Quantum Implementation of LSH

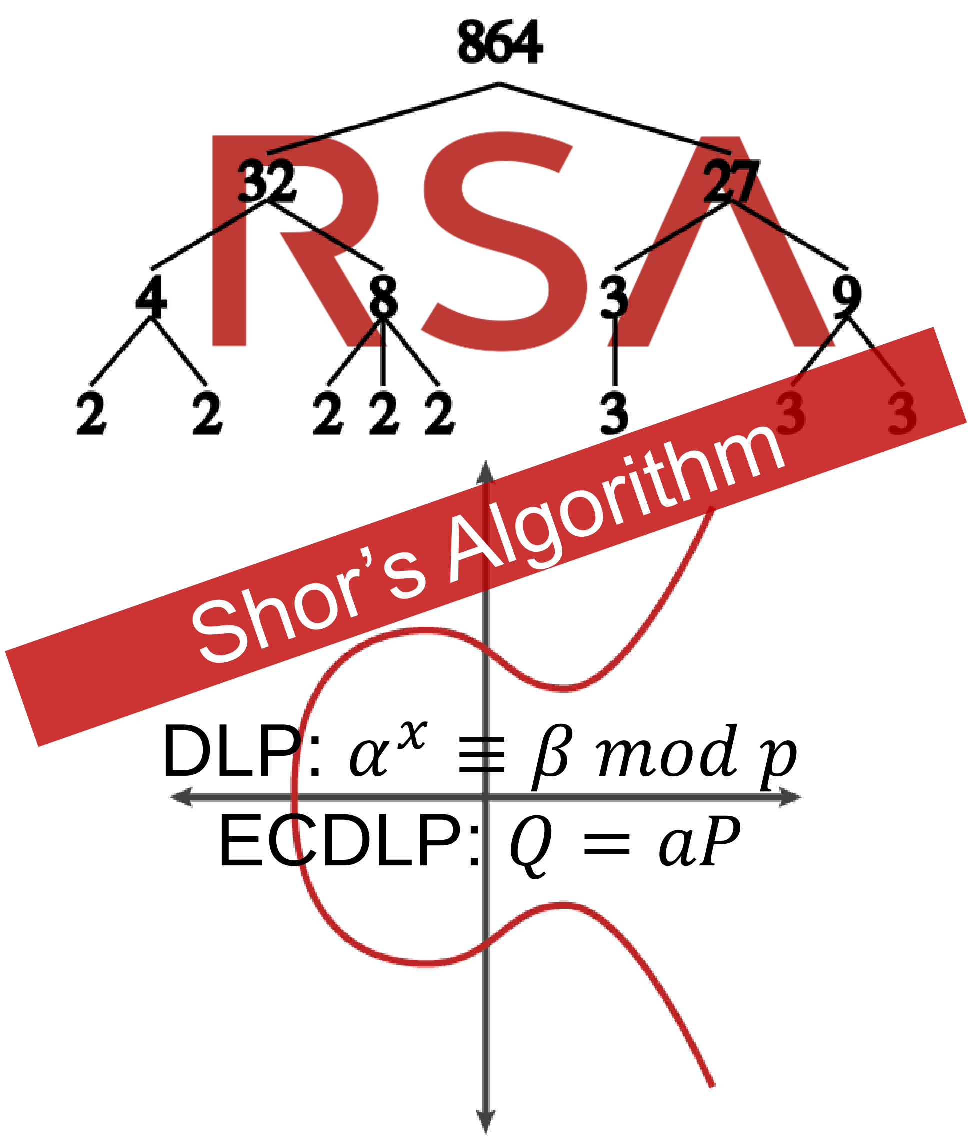
Yujin Oh, Kyungbae Jang, Hwajeong Seo
Hansung University

