SHA3 양자회로 구현

https://youtu.be/6VgPPAVsxBg

IT융합공학부 송경주

Secure Hash Algorithm(SHA)-3

- Secure Hash Algorithm(SHA)-3
  - Secure Hash Algorithm(SHA)-3는 SHA1, SHA2를 대체하기 위해 2015년 National Institute of Standards and Technology (NIST) 에서 공개한 해시함수

- 해시함수 공격 방법 : Preimage attack
  - Preimage → 해시가 주어졌을 때, 원본 메시지를 찾는 방법

  (H = hash(M) 에 대해 H가 주어졌을 때, M을 찾음)
  - n-bit의 해시는 n-bit의 preimage 저항을 가짐

  → 좋은 해시함수일수록 preimage 를 찾는 것이 매우 어려움

Secure Hash Algorithm(SHA)-3

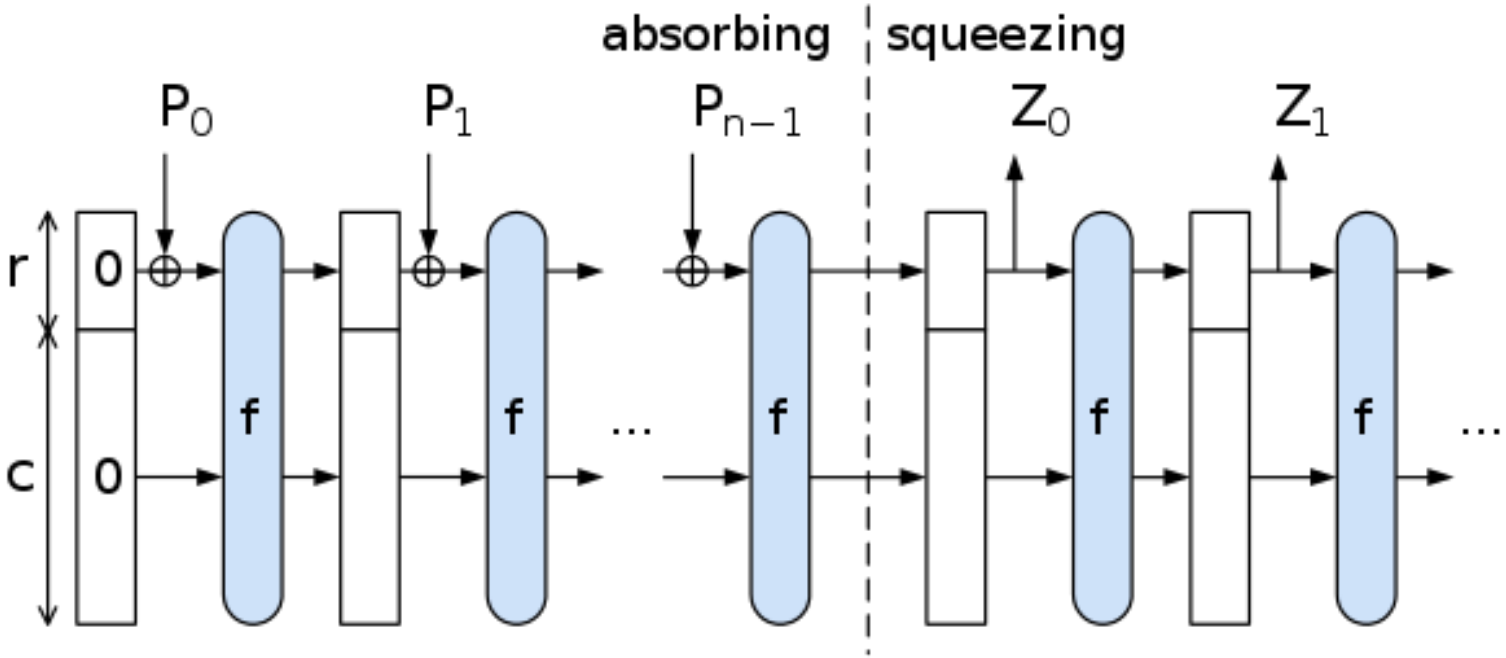
- SHA3 출력 길이 n : 224, 256, 384, 512 bit
  - 충돌 저항성 : $2^{n/2}$
  - Pre-image 저항성 : 2^{n}

