Depth Optimized Implementation of ASCON Quantum Circuit

Yujin Oh - Hansung University

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

Contribution

1. Quantum Circuit Implementation of ASCON AEAD.

→ We present the first implementation of a quantum circuit for ASCON AEAD.

2. Low-Depth Implementation of ASCON AEAD.

→ In our quantum circuit implementation of ASCON, we prioritize achieving a low Toffoli depth and full de pth. We demonstrate the reduction of Toffoli depth and full depth through parallelization. Additionally, to m aintain a reasonable qubit count, we utilize the method of reusing ancilla qubits.

3. Post-quantum Security Assessment of ASCON AEAD.

→ We assess the quantum security of ASCON by estimating the cost of Grover's key search based on ou r implemented quantum circuit for ASCON. This evaluation involves comparing the estimated cost of Grov er's key search for ASCON with the security levels provided by NIST.

Background : NIST criteria

- **The NIST criteria for quantum attack complexity.**
  - **Level -1**

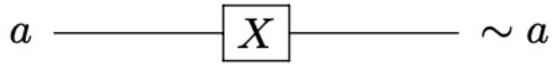
    → attacks on the security definition must require resources comparable to AES-128 key search. $(2^{170} \rightarrow 2^{157})$
  - **Level -3**

    → attacks on the security definition must require resources comparable to AES-192 key search. $(2^{233} \rightarrow 2^{221})$
  - **Level -5**

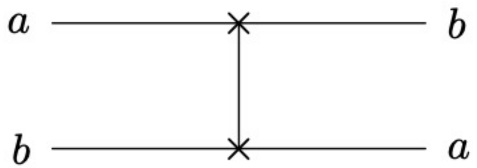
    → attacks on the security definition must require resources comparable to AES-256 key search. $(2^{298} \rightarrow 2^{285})$

Background : Quantum gates

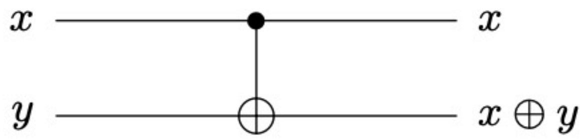
- Reversible quantum circuits for ciphers can be implemented using a variety of representa tive quantum gates.



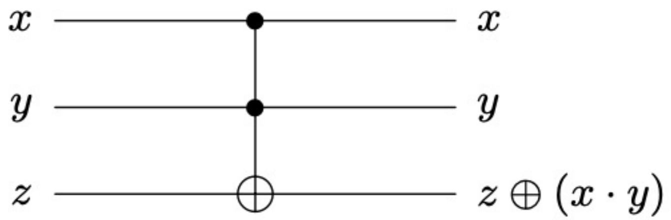
(a) X gate



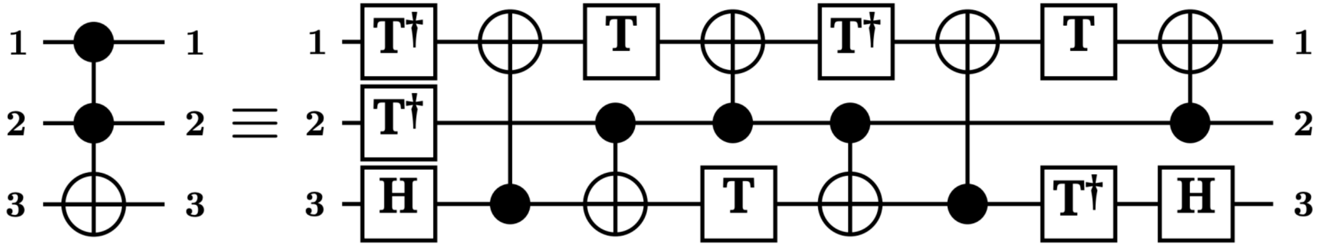
(b) Swap gate



(c) CNOT gate



(d) Toffoli gate



Toffoli gate decomposition (T- depth 4, total depth 8)

