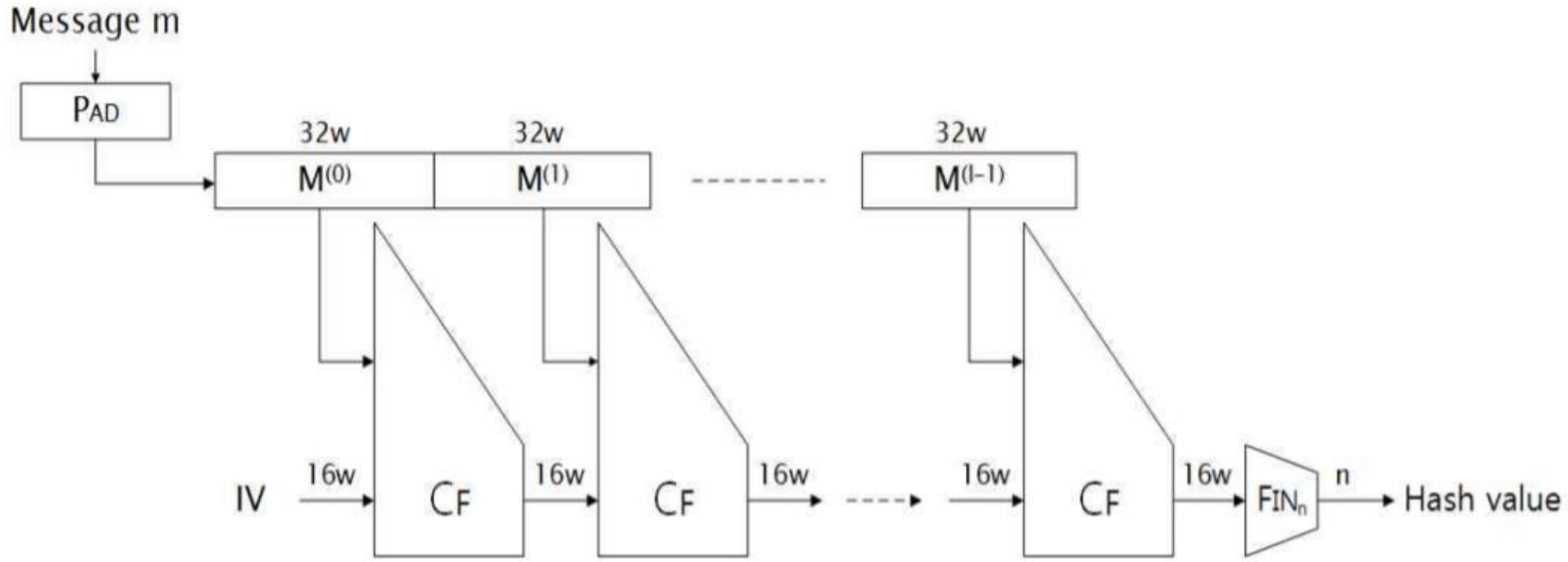
Contents

CryptoCraft LAB

해시함수 LSH

LSH (Lightweight Secure Hash)는 2014년 국가보안기술연구소에서 개발한 해시함수로, 높은 안전성과 우수한 효율성을 제공하는 해시함수. w 비트 워드 단위로 동작하고 n 비트 출력값을 가지는 해시함수 LSH-8w-n 으로 구성된다.