구분	출력 크기	메시지 크기	b	r	c	워드	라운드 수
SHA3 -224	224 -bit	무제한	1600 -bit	1152 -bit	448 -bit	64 -bit	64
SHA3 -256	256 -bit	무제한	1600 -bit	1088-bit	512 -bit	64 -bit	64
SHA3 -384	384 -bit	무제한	1600 -bit	832 -bit	768 -bit	64 -bit	80
SHA3 -512	512 -bit	무제한	1600 -bit	576 -bit	1024 -bit	64 -bit	80

Secure Hash Algorithm(SHA)-3

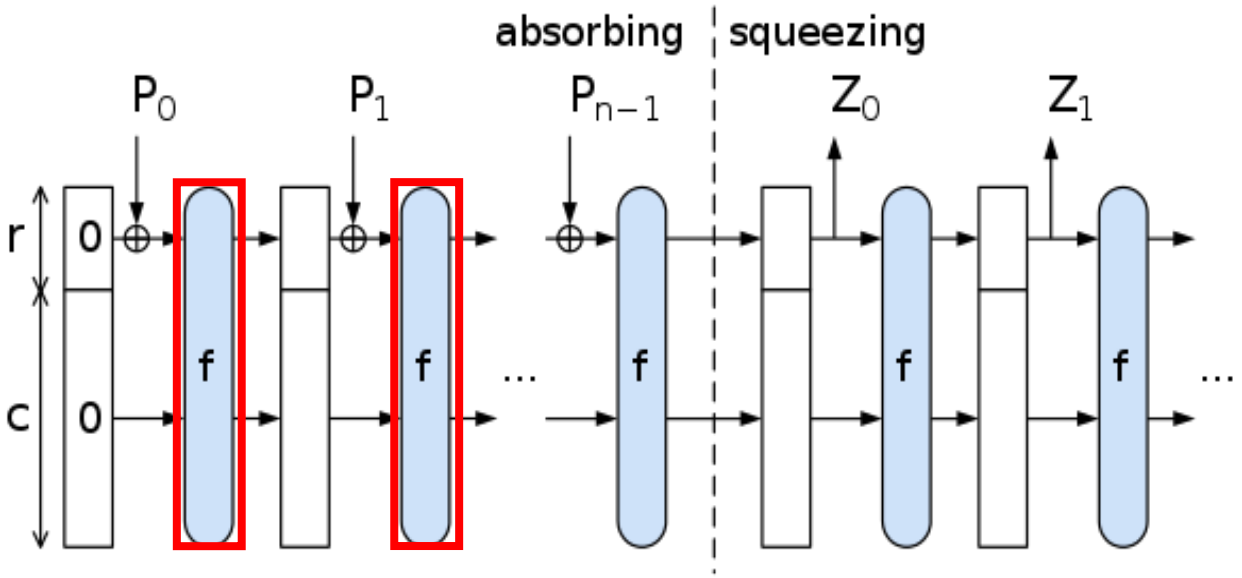
[SHA3 구조]

- Sponge 구조를 사용하여 동작
  : 입력된 데이터가 스펀지 구조에 의해 ‘absorbed’ 및 ‘squeezed’ 단계를 통해 결과를 출력함
- Absorbing : 메시지 블록이 XOR되어 permutation 함수를 통해 변환됨
- Squeezing : output 블록을 함수 f의 반복으로 업데이트함