Background : Grover's key search

- Key search using Grover's Algorithm

1. Using Hadamard gates, n-qubit key has the same amplitude at all state of the qubits.

$$H^{\otimes k}|0\rangle^{\otimes k} = |\psi\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{1}{2^{k/2}}\sum_{x=0}^{2^k-1}|x\rangle$$

2. The encryption of plaintext using the superposition state key in an oracle generates a ciphertext in a super position state. If the ciphertext matches the expected ciphertext, the sign of the corresponding key value is in verted to retrieve the key.

$$f(x) = \begin{cases} 1 \text{ if } Enc_{key}(p) = c \\ 0 \text{ if } Enc_{key}(p) \neq c \end{cases}$$

3. The diffusion operator increases the probability of key observation.

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle|-\rangle$$

Background : ASCON

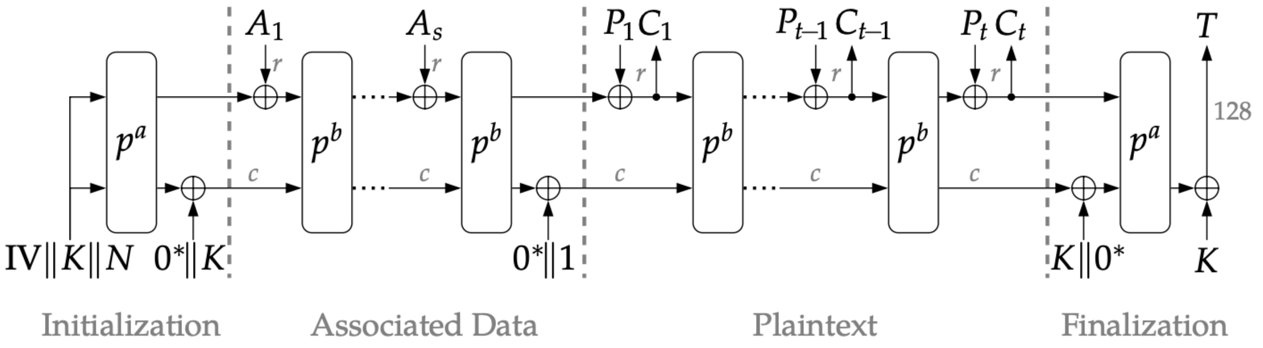
- **ASCON** is a symmetric key cipher that has been standardized in the NIST Lightweight Cryptography standardization.

- ASCON has two variants
  - ASCON-AEAD
  - a hash function

- ASCON – AEAD has two versions
  - ASCON-128
  - ASCON-128a

- ASCON-AEAD encryption process

→ $Initialization, Associated\ data, Plaintext, Finalization$



Background : ASCON

- The main components of all schemes in ASCON are two 320-bit permutations (p^a & p^b, different numbers of rounds)

- The permutations functions include add constants, a substitution layer with a 5-bit S-box, and a Linear layer with 64-bit diffusion functions.($p = p_L \circ p_S \circ p_c$)

- The 320-bit state *S* is split into five 64-bit registers words x_i

$$S = x_0 \parallel x_1 \parallel x_2 \parallel x_3 \parallel x_4$$

Proposed Method

- Our primary focus lies on **ASCON-128**, which corresponds to the AEAD variant.
  - Associated data and plaintext are both of **32-bits**.

- In quantum circuit for **ASCON**, the most resources are generally required for **Permutation** implementation.

- To achieve an efficient quantum circuit for ASCON, optimizing resources for implementing the **Substitution Layer (S-box) and Linear Layer** in permutation is crucial.