HANSUNG UNIVERSITY 한성대학교

CryptoCraft LAB

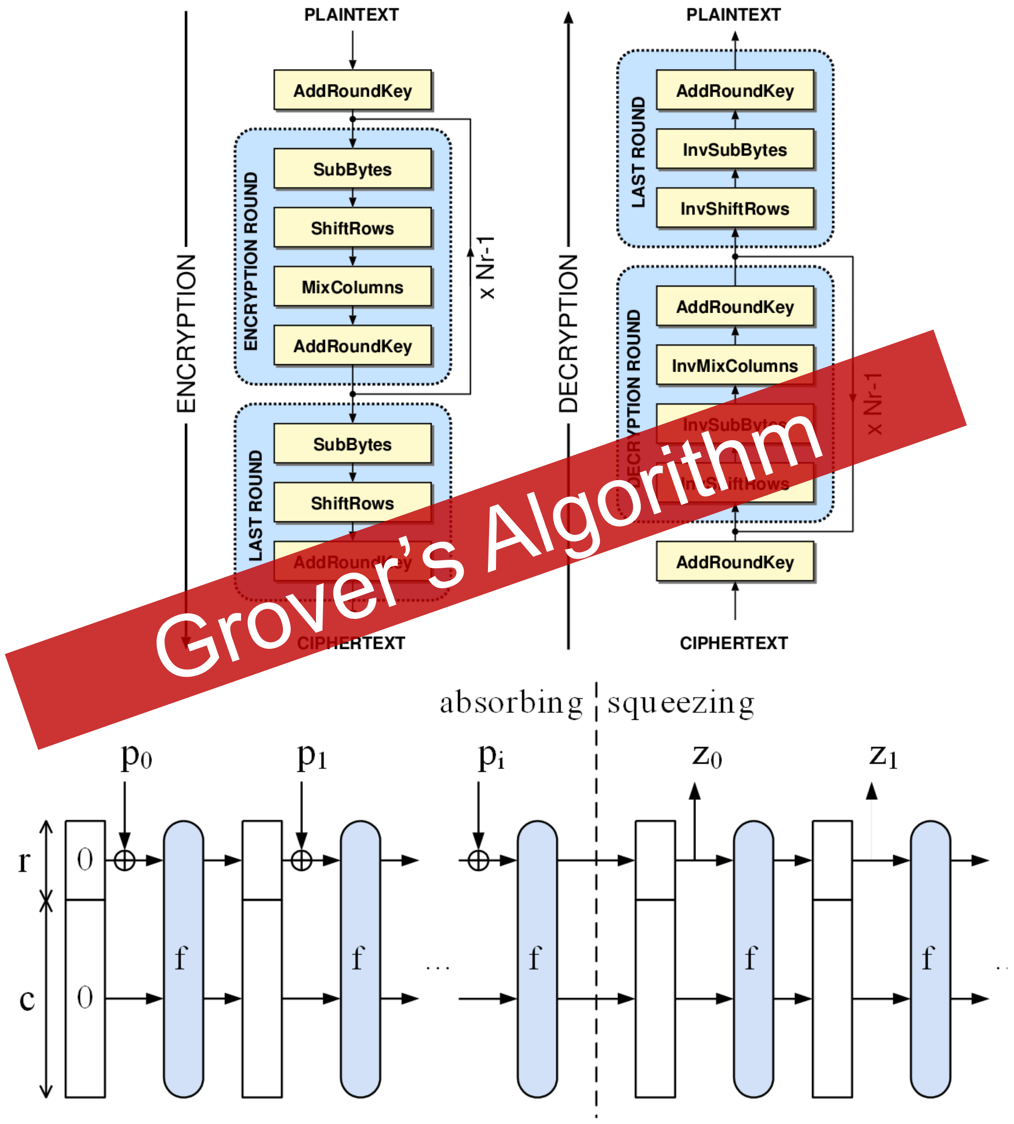
- Introduction
- Background
- Proposed Method
- Performance & Evaluation
- Conclusion