해시함수 LSH - 전체구조

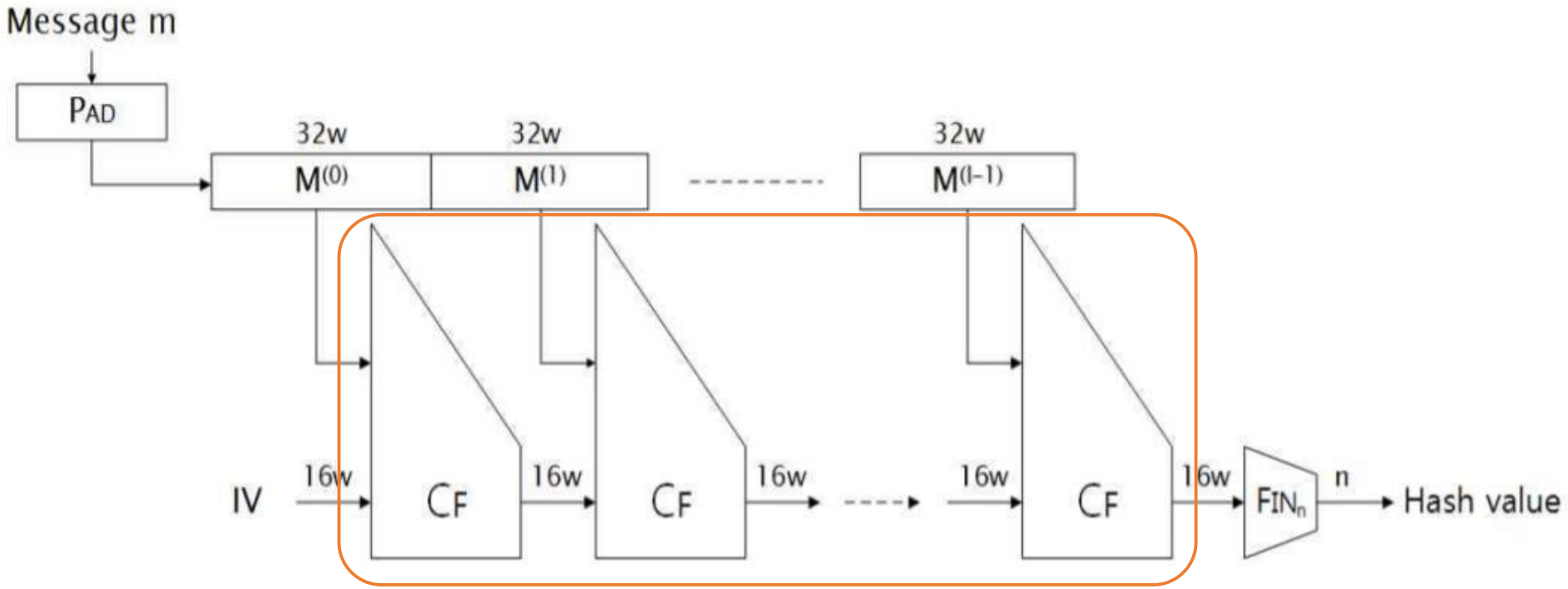


초기화(Initialization): 입력 메시지를 메시지 블록 비트 길이의 배수가 되도록 패딩을 한 후, 이를 메시지 블록 단위로 분할한다. 그리고 연결 변수를 IV로 초기화한다.

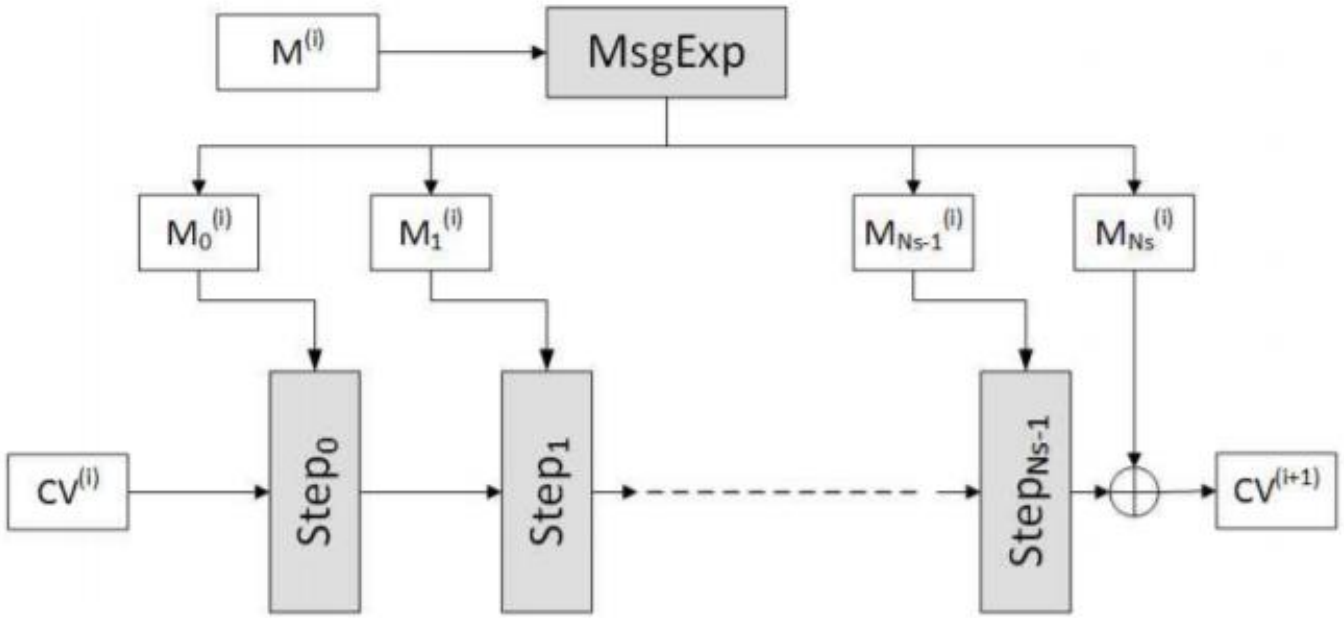
압축(Compression): 32 워드 배열 메시지 블록을 압축 함수의 입력으로 하여 얻은 출력값으로 연결 변수를 갱신하며, 이를 마지막 메시지 블록을 처리할 때까지 반복 하여 메시지를 압축한다.