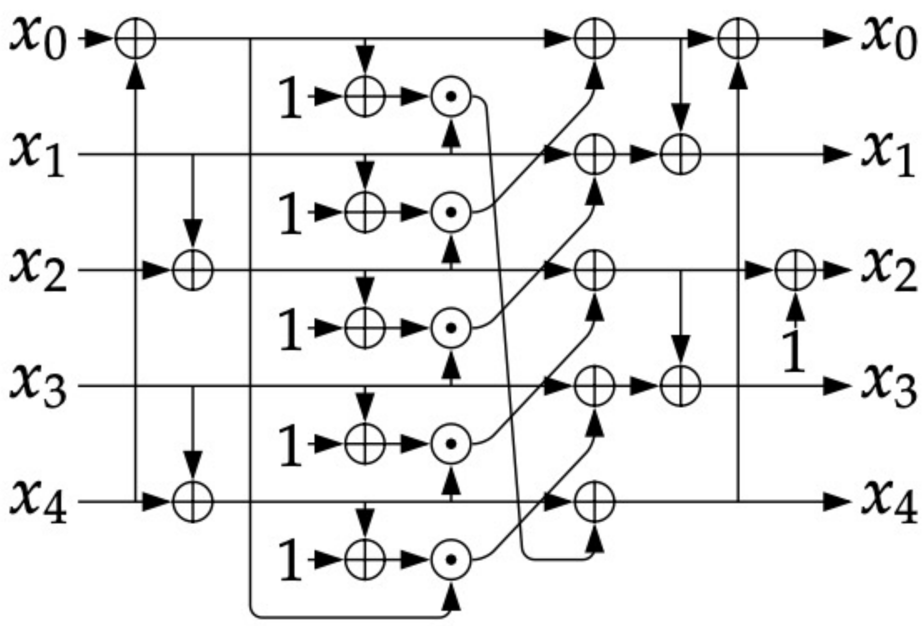
- We focus on **optimizing depth** of the ASCON-128 quantum circuit for optimal performance in Grover's algorithm.

Proposed Method

- **Implementation (with Parallelization) of S-box.**

Because of the reversible nature of quantum computing, the implementation of S-boxes using **look-up table is not suitable.**

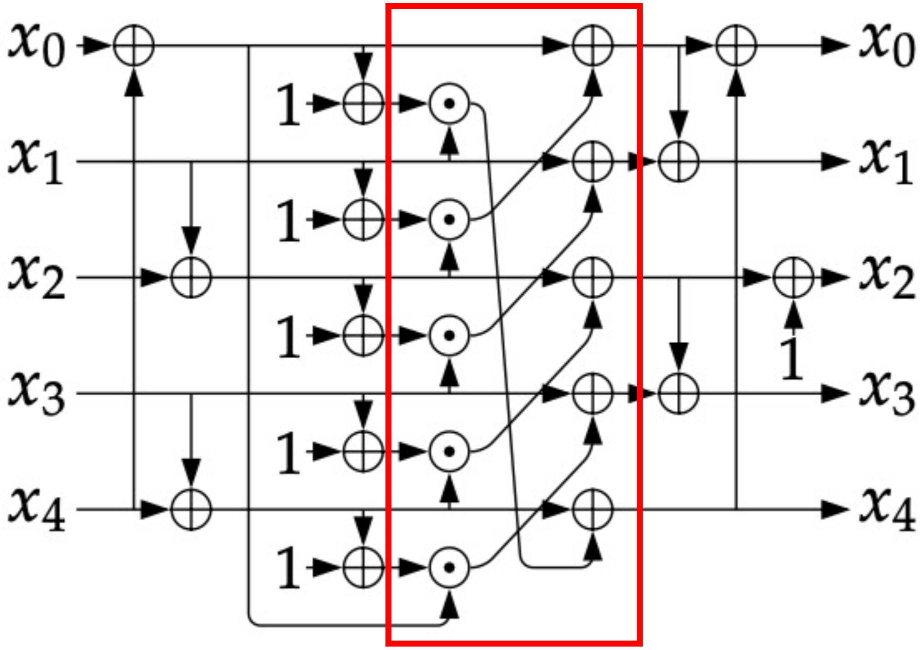
We implement S-box quantum circuits based on **Boolean expression** using quantum gates.