Secure Hash Algorithm(SHA)-3

- SHA3 f function : 5개의 단계로 동작
  - $\theta(theta),\ \rho(rho),\ \pi(pi),\ \chi(chi),\ \iota(iota)$
- 입력 b비트에 따라 $12 + 2l$만큼의 라운드로 f함수가 진행됨
- SHA3 State = $5\times5\times\omega$ 의 3차원 행렬



b(bit)	25	50	100	200	400	800	1600
w	1	2	4	8	16	32	64
l	0	1	2	3	4	5	6

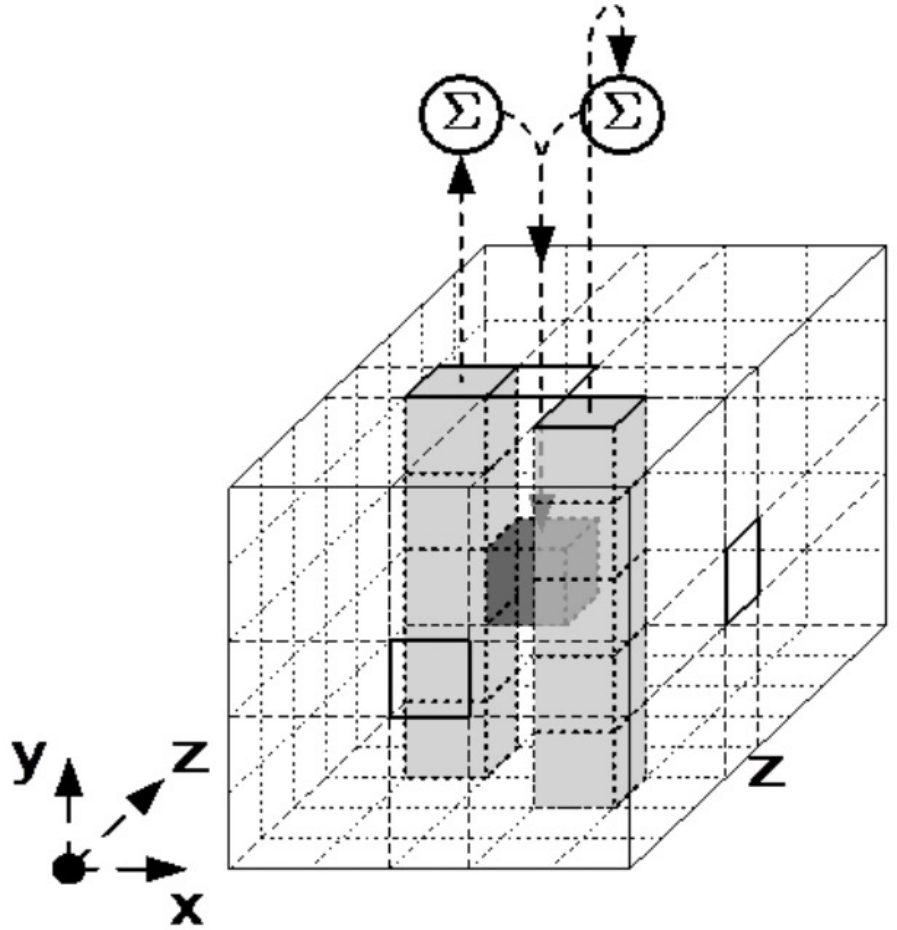
< b비트에 따른 w와 l값 >

Secure Hash Algorithm(SHA)-3

- $\theta(theta)$

1. $((x-1),z)$ column 비트들의 합과 $((x+1),(z-1))$ column 비트들의 합을 XOR

2. 1의 결과를 (x,y,z)비트에 XOR 함
   (최종 결과 값 : (x,y,z)에 저장)



Secure Hash Algorithm(SHA)-3 Quantum circuit

- $\theta(theta)$ Quantum circuit

  Classic : C, D, temp, A_out (state(1600) 크기의 temp 4개 사용)

```
for i in range(5):
        for j in range(5):
                for k in range(64):
                        C=sum([A[(i-1)%5][ji][k] for ji in range(5)]) % 2
                        D=sum([A[((i+1) % 5)][ji][(k-1)%64] for ji in range(5)]) % 2
                        temp=C+D+A[i][j][k] % 2
                        A_out[i][j][k]=temp
```