완료(Finalization): 압축 과정을 통해 연결 변수에 최종 저장된 값으로부터 n 비트 길이의 해시함수 출력값을 생성한다.

해시함수 LSH - 전체구조



해시함수 LSH - 압축함수



1 입력 : 메시지 m

2 처리 과정

$m_p \leftarrow pad(m)$; // 메시지 패딩 처리

m_p로부터 메시지 블록 ($M^{(0)}, M^{(1)}, \cdots, M^{(t-1)}$) 생성;

$CV^{(0)} \leftarrow IV$; // 초기화 단계 끝

for i = 0 to (t − 1) do

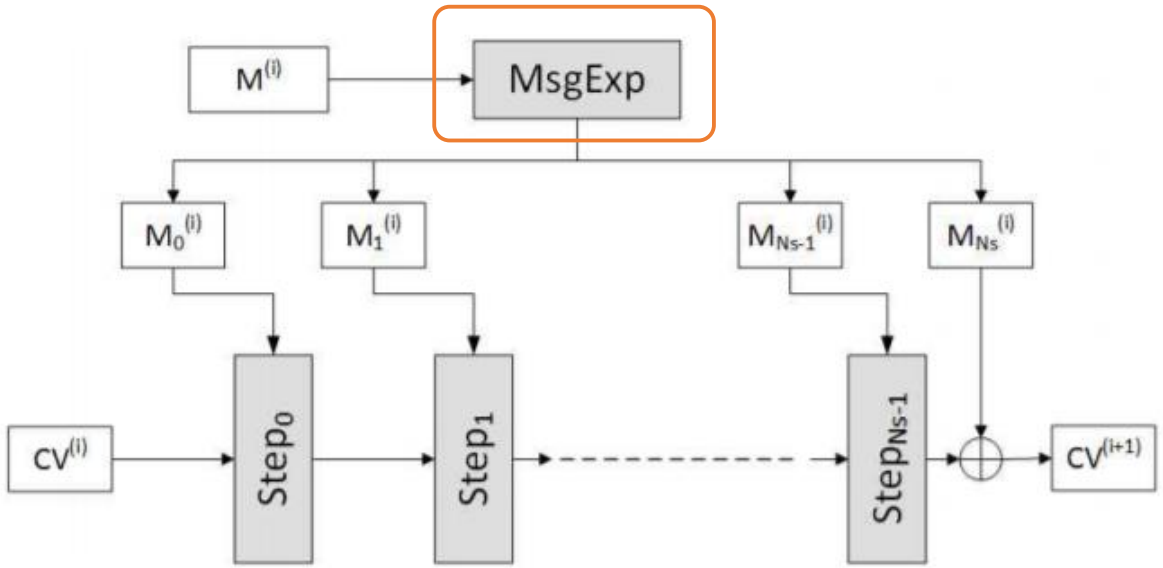
$CV^{(i+1)} \leftarrow CF(CV^{(i)}, M^{(i)})$;

