ASCON AEAD 및 HASH 양자회로 구현

https://www.youtube.com/watch?v=urn3CHPcsZo

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

ASCON

- **ASCON** – NIST 경량암호 표준화로 선정된 암호 제품군
  - NIST의 권장사항은 Ascon-128 또는 Ascon-128a와 결합된 Ascon-Hash.
  - 모든 체계는 128비트 보안을 제공하고 내부적으로 동일한 320비트 순열을 사용하므로 AEAD와 Hash모두 구현하기 좋음.

- ○ **AEAD**
  - ○ **ASCON -128**
  - ○ **ASCON - 128a**
- ○ **Hash**
  - ○ **ASCON - HASH**
  - ○ **ASCON - XOF**
- ○ **Variant**
  - ○ **ASCON - 80pq**

ASCON parameter

- **ASCON AEAD**

Name	Algorithms	Bit size				Rounds	
		Key	Nonce	Tag	block	p^a	p^b
ASCON-128	$\mathcal{E}, D_{128,64,12,6}$	128	128	128	64	12	6
ASCON-128a	$\mathcal{E}, D_{128,128,12,8}$	128	128	128	128	12	8

- **ASCON HASH**

Name	Algorithms	Bit size		Rounds	
		Hash	block	p^a	p^b
ASCON-HASH	$\chi_{256,64,12}$ with $\ell = 256$	256	64	12	0

ASCON - AEAD

- *Initialization, processing Associated Data, Processing Plaintext, Finalization*

  모든 프로세스마다 Permutation 포함.

- $IV_{k,r,a,b} \leftarrow k \parallel r \parallel a \parallel b \parallel 0^{160-k} = \begin{cases} 80400c0600000000 & \text{for ASCON-128} \\ 80800c0800000000 & \text{for ASCON-128a} \\ a0400c06 & \text{for ASCON-80pq} \end{cases}$

  (k = key size , r = data block , $(a\ ,\ b)$ = round number)

- $S \leftarrow IV_{k,r,a,b} \parallel K \parallel N$

  (K = key , N = Nonce)

Initialization
$S \leftarrow IV_{k,r,a,b} \parallel K \parallel N$
$S \leftarrow p^a(S) \oplus (0^{320-k} \parallel K)$
Processing Associated Data
if $|A| > 0$ **then**
$A_1 \dots A_s \leftarrow r$-bit blocks of $A \| 1 \| 0^*$
for $i = 1, \dots, s$ **do**
$S \leftarrow p^b((S_r \oplus A_i) \parallel S_c)$
$S \leftarrow S \oplus (0^{319} \parallel 1)$
Processing Plaintext
$P_1 \dots P_t \leftarrow r$-bit blocks of $P \| 1 \| 0^*$
for $i = 1, \dots, t-1$ **do**
$S_r \leftarrow S_r \oplus P_i$
$C_i \leftarrow S_r$
$S \leftarrow p^b(S)$
$S_r \leftarrow S_r \oplus P_t$
$\tilde{C}_t \leftarrow \lfloor S_r \rfloor_{|P| \bmod r}$
Finalization
$S \leftarrow p^a(S \oplus (0^r \parallel K \parallel 0^{320-r-k}))$
$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
return $C_1 \parallel \dots \parallel C_{t-1} \parallel \tilde{C}_t, T$

ASCON - HASH

- *Initialization, Absorbing, Squeezing*

  모든 프로세스마다 Permutation 포함.

- $IV_{k,r,a,b} \leftarrow 0^8 \parallel r \parallel a \parallel 0^8 \| h = \begin{cases} 00400c0000000000 & \text{for ASCON-XOF} \\ 00400c0000000100 & \text{for ASCON-HASH} \end{cases}$

  (r = data block , a = round number, h = output length limit)

- $S \leftarrow p^a(IV_{k,r,a,b} \parallel 0^{256})$

- $$S \leftarrow \begin{array}{l} ee9398aadb67f03d \;\| \\ 8bb21831c60f1002 \;\| \\ b48a92db98d5da62 \;\| \\ 43189921b8f8e3e8 \;\| \\ 348fa5c9d525e140 \;\| \end{array}$$

