Quantum Implementation of SHA3

https://youtu.be/vXGus6a-h8E

장경배

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

NIST Post-quantum Security

- NIST는 AES, SHA3에 대한 양자 키 복구 공격 (i.e., Grover' search) 의 복잡도에 따라 양자 후 보안 강도를 정의하고 있음
  - AES: Grassl et al.의 2016년도 연구를 인용 → Jaques et al.의 2020년도 연구를 인용
  - **SHA3: 아직 없음**

AES-128	2^{157}/MAXDEPTH quantum gates or 2^{143} classical gates
SHA3-256	2^{146} classical gates
AES-192	2^{221}/MAXDEPTH quantum gates or 2^{207} classical gates
SHA3-384	2^{210} classical gates
AES-256	2^{285}/MAXDEPTH quantum gates or 2^{272} classical gates
SHA3-512	2^{274} classical gates

SHA3 Quantum Implementation

- **SHA3 (Keccak) 해시 함수 양자 회로 구현, Depth 최적화**
  - **PQC의 Encapsulation, Decapsulation에서도 사용됨**

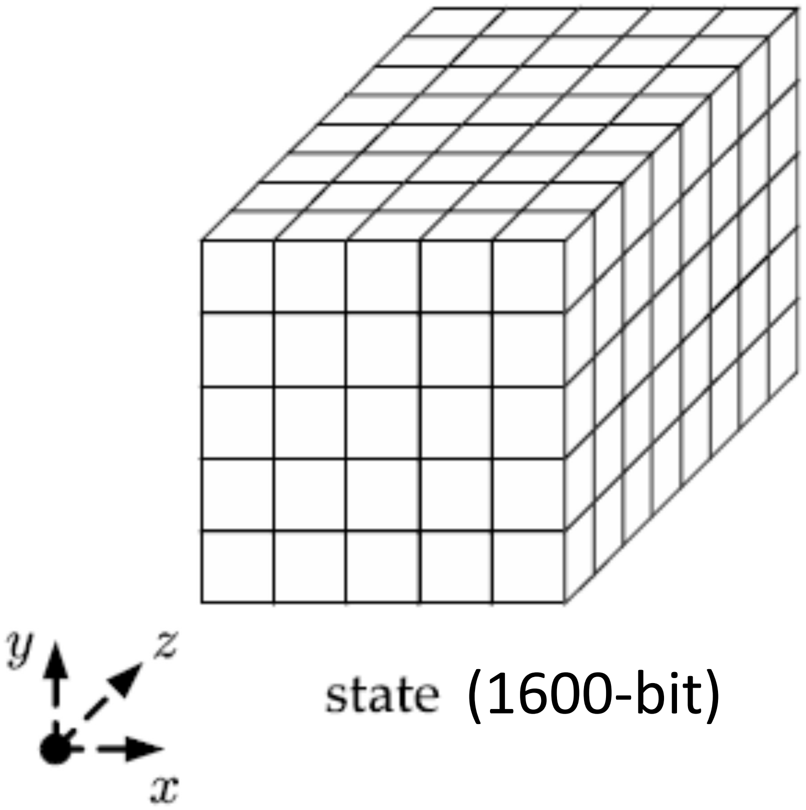
SHA3 (Keccak)	Qubits	Clifford	T	Toffoli depth	Full depth
[1]	3,200	33,438,805	591,360	264	10,128
[2]	55,360	975,448	268,800	184	2,860
[Ours]	**49,280**	**614,486**	**268,800**	**24**	**794**

AES	Qubits	Clifford	T	Toffoli depth	Full depth
AES-128	7,520	181,088	85,680	40	799

[1] M. Amy, O. D. Matteo, V. Gheorghiu, M. Mosca, A. Parent, J. Schanck, "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3", *SAC 2016*, 2016.
[2] G. Song, K. Jang, H. Seo, "Improved Low-Depth SHA3 Quantum Circuit for Fault-Tolerant Quantum Computers", *Applied Sciences,* 2023.