end for // 압축 단계 끝

$h \leftarrow FIN_n(CV^{(t)})$; // 완료 단계 끝

3 출력 : $h \in \{0,1\}^n$

해시함수 LSH - 압축함수

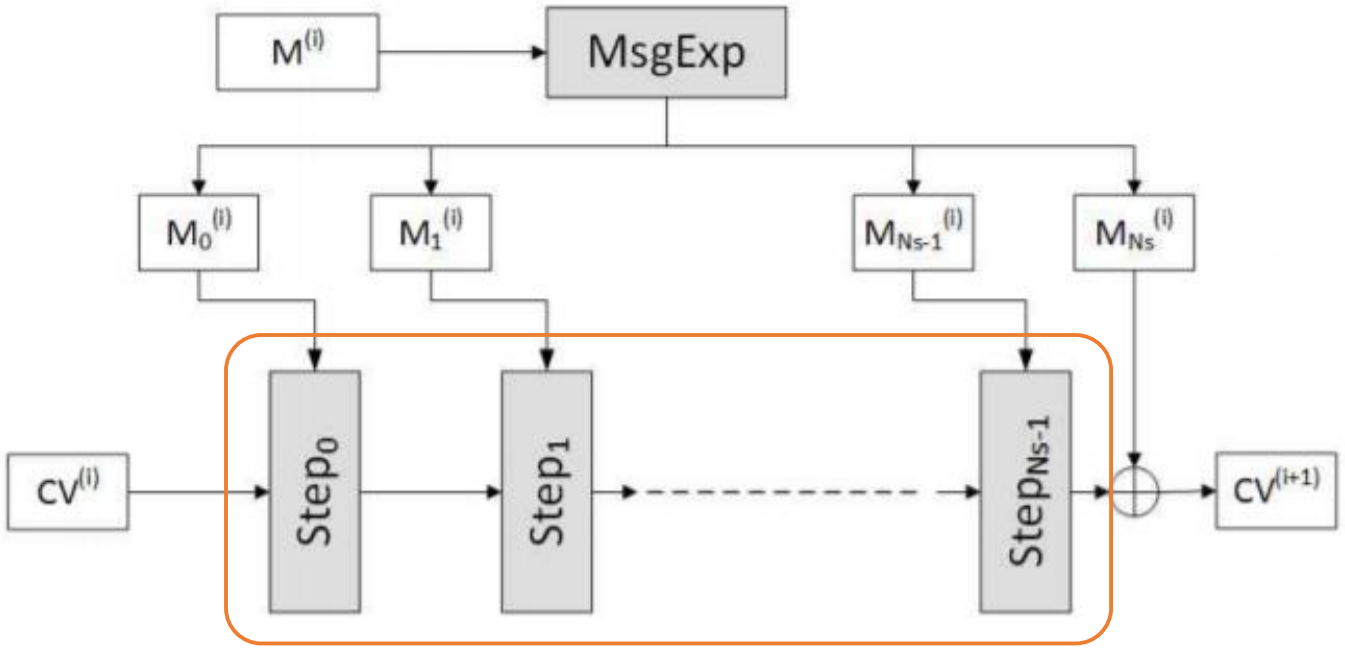


$M_0^{(i)} \leftarrow (M^{(i)}[0], M^{(i)}[1], \cdots, M^{(i)}[15]),$

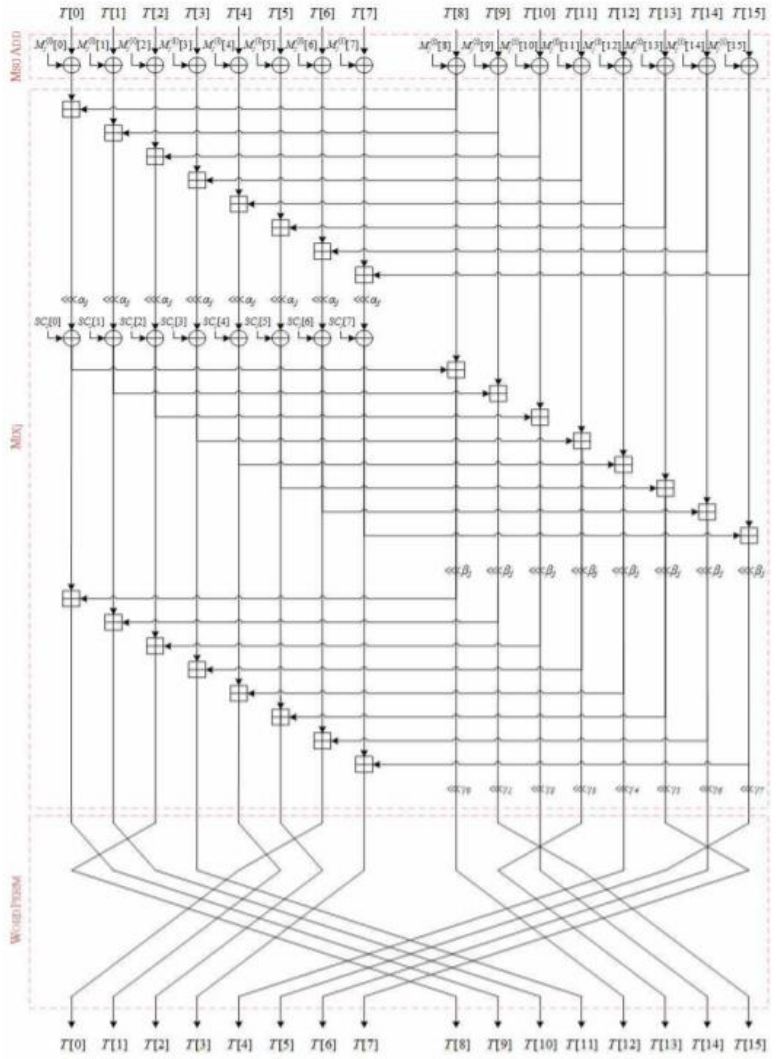
$M_1^{(i)} \leftarrow (M^{(i)}[16], M^{(i)}[17], \cdots, M^{(i)}[31]),$