Initialization
$S \leftarrow p^a(\mathrm{IV}_{h,r,a} \| 0^c)$
Absorbing
$M_1 \ldots M_s \leftarrow M \| 1 \| 0^*$
for $i = 1, \ldots, s$ **do**
$S \leftarrow p^a((S_r \oplus M_i) \| S_c)$
Squeezing
for $i = 1, \ldots, t = \lceil \ell / r \rceil$ **do**
$H_i \leftarrow S_r$
$S \leftarrow p^a(S)$
return $\lfloor H_1 \| \ldots \| H_t \rfloor_\ell$

Permutation

• Permutation

ASCON의 주요 구성요소는 320bit 순열

$P = P_L \circ P_S \circ P_C$

320비트 S는 5개의 64비트 레지스터 워드 $x_i, S = x_0 \parallel x_1 \parallel x_2 \parallel x_3 \parallel x_4$로 분할 ($x_0 = MSB, x_4 = LSB$)

- **Addition of Constants**(P_c)

  x_2 에 constant C_r add. $x_2 \leftarrow x_2 \oplus C_r$

$r = i \;\; for \;\; p_a \;\; \& \;\; r = i + a - b \;\; for \;\; p_b$

p^{12}	p^8	p^6	Constant c_r	p^{12}	p^8	p^6	Constant c_r
0			00000000000000000f0	6	2	0	0000000000000000096
1			00000000000000000e1	7	3	1	0000000000000000087
2			00000000000000000d2	8	4	2	0000000000000000078
3			00000000000000000c3	9	5	3	0000000000000000069
4	0		00000000000000000b4	10	6	4	000000000000000005a
5	1		00000000000000000a5	11	7	5	000000000000000004b

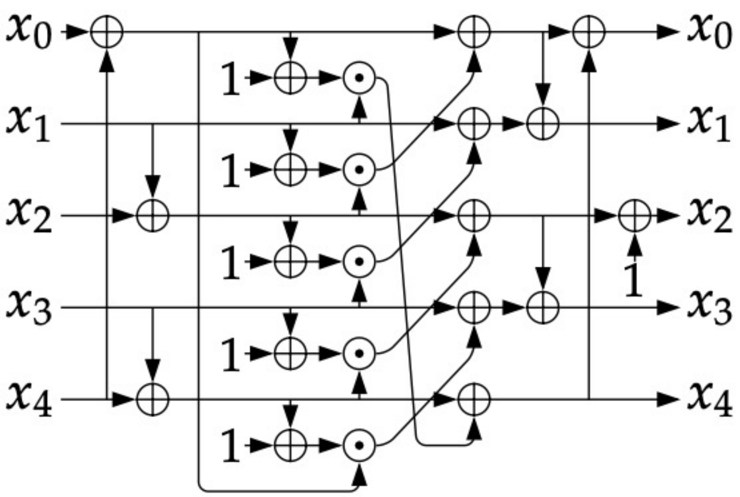
Permutation

- **Substitution Layer** (P_S)

5비트 S-box 를 64개의 병렬로 구성하여 상태 S를 5개 레지스터의 각 비트 슬라이스로 변환

Table 5: Ascon's 5-bit S-box $\mathcal{S}$ as a lookup table.

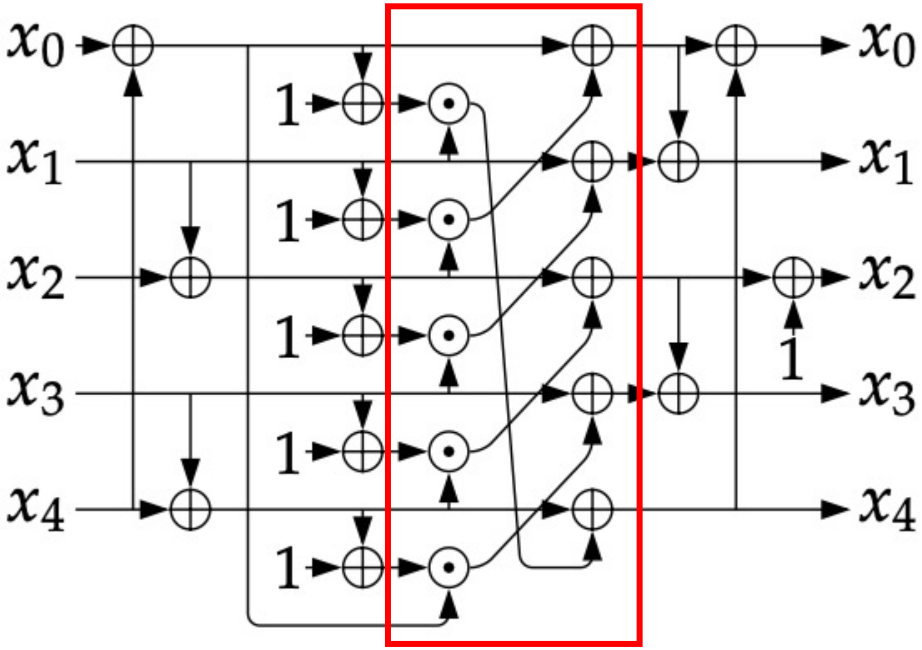
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$\mathcal{S}(x)$	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17