  Quantum : anc (state(1600) 크기의 anc 1개 사용) → 4800 큐비트 절약

  But 매 라운드마다 1600 큐비트 사용

  1. CNOT 게이트를 사용하여 input[x-1][-][z] 에 있는 column들의 합을 모두 anc에 저장
  2. CNOT 게이트를 사용하여 input[x+1][-][z+1] 에 있는 column들의 합을 모두 anc에 저장 (1과 2 중첩)
  3. CNOT 게이트를 사용하여 anc와 기존 input[x][y][z] 에 대한 mod 2 덧셈 진행 → 저장은 anc

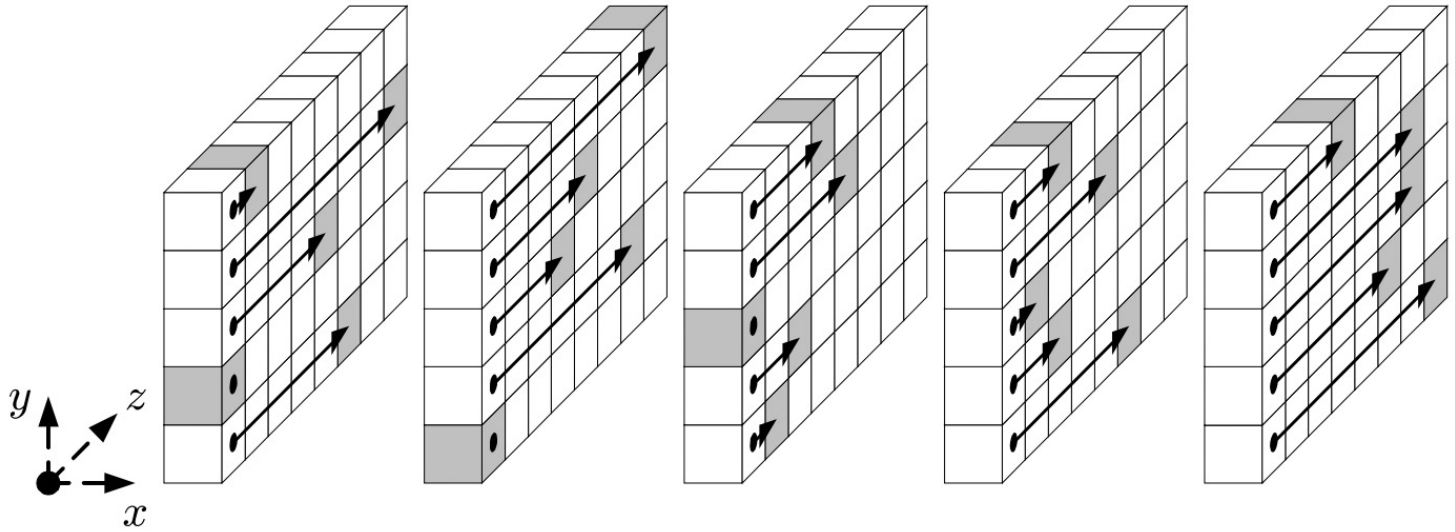
  ★ **1-2-3 반복에서 input[x][y][z]는 이후 1,2 에서 사용되므로 값이 유지되어야 함**

Secure Hash Algorithm(SHA)-3

- $\rho(rho)$
  각 lane에서 정해진 offset 만큼 Rotate하는 과정

	$x=3$	$x=4$	$x=0$	$x=1$	$x=2$
$y=2$	153	231	3	10	171
$y=1$	55	276	36	300	6
$y=0$	28	91	0	1	190
$y=4$	120	78	210	66	253
$y=3$	21	136	105	45	15

Table 2: Offsets of ρ [8]

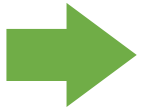


Secure Hash Algorithm(SHA)-3 Quantum circuit

- $\rho(rho)$ Quantum circuit

	$x=3$	$x=4$	$x=0$	$x=1$	$x=2$
$y=2$	153	231	3	10	171
$y=1$	55	276	36	300	6
$y=0$	28	91	0	1	190
$y=4$	120	78	210	66	253
$y=3$	21	136	105	45	15