$$
\begin{aligned}
&x_0 = x_0 \oplus x_4, \quad x_4 = x_4 \oplus x_3, \quad x_2 = x_2 \oplus x_1, \\
&t_0 = x_0, \quad t_1 = x_1, \quad t_2 = x_2, \quad t_3 = x_3, \quad t_4 = x_4, \\
&t_0 = \sim t_0, \quad t_1 = \sim t_1, \quad t_2 = \sim t_2, \quad t_3 = \sim t_3, \quad t_4 = \sim t_4, \\
&t_0 = t_0 \cdot x_1, \quad t_1 = t_1 \cdot x_2, \quad t_2 = t_2 \cdot x_3, \quad t_3 = t_3 \cdot x_4, \quad t_4 = t_4 \cdot x_0, \\
&x_0 = x_0 \oplus t_1, \quad x_1 = x_1 \oplus t_2, \quad x_2 = x_2 \oplus t_3, \quad x_3 = x_3 \oplus t_4, \quad x_4 = x_4 \oplus t_0, \\
&x_1 = x_1 \oplus x_0, \quad x_0 = x_0 \oplus x_4, \quad x_3 = x_3 \oplus x_2, \quad x_2 = \sim x_2.
\end{aligned} \tag{3}
$$

Proposed Method

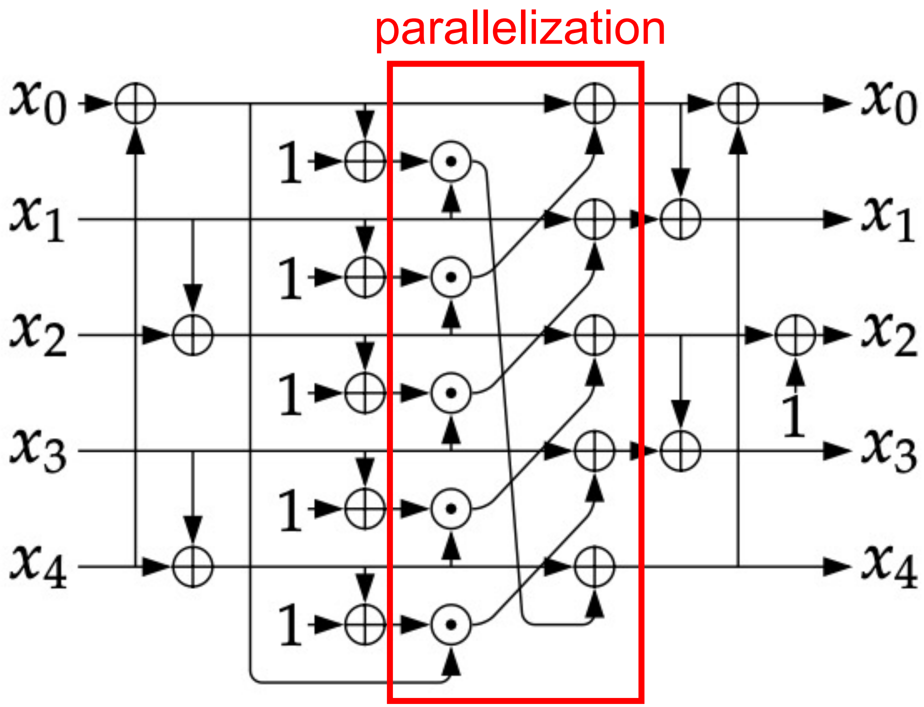
- **Implementation (with Parallelization) of S-box.**
  - We need ancilla qubits(t_0~t_4) for combinations of AND and XOR operations.
  - 64 S-boxes in the substitution layer → a total 320(5 x 64) ancilla qubits.
  - Toffoli gates are executed **sequentially** → **increasing toffoli depth**