Introduction



DLP: $\alpha^x \equiv \beta \bmod p$

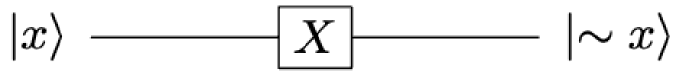
ECDLP: $Q = aP$



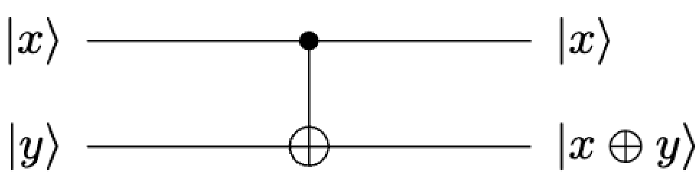
Background : Quantum gates

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

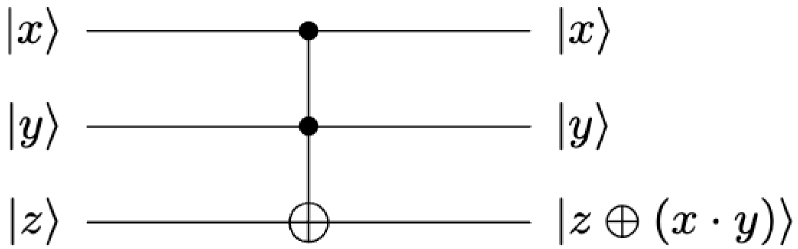
(a) H gate



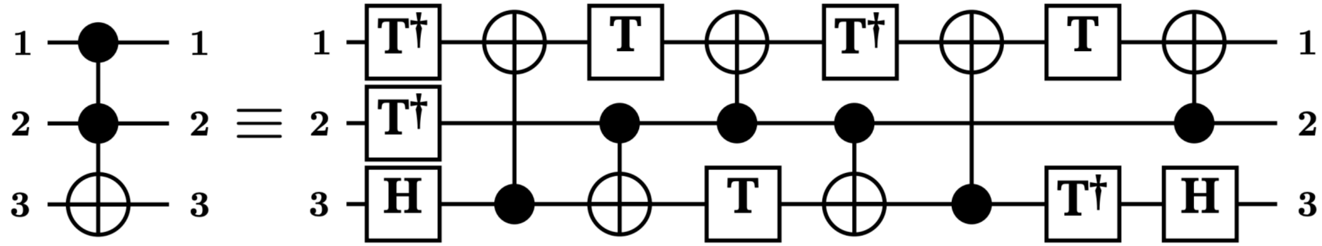
(b) X gate



(c) CNOT gate



(d) Toffoli gate



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

Background : Grover's algorithm

- Grover's Algorithm

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

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

2. The target function is placed in the oracle and returns the solution using the superposition state of input. If the quantum circuit finds a solution for the target function, the amplitude of the specific input in a superpositio n state changes negatively.

$$f(x) = \begin{cases} 1 \text{ if } \mathrm{Hash}(x) = \text{target output} \\ 0 \text{ if } \mathrm{Hash}(x) \neq \text{target output} \end{cases}$$

3. The diffusion operator enhance the probability for measuring the solution returned by the oracle.

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

Background : Quantum collision search

- Quantum collision search using Grover algorithm
  - There are various quantum collision attack using Grover algorithm.

  - BHT algorithm
    - The search complexity of $O(2^{\frac{n}{3}})$, quantum memory $O(2^{\frac{2n}{3}})$.