$M_j^{(i)}[I] \leftarrow M_{j-1}^{(i)}[I] \boxplus M_{j-2}^{(i)}[\tau(I)] \qquad (0 \leq I \leq 15,\ 2 \leq j \leq N_s).$

해시함수 LSH - 압축함수



해시함수 LSH - 단계함수

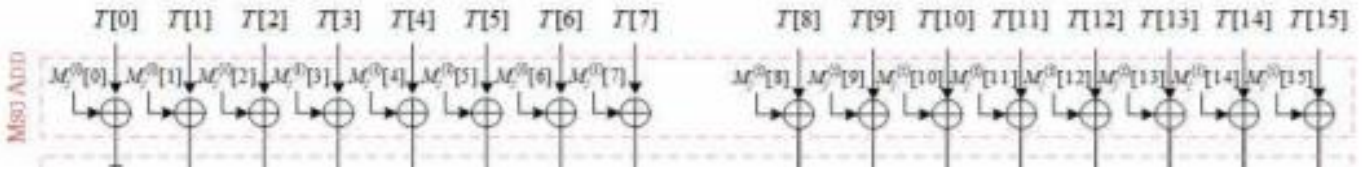


메시지 덧셈 함수 (MsgAdd)

섞음 함수 (Mix)

워드 단위 순환 함수 (WordPerm)

해시함수 LSH – 메시지 덧셈 함수(MsgAdd)



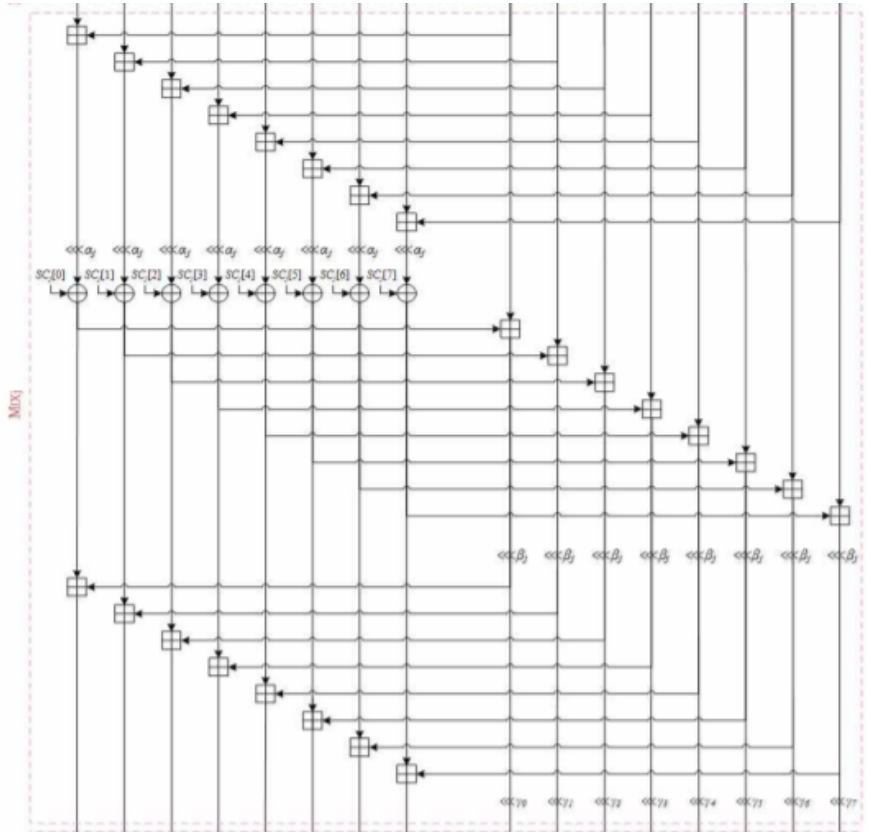
$$\mathrm{MsgAdd}(X,\ Y) := (X[0] \oplus Y[0], \cdots, X[15] \oplus Y[15]).$$