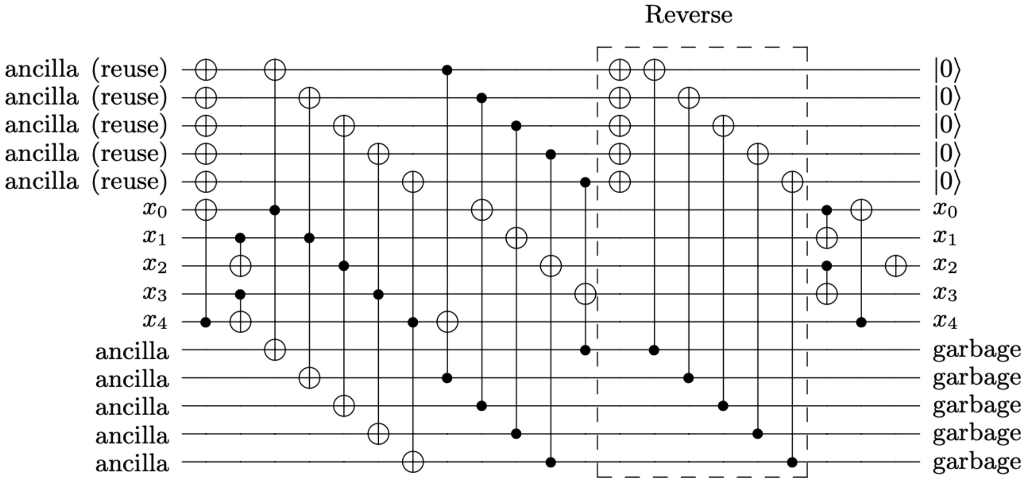


$x_0 = x_0 \oplus x_4,\ x_4 = x_4 \oplus x_3,\ x_2 = x_2 \oplus x_1,$
$t_0 = x_0,\ t_1 = x_1,\ t_2 = x_2,\ t_3 = x_3,\ t_4 = x_4,$
$t_0 = \sim t_0,\ t_1 = \sim t_1,\ t_2 = \sim t_2,\ t_3 = \sim t_3,\ t_4 = \sim t_4,$
$t_0 = t_0 \cdot x_1,\ t_1 = t_1 \cdot x_2,\ t_2 = t_2 \cdot x_3,\ t_3 = t_3 \cdot x_4,\ t_4 = t_4 \cdot x_0,$
$x_0 = x_0 \oplus t_1,\ x_1 = x_1 \oplus t_2,\ x_2 = x_2 \oplus t_3,\ x_3 = x_3 \oplus t_4,$
$x_4 = x_4 \oplus t_0,\ x_1 = x_1 \oplus x_0,\ x_0 = x_0 \oplus x_4,$
$x_3 = x_3 \oplus x_2,\ x_2 = \sim x$

Permutation

- **Substitution Layer** (P_S)






```
for i in range(64):
    Toffoli_gate(eng, ancilla_x1[i], new_ancilla_x2[i], x0[i])
for i in range(64):
    Toffoli_gate(eng, ancilla_x2[i], new_ancilla_x3[i], x1[i])
for i in range(64):
    Toffoli_gate(eng, ancilla_x3[i], new_ancilla_x4[i], x2[i])
for i in range(64):
    Toffoli_gate(eng, ancilla_x0[i], new_ancilla_x1[i], x4[i])
for i in range(64):
    Toffoli_gate(eng, ancilla_x4[i], new_ancilla_x0[i], x3[i])
```

Ancilla qubit를 더 사용하여 병렬로 연산
→ Toffoli depth 와 Full depth 부분에서 최적화

Permutation

- **Linear Layer** (P_L)