Table 2: Offsets of ρ [8]



	$x = 3$	$x = 4$	$x = 0$	$x = 1$	$x = 2$
y = 2	25	39	3	10	43
y = 1	55	20	36	44	6
y = 0	28	27	0	1	62
y = 4	56	14	18	2	61
y = 3	21	8	41	45	15

```
for i in range(5):
    for j in range(5):
        for k in range(64):
            result[i][j][k] = x[i][j][k - rhomatrix[i][j]]
```

Secure Hash Algorithm(SHA)-3

- $\pi(pi)$ Quantum circuit

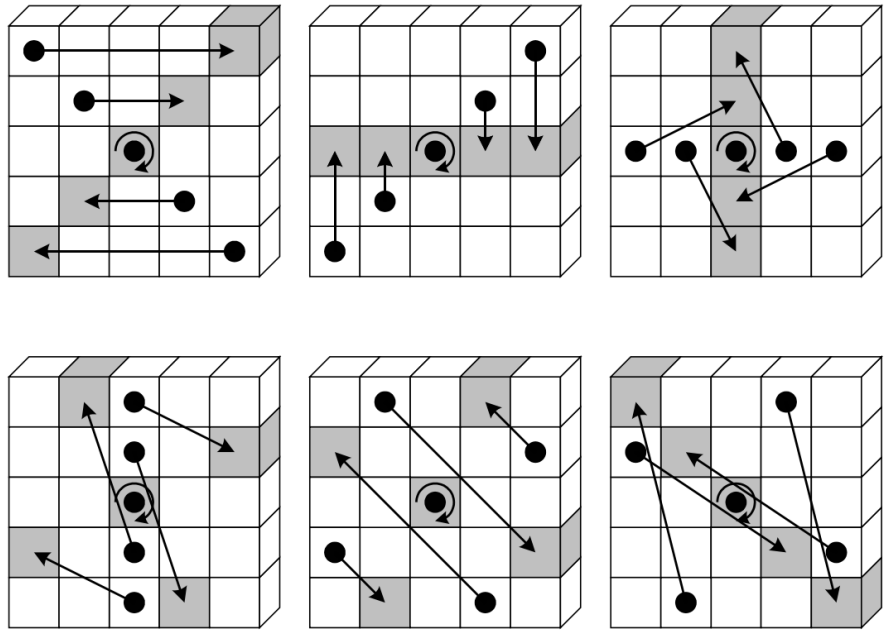
State 내에서 lane의 자리를 재배치하는 과정

Steps:

1. For all triples (x, y, z) such that $0 \le x < 5$, $0 \le y < 5$, and $0 \le z < w$, let
$$\mathbf{A}'[x, y, z] = \mathbf{A}[(x + 3y) \bmod 5, x, z].$$
2. Return $\mathbf{A}'$.

- SWAP 게이트로 단순 Bit Rotation 수행

```
for i in range(5):
    for j in range(5):
        for k in range(64):
            result[i][j][k] = x[pi_matrix[i][j]][i][k]
```