[메시지 덧셈 함수 (MsgAdd)]

512bit로 들어온 메시지를 1word(32bit)씩 16개로 나눠 함수를 수행함.

메시지 512bit를 1word 단위로 나눈 M[0]~M[15] 와 32bit 단계상수 T_0~T_{15} 를 Ripple-carry-add 한다.

해시함수 LSH – 섞음 함수(Mix)



[섞음 함수 (Mix)]

두 개의 워드 쌍 T[i], T[i+8] (0≤i≤7) 으로 Mix 함수가 동작된다.

해시함수 LSH – 섞음 함수(Mix)

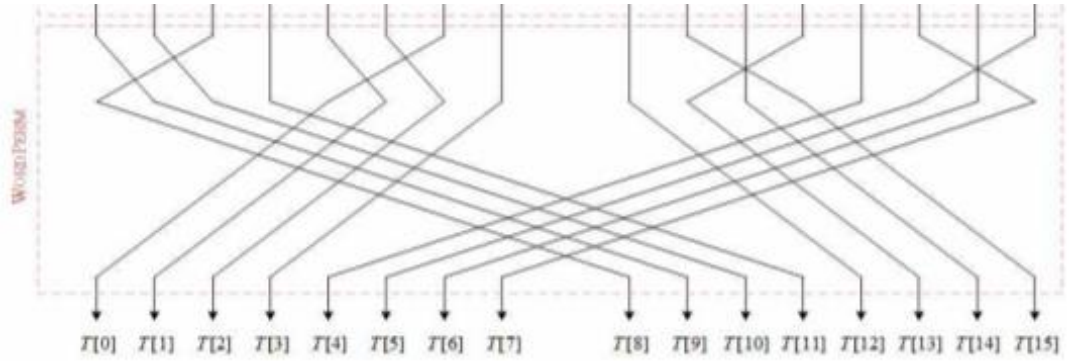
Algorithm 1 : Quantum circuit implementation of Mix

Input: T[i], T[i+8], SC[i] (0≤i≤7)
Output: T = {T[0] ⋯ T[15]}
1: **ripple_carry_add** (T[i], T[i+8])
2: **a_rotation**(T[i])
3: **Applying X gate** to T[i] according to SC[i]
5: **ripple_carry_add**(T[i], T[i+8])
6: **b_rotation**(T[i+8])
7: **ripple_carry_add**(T[i+8], T[i])
8: **c_rotation**(T[i+8])
9: **return** T = {T[0] ⋯ T[15]}

[섞음 함수 (Mix)]

두 개의 워드 쌍 T[i], T[i+8] (0≤i≤7) 으로 한번의 Mix를 동작하며 총 8번의 Mix 함수가 사용된다.

해시함수 LSH – 워드 단위 순환 함수(WordPerm)