  - **CNS algorithm**
    - The search complexity of $O(2^{\frac{2n}{5}})$, classical memory $O(2^{\frac{n}{5}})$.
    - Note that the **CNS algorithm** can be parallelized to reduce the search complexity of $\boldsymbol{O(2^{\frac{n}{5}})}$.
    - By utilizing 2s quantum instances in parallel
      - The search complexity for finding collisions is reduced to $\boldsymbol{O(2^{\frac{2n}{5}-\frac{3s}{5}})}$**, with** $\boldsymbol{s \leq \frac{n}{4}}$**.**
      - In [9], the authors defined a parallelization strength of $\boldsymbol{s = \frac{n}{6}}$
      - Following this approach, we also define a parallelization strength of $s = \frac{n}{6}$

[9] Jang, K., Lim, S., Oh, Y., Kim, H., Baksi, A., Chakraborty, S., Seo, H.: Quantum implementation and analysis of sha-2 and sha-3. Cryptology ePrint Archive (2024) 4, 9

Background : LSH

- Description of LSH
  - **LSH** is a **Korean cryptographic hash algorithm** included among the validation subjects of the **KCMVP**.
  - Initialization
    - A given input message undergoes **one-zero padding**.
    - Following this, the padded input message is divided into 32-bit word array messages.
  - Compression
    - MsgExp, Step (MsgAdd, Mix, WordPerm)
  - Finalization
    - The finalization function produces an n-bit hash value.

$$\mathbf{h} \leftarrow (CV^t[0] \oplus CV^t[8], ..., CV^t[7] \oplus CV^t[15])$$
$$\mathbf{h} = (h[0] \,||\, ... \,||\, h[w-1])$$
$$h \leftarrow (h[0] \,||\, ... \,||\, h[w-1])_{[0:n-1]}$$