SHA3

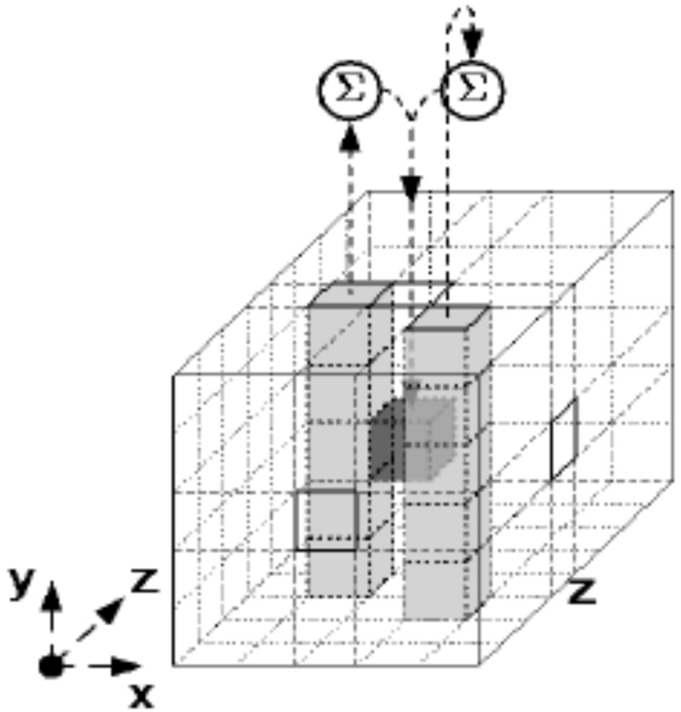
- **Round function**
  - **Theta (θ)**
  - Rho (ρ)
  - Pi (π)
  - **Chi (χ)**
  - **Iota (ι)**

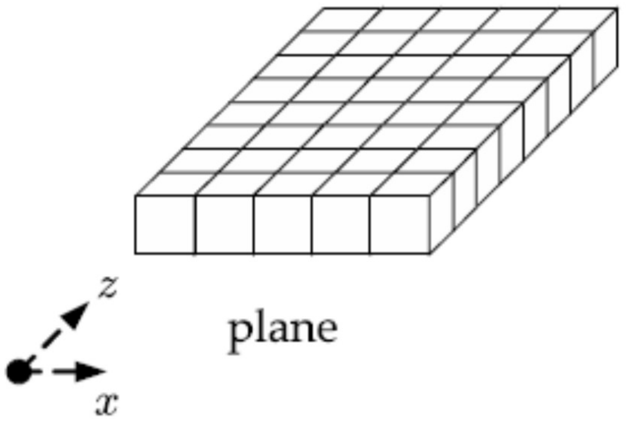


(1600-bit)

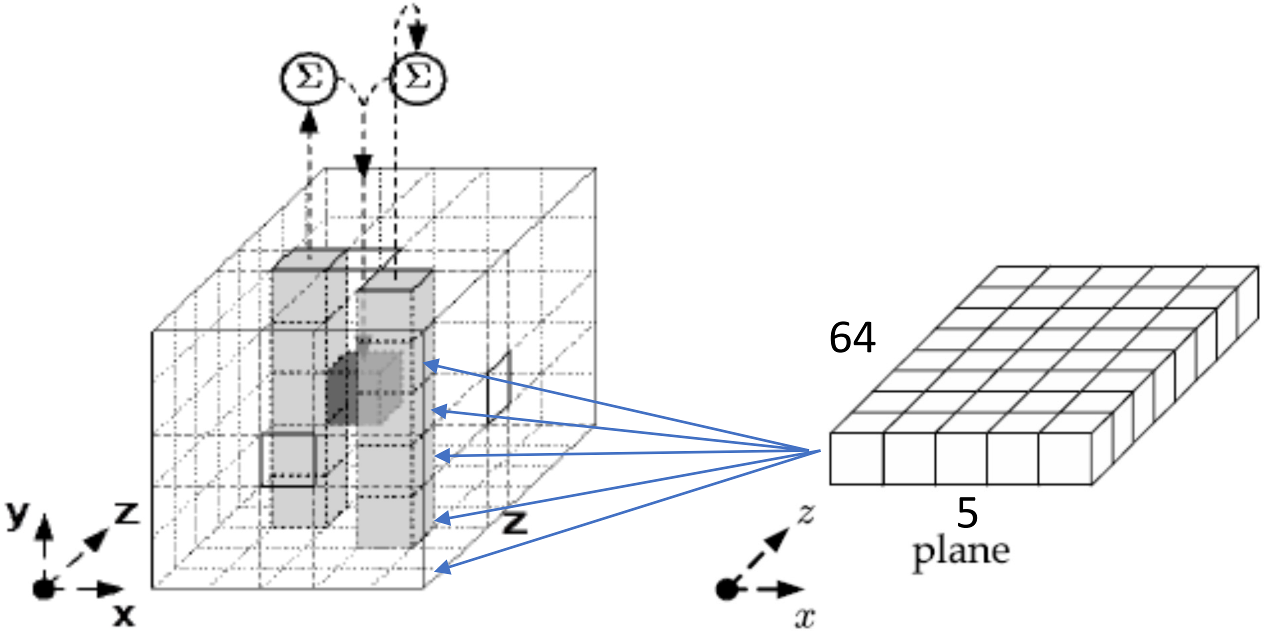
Theta (θ)