Secure Hash Algorithm(SHA)-3

- $\chi(chi)$ Quantum circuit

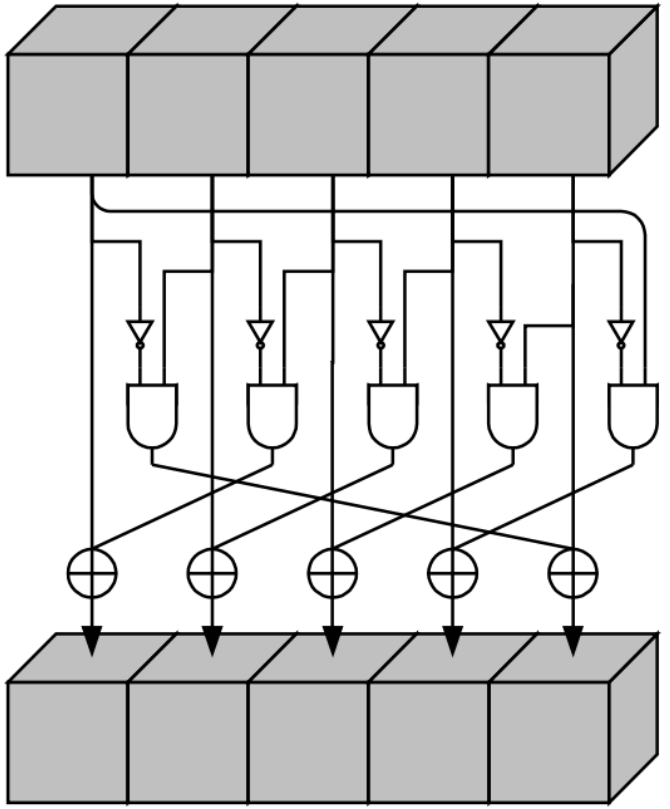
  오른쪽 2개 비트를 곱셈 연산한 결과와 XOR 연산하는 과정

Steps:

1. For all triples (x, y, z) such that $0 \le x < 5$, $0 \le y < 5$, and $0 \le z < w$, let
   $$\mathbf{A}'[x,y,z] = \mathbf{A}[x,y,z] \oplus ((\mathbf{A}[(x+1) \bmod 5, y, z] \oplus 1) \cdot \mathbf{A}[(x+2) \bmod 5, y, z]).$$
2. Return $\mathbf{A}'$.

중간 Temp 결과를 저장하지 않고 input에 바로 저장되도록 함

반복 x=0 : input[1][y][z]와 input[2][y][z] 을 사용하여 input[0][y][z] Update
반복 x=1 : input[2][y][z]와 input[3][y][z] 을 사용하여 input[1][y][z] Update
반복 x=2 : input[3][y][z]와 input[4][y][z] 을 사용하여 input[2][y][z] Update
반복 x=3 : input[4][y][z]와 input[0][y][z] 을 사용하여 input[3][y][z] Update
반복 x=4 : input[0][y][z]와 input[1][y][z] 을 사용하여 input[4][y][z] Update



Input[0][y][z], input[1][y][z]를 각각 저장하기 위한 320+320 = 640 큐비트만 사용

Secure Hash Algorithm(SHA)-3

- $\iota(iota)$ Quantum circuit

  Lane(0,0)이 라운드 상수와 XOR 연산을 수행하는 과정

Steps:

1. For all triples (x, y, z) such that $0 \le x < 5$, $0 \le y < 5$, and $0 \le z < w$, let $\mathbf{A}'[x, y, z] = \mathbf{A}[x, y, z]$.
2. Let $RC = 0^w$.
3. For j from 0 to ℓ, let $RC[2^j - 1] = rc(j + 7i_r)$.
4. For all z such that $0 \le z < w$, let $\mathbf{A}'[0, 0, z] = \mathbf{A}'[0, 0, z] \oplus RC[z]$.
5. Return $\mathbf{A}'$.

- RC는 상수이므로 Classic 계산으로 연산
- RC와 input의 CNOT 연산 → RC는 Classic 값이며 input는 Quantum 값이므로 RC 값에 조건문을 걸어 input에 X 게이트 수행

```
for l in range(7):
    if rc[l + 7 * r] % 2 == 1:
        X | x[0][0][2 ** l - 1]
```

Secure Hash Algorithm(SHA)-3

- 양자자원 추정 결과

이전 논문 연구 결과

	X	$P/P^\dagger$	$T/T^\dagger$	H	CNOT	T-depth	Depth
θ	0	0	0	0	17600	0	275
θ^{-1}	0	0	0	0	1360000	0	25
χ	0	0	11200	3200	14400	15	55
χ^{-1}	0	0	13440	3840	18880	18	66
ι	85	0	0	0	0	0	24
SHA3-256	85	0	591360	168960	33269760	792	10128
SHA3-256 (Opt.)	85	46080	499200	168960	34260480	432	11040

Qubit : 3,200

제안 양자회로 결과

Qubit	T	CNOT	X	Depth
55,360	115,200	591,360	76,886	2,020

Q & A