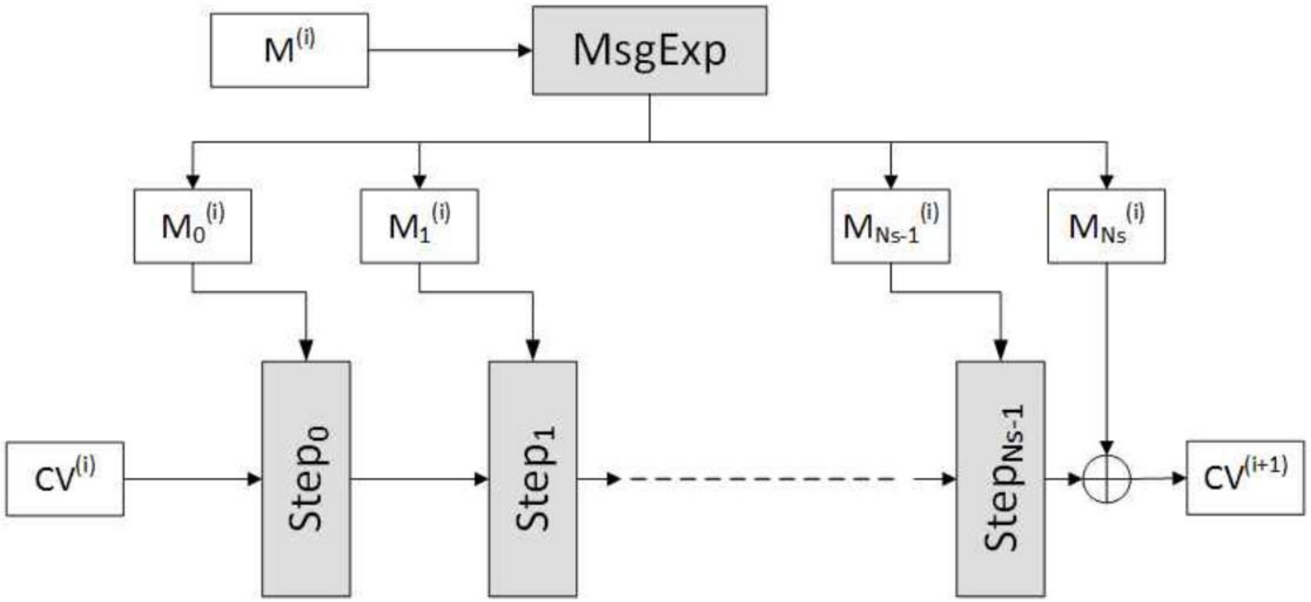
Background : LSH

- Description of LSH (Compression function)
  - MsgExp

$$\mathbf{M}_0^{(i)} \leftarrow (M^{(i)}[0], ..., M^{(i)}[15]),\ \mathbf{M}_1^{(i)} \leftarrow (M^{(i)}[16], ..., M^{(i)}[31])$$
$$\mathbf{M}_j^{(i)} \leftarrow (M_j^{(i)}[0], ..., M_j^{(i)}[15])_{j=2}^{N_s}$$
$$M_j^{(i)}[l] \leftarrow M_{j-1}^{(i)}[l] \boxplus M_{j-2}^{(i)}[\tau(l)]\ for\ 0 \leq l \leq 16$$

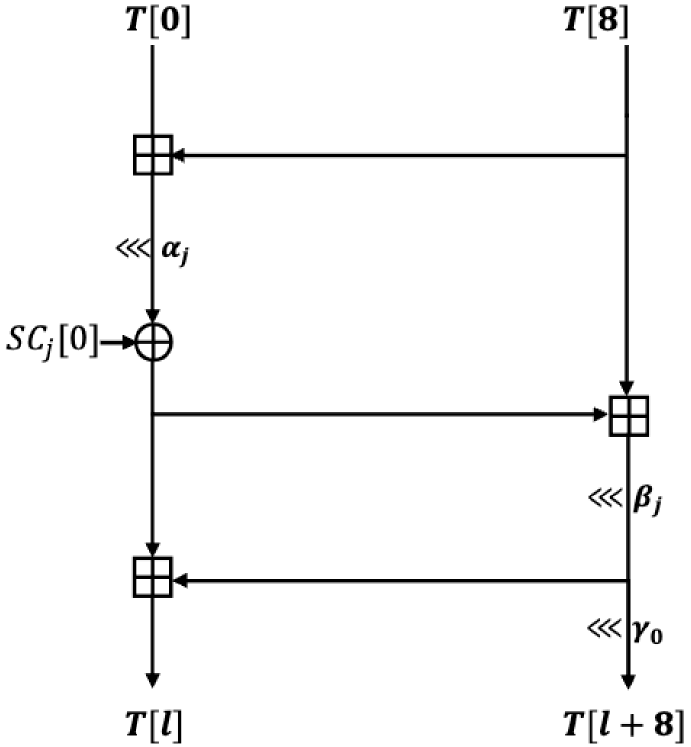
Table 1: The permutation τ and σ

l	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\tau(l)$	3	2	0	1	7	4	5	6	11	10	8	9	15	12	13	14
$\sigma(l)$	6	4	5	7	12	15	14	13	2	0	1	3	8	11	10	9



Background : LSH

- Description of LSH (Compression function)
  - Step
    - MsgADD
      - Input : $CV^{(i)} = T[0], \dots, T[15]$ and $M_j^{(i)} = \left(M_j^{(i)}[0], \dots, M_j^{(i)}[15]\right)_{j=2}^{N_s}$
      - MSGADD(T,M) ← $T[0] \oplus M[0], \dots, T[15] \oplus M[15])$
    - Mix
      - $(T[l], T[l+8]) \leftarrow Mix_{j,l}\,(T[l], T[l+8])\ for\ 0 \le l < 8$
    - WordPerm
      - $WordPerm(X) = X[\sigma(0)], \cdots, X[\sigma(15)]$



Mix function

Table 2: Bit rotation amounts: α_j, β_j and γ_l

Algorithm	j	α_j	β_j	γ_0	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7
LSH-256-n	even	29	1	0	8	16	24	24	16	8	0
	odd	5	17								
LSH-512-n	even	23	59	0	16	32	48	8	24	40	56
	odd	7	3								

Table 1: The permutation τ and σ

l	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\tau(l)$	3	2	0	1	7	4	5	6	11	10	8	9	15	12	13	14
$\sigma(l)$	6	4	5	7	12	15	14	13	2	0	1	3	8	11	10	9

Proposed Method

- Our main focus is to **optimize the circuit depth** of LSH for the efficiency of the **Grover collision attack**.