- **y축의 두 column들의 비트 합을 모든 x, y, z에 XOR**
  - $S[x][y][z] = S[x][y][z]) \oplus \left(\bigoplus_{i=0}^{4} S[x-1][i][z]) \oplus S[x+1][i][z-1] \right)$





Theta (θ)



*** [1] θ : 17600 (11 x 1600)**

$S[x][y][z]) \oplus \rightarrow +\,1$ CNOT gate

$\left(\bigoplus_{i=0}^{4} S[x-1][i][z]) \oplus S[x+1][i][z-1]\right) \rightarrow +\,10$ CNOT gates

$[x][y][z]$ range $\rightarrow \times\, 5 \times 5 \times 64 = 1600.$

(θ^{-1}: 136000)

Table 1: Quantum resources required for implementations of θ.

Source	#CNOT	#Qubit	Depth
[1]*	1377600	3200*	300
[2]	24000	3200	79
Ours	4800	1920	23

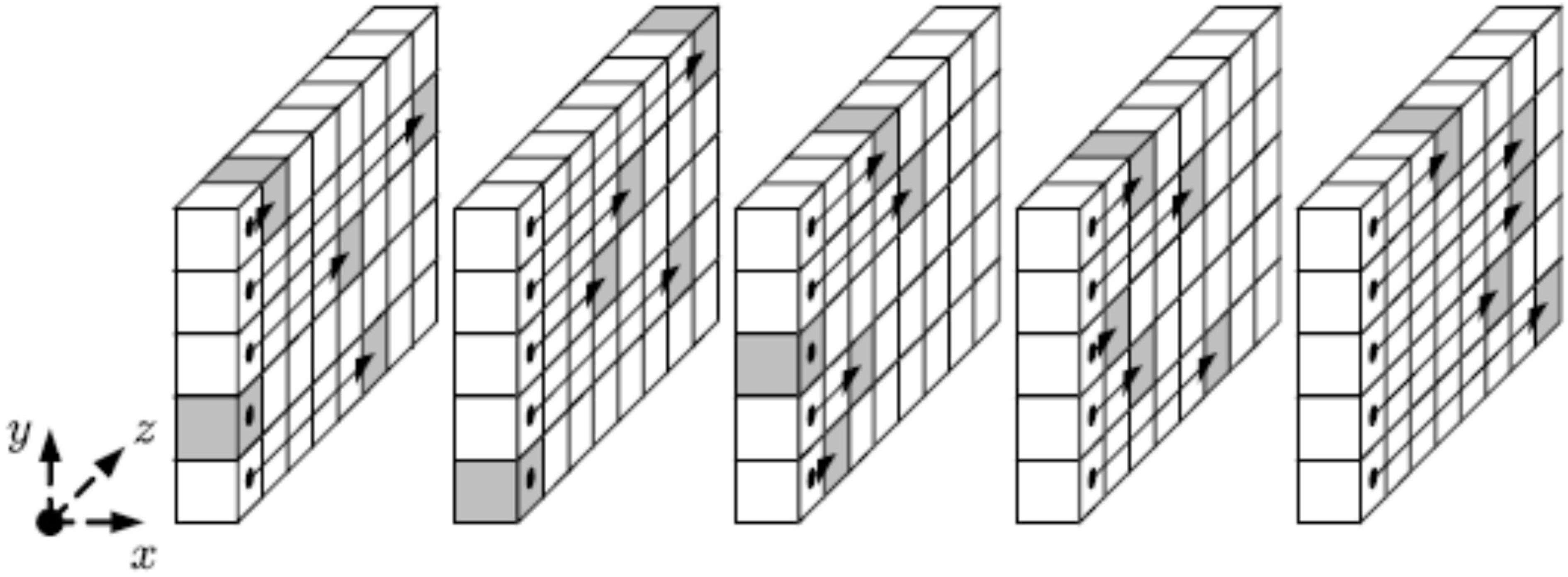
✲: include both θ and θ^{-1}.