각 64비트 레지스터 워드 x_i 내에서 선형 연산 진행 (Naïve한 구현)

$$
\begin{aligned}
\boxed{x_0 \leftarrow x_0} &\oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28) \\
x_1 \leftarrow x_1 &\oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \\
x_2 \leftarrow x_2 &\oplus (x_2 \ggg 01) \oplus (x_2 \ggg 06) \\
x_3 \leftarrow x_3 &\oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17) \\
x_4 \leftarrow x_4 &\oplus (x_4 \ggg 07) \oplus (x_4 \ggg 41)
\end{aligned}
$$

```python
def LinearDiffusion_Layer(eng,x0,x1,x2,x3,x4):
    new_x0 = eng.allocate_qureg(64)
    new_x1 = eng.allocate_qureg(64)
    new_x2 = eng.allocate_qureg(64)
    new_x3 = eng.allocate_qureg(64)
    new_x4 = eng.allocate_qureg(64)
```

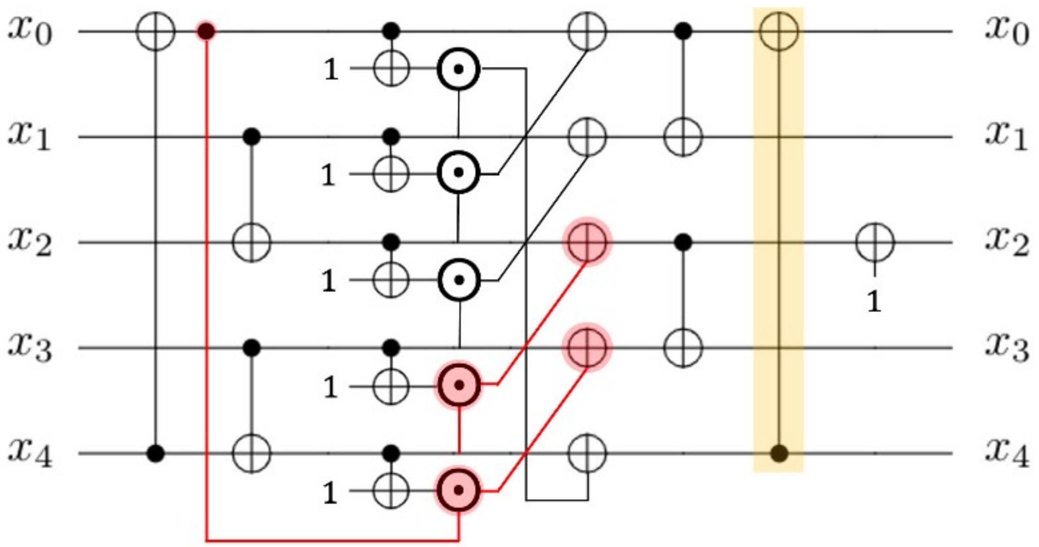
Table 1: Comparison of quantum resources required for ASCON linear layer.

Linear layer	Source	#CNOT	#Qubit	Depth
Out-of-place	This work	960	640	3
Naïve (binary matrix)	RBC'23 [18]	960	640	26
Gauss-Jordan	RBC'23 [18]	2,413	320	358
PLU	RBC'23 [18]	2,413	320	288
Modified [19]	RBC'23 [18]	1,595	320	119

18. S.Roy, A.Baksi, and A.Chattopadhyay, "Quantum implementation of ascon linear layer," Cryptology ePrint Archive, 2023. 9

Permutation

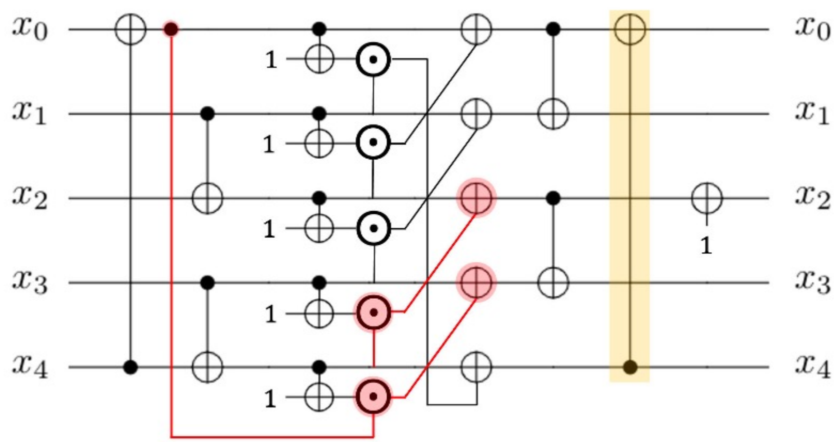
- Efficient Implementation of Lightweight Hash Functions on GPU and Quantum Computers for IoT Applications (ASCON-HASH)
  - **qubit 최적화를 위해 Linear Layer에서 사용하는 ancilla qubit를 이용한 S-box**
  - **Substitution Layer와 Linear Layer ancilla qubit(temp qubit) 공유**