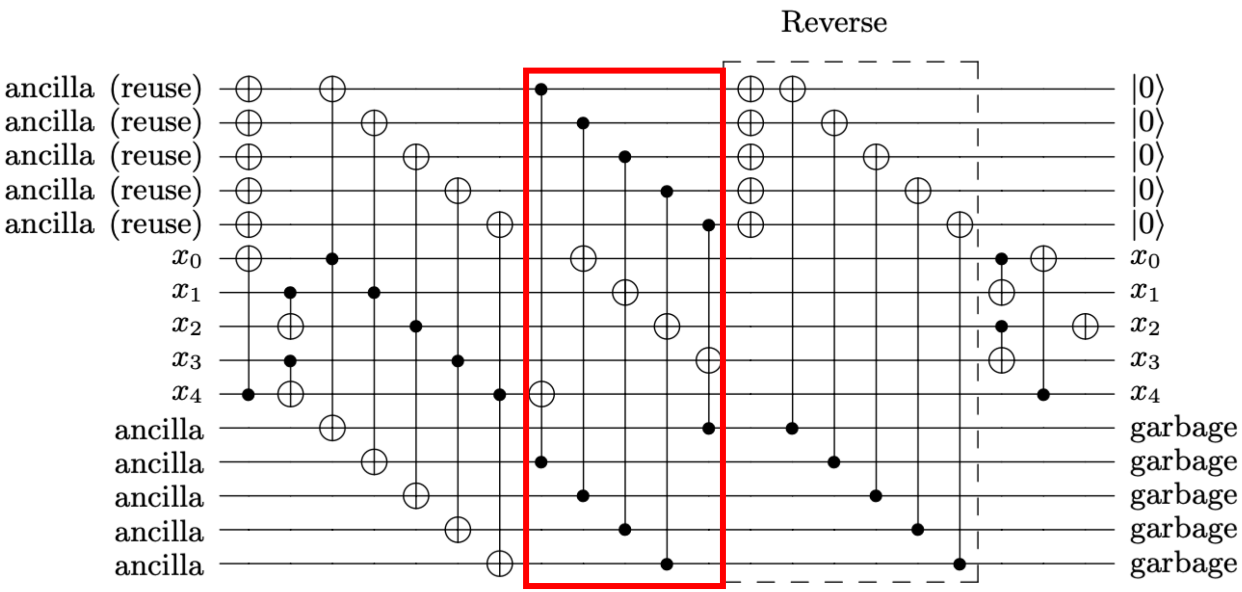


$$
\begin{aligned}
&x_0 = x_0 \oplus x_4, \quad x_4 = x_4 \oplus x_3, \quad x_2 = x_2 \oplus x_1,\\
&\boxed{t_0 = x_0, \quad t_1 = x_1, \quad t_2 = x_2, \quad t_3 = x_3, \quad t_4 = x_4,}\\
&t_0 =\sim t_0, \quad t_1 = \sim t_1, \quad t_2 = \sim t_2, \quad t_3 = \sim t_3, \quad t_4 = \sim t_4,\\
&t_0 = t_0 \cdot x_1, \quad t_1 = t_1 \cdot x_2, \quad t_2 = t_2 \cdot x_3, \quad t_3 = t_3 \cdot x_4, \quad t_4 = t_4 \cdot x_0,\\
&x_0 = x_0 \oplus t_1, \quad x_1 = x_1 \oplus t_2, \quad x_2 = x_2 \oplus t_3, \quad x_3 = x_3 \oplus t_4, \quad x_4 = x_4 \oplus t_0,\\
&x_1 = x_1 \oplus x_0, \quad x_0 = x_0 \oplus x_4, \quad x_3 = x_3 \oplus x_2, \quad x_2 = \sim x_2.
\end{aligned} \tag{3}
$$