✳: 1600 qubits are reused.

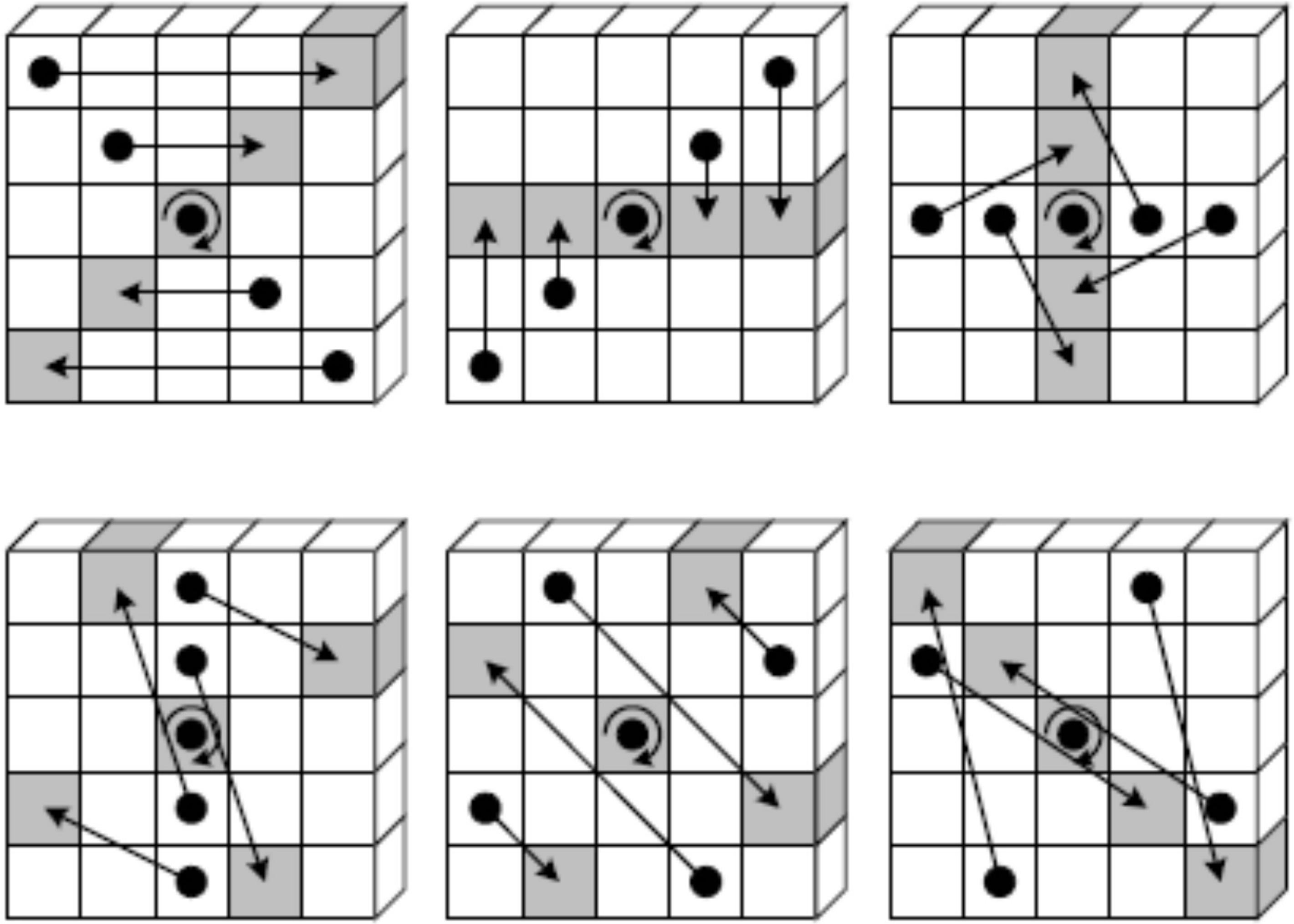
Ours θ: **4800 (1600 + 3200)**

plane setting: 320 x 5 = 1600
XOR: 2 X 1600 = 3200

Rho (ρ)