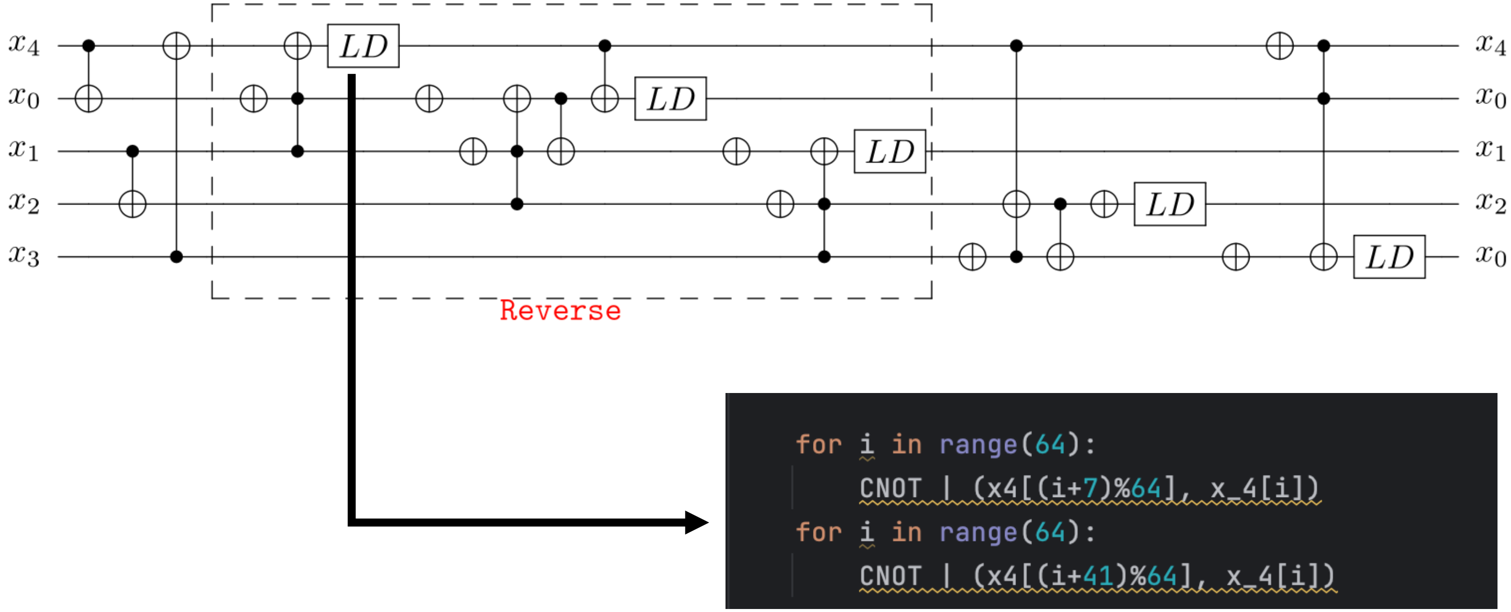


$$
\begin{aligned}
x_0 &\leftarrow x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28)\\
x_1 &\leftarrow x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39)\\
x_2 &\leftarrow x_2 \oplus (x_2 \ggg 01) \oplus (x_2 \ggg 06)\\
x_3 &\leftarrow x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17)\\
x_4 &\leftarrow x_4 \oplus (x_4 \ggg 07) \oplus (x_4 \ggg 41)
\end{aligned}
$$

Permutation

- Efficient Implementation of Lightweight Hash Functions on GPU and Quantum Computers for IoT Applications (ASCON-HASH)
  - **qubit 최적화를 위해 Linear Layer에서 사용하는 ancilla qubit를 이용한 S-box**
  - **Substitution Layer와 Linear Layer ancilla qubit(temp qubit) 공유**