Proposed Method

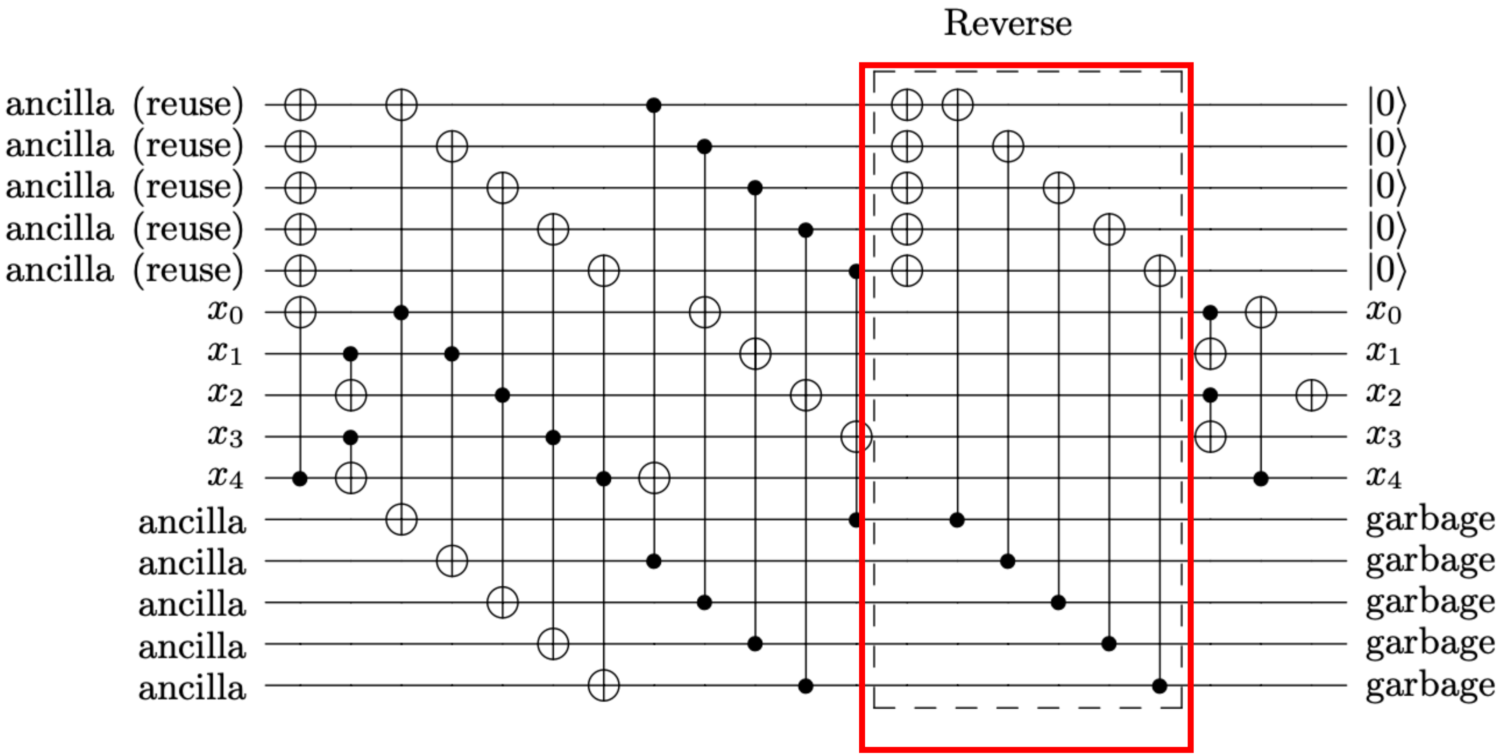
- **Implementation (with Parallelization of S-box)**
  - We optimize our implementation by **parallelizing all Toffoli gates**.
    → **Toffoli depth is one**
  - We allocate two ancilla qubit sets(320 x 2).





Proposed Method

- **Reusing ancilla set with reverse operation**
  - We reuse the one ancilla qubit sets through **reverse operation.**
    → Only **the one initial allocation** of 320 qubits is required.



Proposed Method

- **Quantum Implementation of Linear Layer**
  - In [18], various implementation approaches for the linear layer of ASCON are presented.
  - The naïve implementation requires a higher qubit count, it provides a lower-depth quantum circuit.

    → Aligned with our goal (low depth)

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

In-place

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

Proposed Method

- **Quantum Implementation of Linear Layer**
  - We choose to implement the quantum circuit of the linear layer with **additional qubits**.
  - To store the output of the linear layer, **320 ancilla qubits** are allocated for each round (i.e., out-of-place) in our method.
  - To enhance optimization beyond the naive implementation, we consider **the order of CNOT gates** during implementation.