- In quantum circuit for **LSH**, the most resources are generally required for **adders**.

  We use depth-optimized adders and parallelization.

- For the sake of simplicity, we primarily focus on explaining LSH-256-256.

- We set the input length to be equal to the hash length for implementation.

Proposed Method

- **Quantum adder for optimizing the depth**
  - To implement the MsgExp function and Mix function, we use a quantum adder.
  - Commonly used types of quantum adders : **ripple-carry adder (RCA)** and **carry-lookahead adder (CLA).**

  - The RCA adder operates in a sequential manner, where it calculates the carry-out from the previous stage before proceeding with the addition in the next stage.

    → Leads to high depth

  - The CLA operates accelerates addition **by pre-computing carry values** for each stage.

    → **Because of parallel process, reduce the depth.**

Proposed Method

- **Quantum adder for optimizing the depth**
  - We utilize a **Draper adder** [5], which is a carry-lookahead adder.
    - This adder can be implemented both in-place and out-of-place
  - The out-of-place Draper adder has about half the depth compared to the in-place adder
    - But requires 32-bit output qubits for each adder.
    - 32,768 (1024 × 32) qubits are garbage qubits, with a total of 1024 adders.

      → **We opt for the in-place adder.**
  - Draper in-place adders

    → We can reuse **all ancilla qubits** (**53** qubits) except for the input and output qubits in other operations

Table 3: Comparison of quantum resources required for adder (32-bit).

Adder	Operation	#CNOT	#Toffoli	Toffoli depth	#Qubit (reuse)	Depth
Cuccaro [4]	in-place	153	61	61	65 (1)	66
Draper [5]	in-place	123	254	22	117 (53)	28
	out-of-place	94	127	11	118 (22)	14

※: Estimation of undecomposed resources

[5] Draper, T.G., Kutin, S.A., Rains, E.M., Svore, K.M.: A logarithmic-depth quantum carry-lookahead adder. arXiv preprint quant-ph/0406142 (2004

Proposed Method

- **Parallel addition of MsgExp and Mix Functions**
  - 16 adders are needed to update $M_i^{(i)}$

$$\mathbf{M}_0^{(i)} \leftarrow (M^{(i)}[0], ..., M^{(i)}[15]),\ \mathbf{M}_1^{(i)} \leftarrow (M^{(i)}[16], ..., M^{(i)}[31])$$
$$\mathbf{M}_j^{(i)} \leftarrow (M_j^{(i)}[0], ..., M_j^{(i)}[15])_{j=2}^{N_s}$$
$$M_j^{(i)}[l] \leftarrow M_{j-1}^{(i)}[l] \boxplus M_{j-2}^{(i)}[\tau(l)]\ for\, 0 \le l \le 16$$

  - We can initially allocate 53 ancilla qubits and reuse them throughout.