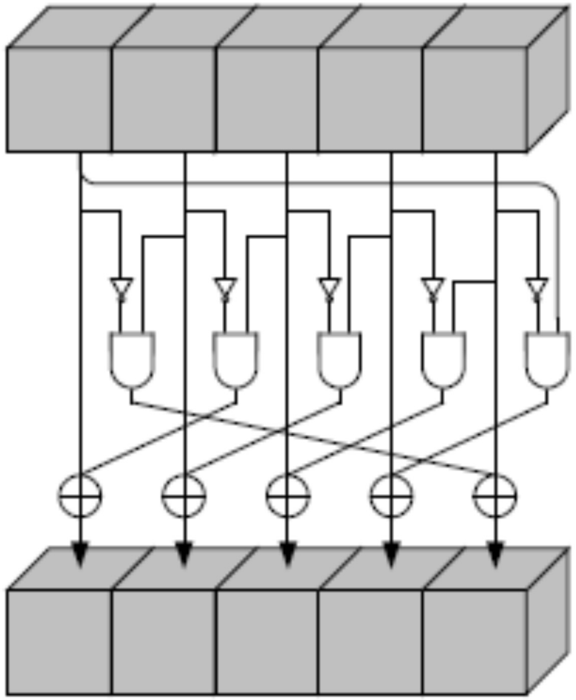


Pi (π)



Chi (χ)

- **유일하게 AND (Toffoli) 연산이 사용되는 부분**



Order	Required Qubit		Update Target
$x = 0$	$X[1][y][z]$	$X[2][y][z]$	$X[0][y][z]$
$x = 1$	$X[2][y][z]$	$X[3][y][z]$	$X[1][y][z]$
$x = 2$	$X[3][y][z]$	$X[4][y][z]$	$X[2][y][z]$
$x = 3$	$X[4][y][z]$	$\circledast\ X[0][y][z]$	$X[3][y][z]$
$x = 4$	$X[0][y][z]$	$\circledast\ X[1][y][z]$	$X[4][y][z]$

Chi (χ)

- WISA'22 카라추바 곱셈 기법과 유사하게 구현
  - 모든 값들을 사전에 준비하고, 한 번에 병렬 연산 → Toffoli depth: 1

Order	Required Qubit		Update Target
$x=0$	$X[1][y][z]$	$X[2][y][z]$	$X[0][y][z]$
$x=1$	$X[2][y][z]$	$X[3][y][z]$	$X[1][y][z]$
$x=2$	$X[3][y][z]$	$X[4][y][z]$	$X[2][y][z]$
$x=3$	$X[4][y][z]$	$\circledast\ X[0][y][z]$	$X[3][y][z]$
$x=4$	$X[0][y][z]$	$\circledast\ X[1][y][z]$	$X[4][y][z]$

		Copy	Copy
Toffoli	**(Control 1**	**Control 2**	**Result)**
		Reverse	
	Garbage	0 (reuse)	

Chi (χ)

```python
rows = 25
cols = 64
result_state = [[0 for j in range(cols)] for i in range(rows)]
for i in range(rows):
    for j in range(cols):
        result_state[i][j] = eng.allocate_qubit()
```

큐비트 할당

```python
state_copy_x(eng, state, copy_state)
state_copy(eng, state, result_state)
```

Copy

```python
for i in range(5): # 32
    for j in range(5):
        for k in range(64):
            Toffoli_gate(eng, copy_state[i*5+((j+1)%5)][k], state[i*5+((j+2)%5)][k], result_state[j+i*5][k])
```

병렬 연산

```python
state_copy_x(eng, state, copy_state)
```

Reverse

```python
state = result_state
```

Iota (ι) & Performance

- **Round 상수 XOR 연산**
  - **Classical-Quantum 구현**으로 분류됨
  - X 게이트만이 사용되며, 간단히 구현 가능

- **결과**

SHA3 (Keccak)	Qubits	Clifford	T	Toffoli depth	Full depth
[1]	3,200	33,438,805	591,360	264	10,128
[2]	55,360	975,448	268,800	184	2,860
[Ours]	**49,280**	**614,486**	**268,800**	**24**	**794**

Thank you!