→ Our implementation of Linear Layer achieves the lowest depth.

$$
\begin{aligned}
\boxed{x_0} &\leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28),\\
\boxed{x_1} &\leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39),\\
\boxed{x_2} &\leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6),\\
\boxed{x_3} &\leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17),\\
\boxed{x_4} &\leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41).
\end{aligned}
$$

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

Proposed Method

- **Constructing ASCON AEAD Quantum Circuit.**

Algorithm 1: Quantum circuit implementation of ASCON-128.

Input: $S = x_0||x_1||x_2||x_3||x_4$, pt, A , $ancilla$
Output: ct , T

$S \leftarrow \mathsf{Permutation}^a(S, ancilla)$ ▷ Initialization
$\boxed{x_3 \leftarrow \text{CNOT64}(key_0,\ x_3)}$
$\boxed{x_4 \leftarrow \text{CNOT64}(key_1,\ x_4)}$

$x_0[32:64] \leftarrow \text{CNOT32}(A,\ x_0[32:64])$ ▷ Processing Associated Data

$\boxed{x_0[31] \leftarrow \text{NOT}(x_0[31])}$ ▷ $A||1||0^r - 1 - (|A| \pmod r))$ XORed with x_0

$S \leftarrow \mathsf{Permutation}^b(S, ancilla)$

$x_4[0] \leftarrow \text{NOT}(x_4[0])$ ▷ Last bit of S XORed with 1

$x_0[32:64] \leftarrow \text{CNOT32}(pt,\ x_0[32:64])$ ▷ Processing Plaintext
$ct \leftarrow$ allocate new 32 qubits
$ct \leftarrow x_0[32:64]$

$\boxed{x_0[31] \leftarrow \text{NOT}(x_0[31])}$ ▷ $pt||1||0^{r-1-(|A| \pmod r))}$ XORed with x_0

$x_1 \leftarrow \text{CNOT64}(key_0,\ x_1)$ ▷ Finalization
$x_2 \leftarrow \text{CNOT64}(key_1,\ x_2)$

$S \leftarrow \mathsf{Permutation}^a(S, ancilla)$

$x_3 \leftarrow \text{CNOT64}(key_0,\ x_3)$
$x_4 \leftarrow \text{CNOT64}(key_1,\ x_4)$

$T \leftarrow x_3||x_4$
return ct,T

- All phases include permutation.
- In initialization, align the key qubits with S.
  → padding with zero
  → **XOR with 0 is an identity operation**, only the least significant 128 qubits(x3,x4) need to be XORed.
- During the associated data processing and plaintext processing, the input data are processed in blocks 64 qubits.
  → padding(a single 1 and the least number of 0s).
  →The XOR with 1 is same as **NOT operation.(**X gate)

Performance

- **Estimation of quantum resources required for ASCON-128.**

Our implementation achieves **a low Toffoli depth** but requires a **high number of qubits** which is a result of the trade-off between qubit count and depth.

→ For the trade-off, we report the **TD-M** and **FD-M** cost.
(TD-M : Toffoli Depth x qubit, FD-M : Full Depth x qubit)

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.

NCT(NOT, CNOT,Toffoli) level.

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.

Clifford + T level.

Evaluation

- **Grover's key search**
  - Grover's key search cost: the quantum resources x 2 x $\left[\frac{\pi}{4}\sqrt{2^k}\right]$
  - The quantum attack cost for ASCON-128 is $1.857 \cdot 2^{156}$

→ ASCON-128 can be evaluated as achieving post- quantum security Level1

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.

Conclusion

- This is the **first optimized implementation** of the ASCON-AEAD quantum circuit.
- We utilize multiple methodologies to reduce **Toffoli and full depths**.
- Our ASCON-128 quantum circuit achieves post-quantum security **Level 1**.
- The implementation techniques presented in this paper have the potential to be applied to other quantum circuit implementations of ciphers.

Q & A