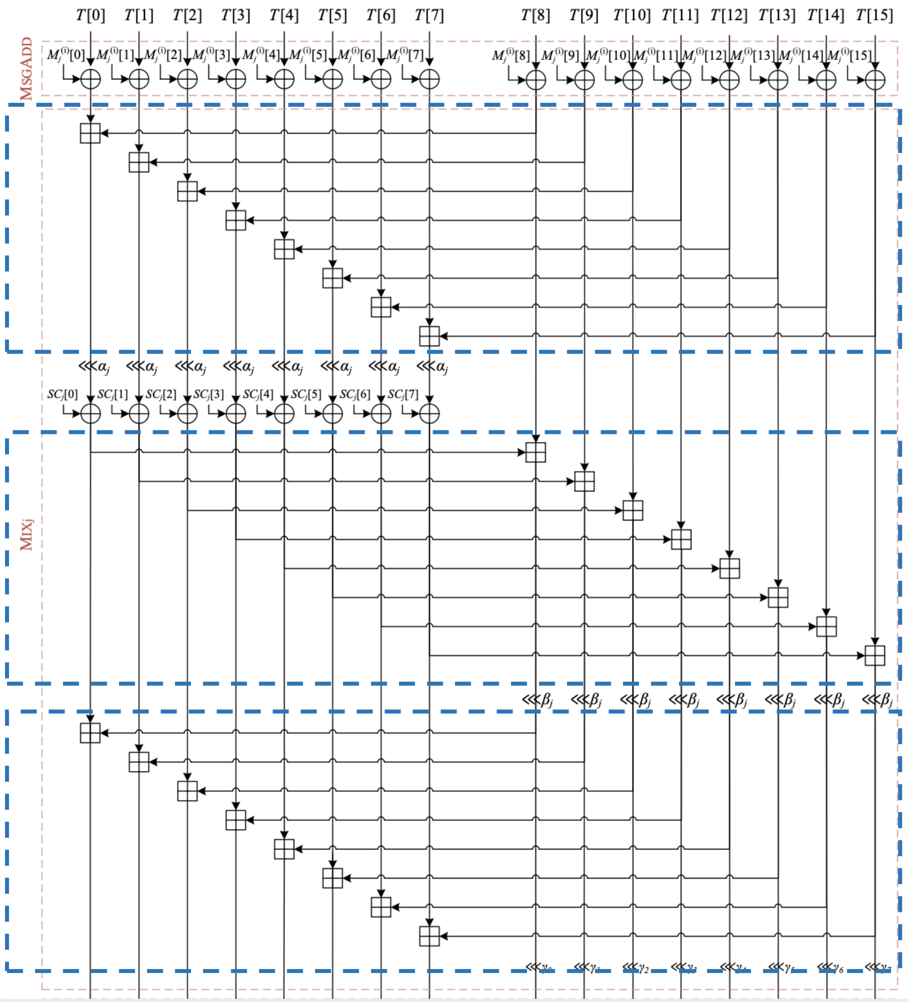
    → The adders are executed sequentially, increasing the depth of the circuit.

  - **To optimize the circuit depth which is our purpose, we employ addition in parallel by allocating more ancilla qubits.**

    → 848 (16 × 53) ancilla qubits are required.

Proposed Method

- **Parallel addition of MsgExp and Mix Functions**
  - **24 (8 × 3) adders** are used and **8** out of the 24 adders can be operated simultaneously
  - In this scenario, the **ancilla qubits** used in the MsgExp function **can be reused**
    - → There is no need to allocate additional ancilla qubits for the adders in the Mix function.
    - → 848 ancilla qubits are initially allocated at once.



However, due to the reuse of qubits, the depth may increase.

Proposed Method

- **Parallel addition of MsgExp and Mix Functions**
  - Table 4 shows the comparison of quantum resources required for MsgExp and Mix function.
  - The parallel operations greatly reduce the **toffoli depth** and **full depth** compared to the sequential operations.