ASCON-AEAD

- ASCON-128 양자 회로 구현

Initialization

$S \leftarrow \mathrm{IV}_{k,r,a,b} \parallel K \parallel N$

$S \leftarrow p^a(S) \oplus (0^{320-k} \parallel K)$

Processing Associated Data

if $|A| > 0$ **then**

$A_1 \ldots A_s \leftarrow r$-bit blocks of $A \| 1 \| 0^*$

for $i = 1, \ldots, s$ **do**

$S \leftarrow p^b((S_r \oplus A_i) \parallel S_c)$

$S \leftarrow S \oplus (0^{319} \parallel 1)$

Processing Plaintext

$P_1 \ldots P_t \leftarrow r$-bit blocks of $P \| 1 \| 0^*$

for $i = 1, \ldots, t-1$ **do**

$S_r \leftarrow S_r \oplus P_i$

$C_i \leftarrow S_r$

$S \leftarrow p^b(S)$

$S_r \leftarrow S_r \oplus P_t$

$\tilde{C}_t \leftarrow \lfloor S_r \rfloor_{|P| \bmod r}$

Finalization

$S \leftarrow p^a(S \oplus (0^r \parallel K \parallel 0^{320-r-k}))$

$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$

return $C_1 \parallel \ldots \parallel C_{t-1} \parallel \tilde{C}_t, T$

$$S = x_0 \parallel x_1 \parallel x_2 \parallel x_3 \parallel x_4 \ (x_0 = MSB , x_4 = LSB)$$

```
def Associated(eng,pb,A,A_len,x0,x1,x2,x3,x4, new_ancilla_x0, new_ancilla_x1, new_ancilla_x2, new_ancilla_x3, new_ancilla_x4):
    for i in range(32):
        CNOT | (A[i],x0[32+i])

    X | x0[31]

    Permutation_b(eng,pb,x0,x1,x2,x3,x4, new_ancilla_x0, new_ancilla_x1, new_ancilla_x2, new_ancilla_x3, new_ancilla_x4)

    X | x4[0]

1 usage
def Plain(eng,pb,pt,pt_len,ct,x0,x1,x2,x3,x4):
    for i in range(32):
        CNOT | (pt[i],x0[32+i])
        CNOT | (x0[32+i], ct[i])

    X | x0[31]

1 usage
def Finalization(eng,pa,x0,x1,x2,x3,x4,key, new_ancilla_x0, new_ancilla_x1, new_ancilla_x2, new_ancilla_x3, new_ancilla_x4):
    for i in range(64):
        CNOT | (key[i + 64], x1[i])
        CNOT | (key[i], x2[i])

    Permutation_a(eng, pa, x0, x1, x2, x3, x4, new_ancilla_x0, new_ancilla_x1, new_ancilla_x2, new_ancilla_x3, new_ancilla_x4)

    for i in range(64):
        CNOT | (key[i + 64], x3[i])
        CNOT | (key[i], x4[i])
```

a hash function of ASCON

- ASCON-HASH 양자 회로 구현

Initialization

$S \leftarrow p^a(\mathrm{IV}_{h,r,a} \parallel 0^c)$

Absorbing

$M_1 \dots M_s \leftarrow M \parallel 1 \parallel 0^*$

for $i = 1, \dots, s$ **do**

$S \leftarrow p^a((S_r \oplus M_i) \parallel S_c)$

Squeezing

for $i = 1, \dots, t = \lceil \ell / r \rceil$ **do**

$H_i \leftarrow S_r$

$S \leftarrow p^a(S)$

return $\lfloor H_1 \parallel \dots \parallel H_t \rfloor_\ell$

$$S = x_0 \parallel x_1 \parallel x_2 \parallel x_3 \parallel x_4 \;(x_0 = MSB, x_4 = LSB)$$

```
    # Absorbing

    for number in range(l):
        if(number != l-1):   # 0,1,2,3
            for i in range(64):
                CNOT | (M[len - (64*(number+1)) + i], x0[i])

            Permutation_a(eng,pa,x0,x1,x2,x3,x4, new_ancilla_x0, new_ancilla_x1, new_ancilla_x2, new_ancilla_x3, new_ancilla_x4)

        else :   #4
            left_len = len-64*number    #0
            start = 64-left_len         #64
            for i in range(left_len):
                CNOT | (M[i], x0[i+start])

            X | x0[63-left_len]

    Permutation_a(eng, pa, x0, x1, x2, x3, x4, new_ancilla_x0, new_ancilla_x1, new_ancilla_x2, new_ancilla_x3, new_ancilla_x4)

    #Squeezing

    for i in range(4):
        for j in range(64):
            CNOT | (x0[j],Hash[64*i+j])

        Permutation_a(eng, pa, x0, x1, x2, x3, x4, new_ancilla_x0, new_ancilla_x1, new_ancilla_x2, new_ancilla_x3, new_ancilla_x4)

    if(resource_check!=1):
        print_state(eng,Hash,64)
```