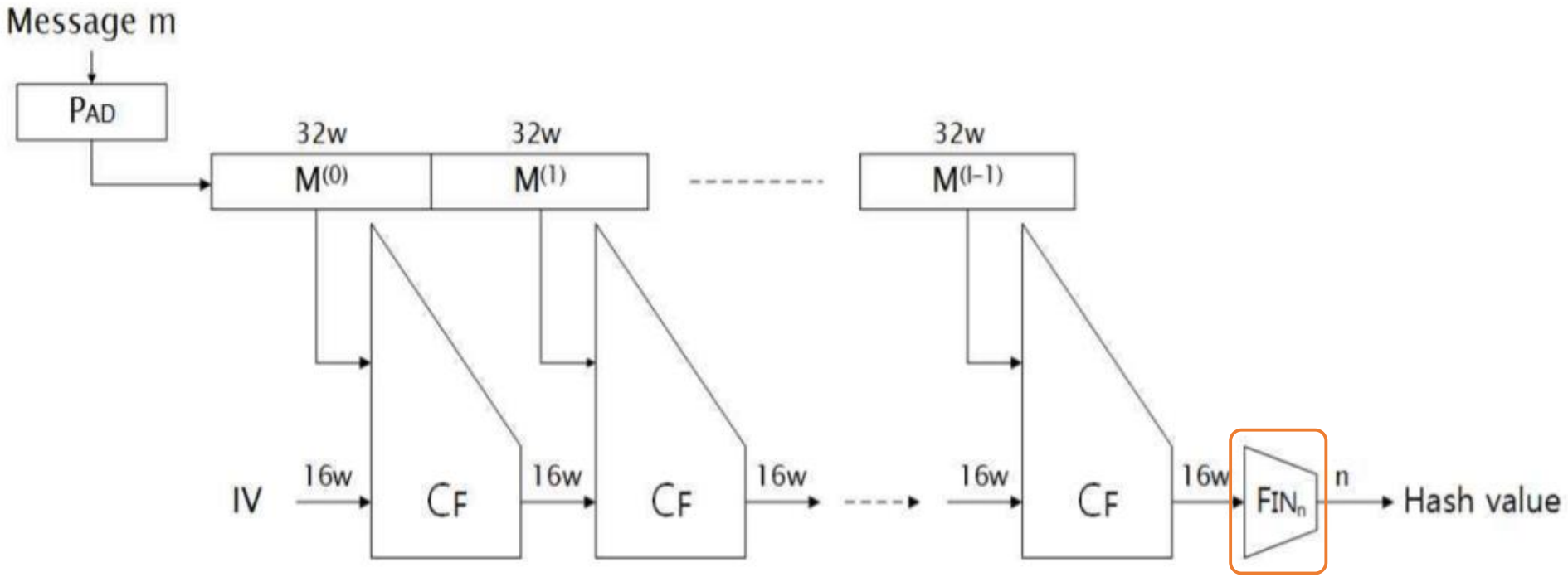


I	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ(I)	6	4	5	7	12	15	14	13	2	0	1	3	8	11	10	9

[워드 단위 순환 함수 (WordPerm)]

Swap을 통해 치환.

해시함수 LSH - 전체구조



해시함수 LSH - Final

Algorithm 2 : Quantum circuit implementation of Final

Input: CV[i], CV[i+8] (0≤i≤7)
Output: h = { h[0] , ⋯ , h[256] }

```
1: for k = 0 to 7
2:     for i = 0 to 31
3:         CNOT (CV_{K+8}[i], CV_K[i])
4: for k = 0 to 7
5:     for i = 0 to 7
6:         CNOT (CV_K[7-i], h_K[i])
7:     for i = 0 to 0
8:         for j = 0 to 31
9:             Swap (CV_K[i], CV_K[i+1])
10:    for i = 0 to 7
11:        CNOT (CV_K[7-i], h[8*(k*4)+i])
12:    for i = 0 to 0
13:        for j = 0 to 31
14:            Swap (CV_K[i], CV_K[i+1])
15:    for i = 0 to 7
16:        CNOT (CV_K[7-i], h[16*(k*4)+i])
17:    for i = 0 to 0
18:        for j = 0 to 31
19:            Swap (CV_K[i], CV_K[i+1])
20:    for i = 0 to 7
21:        CNOT (CV_K[7-i], h[24*(k*4)+i])
```

$$H[I] \leftarrow CV^{(t)}[I] \oplus CV^{(t)}[I+8] \qquad (0 \le I \le 7),$$

$$h_b[s] \leftarrow H[\,\lfloor 8s/w \rfloor\,]^{\gg(8s \bmod w)}_{[7:0]} \qquad (0 \le s \le (w-1)),$$

$$h \leftarrow (h_b[0] \| \cdots \| h_b[w-1])_{[0:n-1]}.$$

해시함수 LSH - 양자 회로 구현을 통한 자원 추정 결과

```
Gate counts:
    Allocate : 1918
    CCX : 63488
    CX : 145408
    Deallocate : 1918
    Measure : 256
    Swap : 344392
    X : 373
```

	Qubits	Toffoli gates	CNOT gates
LSH-256/256	1,918	63,488	145,408

Q & A