양자 자원 추정

- ASCON-128(AEAD)에 대한 양자 자원 추정

Table 2: Required quantum resources for ASCON-128 quantum circuit implementation (ours).

Cipher	#X	#CNOT	#Toffoli	Toffoli depth	#Qubit	Depth	TD-M cost
ASCON-128	21,243	69,600	9,600	30	20,064	304	601,920

※: Associated data and plaintext are both of 32-bits.

Table 3: Required decomposed quantum resources for ASCON-128 quantum circuit implementation (ours).

Cipher	#Clifford	$\#T$	T-depth	#Qubit	Full depth	FD-M cost
ASCON-128	167,643	67,200	120	20,064	513	10,292,832

※: Associated data and plaintext are both of 32-bits.

양자 자원 추정

- ASCON-HASH에 대한 양자 자원 추정

Table 4: Required quantum resources for ASCON-HASH quantum circuit implementation.

Cipher	Source	#X	#CNOT	#Toffoli	Toffoli depth	#Qubit	Depth	TD-M cost
ASCON-HASH	[20]	97,426	159,488	55,296	864	35,392	2,487	30,578,688
	Ours	76,691	249,344	34,560	108	70,272	1,090	7,589,376

※: Input message length is 256-bit.

Table 5: Required decomposed quantum resources for ASCON-HASH quantum circuit implementation.

Cipher	Source	#Clifford	$\#T$	T-depth	#Qubit	Full depth	FD-M cost
ASCON-HASH	[20]	699,282	387,072	3456	35,392	8,427	298,248,384
	Ours	602,515	241,920	432	70,272	1,845	129,651,840

※: Input message length is 256-bit.

결론

- Grover 공격 비용 추정 및 결론

Table 6: Cost of the Grover's key search for ASCON-128 (ours).

Cipher	Total gates	Total depth	Cost (complexity)	#Qubit	TD-M cost	FD-M cost
ASCON-128	$1.180 \cdot 2^{83}$	$1.574 \cdot 2^{73}$	$1.857 \cdot 2^{156}$	20,065	$1.799 \cdot 2^{83}$	$1.925 \cdot 2^{87}$

※: Associated data and plaintext are both of 32-bit.

Table 7: Costs of the Grover's collision search for ASCON-HASH.

Cipher	Source	Total gates	Total depth	Cost (complexity)	#Qubit	TD-M cost	FD-M cost
ASCON-HASH	[20]	$1.63 \cdot 2^{148}$	$1.616 \cdot 2^{141}$	$1.317 \cdot 2^{290}$	35,137	$1.431 \cdot 2^{153}$	$1.745 \cdot 2^{156}$
	Ours	$1.268 \cdot 2^{148}$	$1.415 \cdot 2^{139}$	$1.794 \cdot 2^{287}$	70,273	$1.416 \cdot 2^{151}$	$1.517 \cdot 2^{155}$

※: Input message length is 256-bit.

- **Level 1**: To be considered secure, any attack that compromises the relevant security definition must require computational resources that are at least comparable to those required for a key search on a 128-bit key block cipher, such as AES-128 $(2^{170} \rightarrow 2^{157})$.
- **Level 2**: To be considered secure, any attack that compromises the relevant security definition must require computational resources that are at least comparable to those required for a collision search on a 256-bit hash function, such as SHA-256/SHA3-256.

- ASCON-128
  - NIST post-quantum security Level 1 달성
- ASCON-HASH
  - 양자 컴퓨터 상에서의 보안레벨 제공 X,
  - 클래식 컴퓨터상에서의 보안 레벨만 제공
  - SHA3-256 양자 회로를 기반으로 $1.574 \cdot 2^{295}$ 로 추정
  - 현재 구현되어 있는 양자회로와 비교하면 보안레벨을 달성하지는 못했지만 AES처럼 예상비용을 잠재적으로 줄일 수 있기 때문에 변할 수 있음.

Q & A