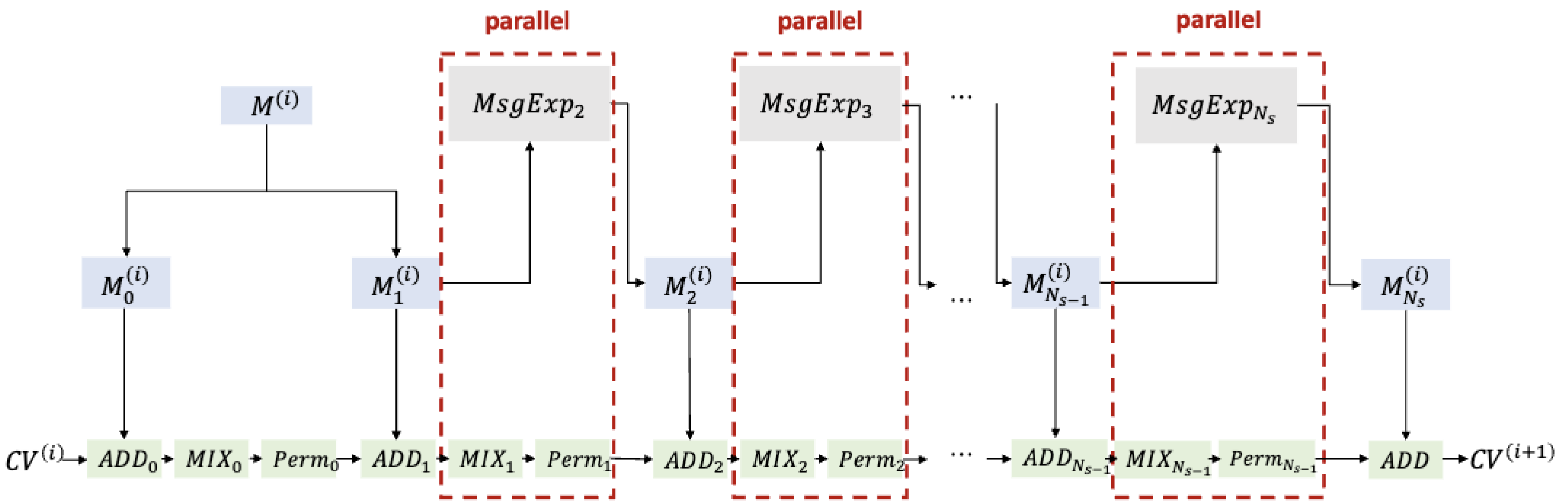
Table 4: Comparison of quantum resources required for each component.

Function	Operation	#CNOT	#Toffoli	Toffoli depth	#Qubit	Depth
MsgExp	Sequential	1,968	4,064	352	1,077	433
	Parallel	1,968	4,064	**22**	1,872	**28**
Mix	Sequential	2,952	6,096	528	565	649
	Parallel	2,952	6,096	**66**	936	**84**

Proposed Method

- **Combined Architecture of Compress Function**
  - The MsgExp function and the Mix function **can operate independently**.
  - However, due to the ancilla qubit reuse in the Mix function, these functions cannot operate in parallel.
    - This architecture can reduce the number of qubits

      → But it increases the circuit depth due to the sequential operations of high complexity.
  - **To optimize the circuit depth,** we execute the MsgExp function and Mix function **in parallel by allocating additional ancilla qubits.**



Proposed Method

- **Combined Architecture of Compress Function**
  - In previous work [17], Song et al. conducted **sequential operations** in the Compression function.
  - In contrast, **we implements the Mix and MsgExp functions in parallel.**
    - Specifically, the $\boldsymbol{i\text{-}th}$ **Mix function** and the $\boldsymbol{i+1\text{-}th}$ **MsgExp function** can execute **in parallel**.

      → **effectively reducing the circuit depth.**
  - To enable this parallel process, we additionally allocate **424 (8 × 53) ancilla qubits** for Mix function.

    → **We initially allocate 1,272(848+424) ancilla qubits at once and reuse them each round.**

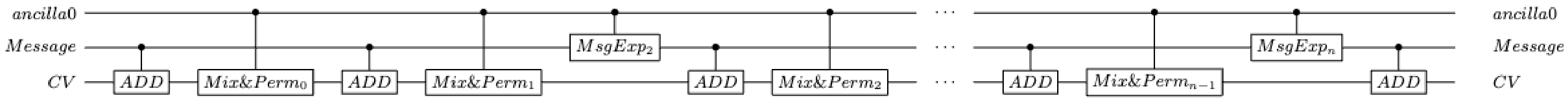


Fig. 4: Compression function in [17] using a sequential process

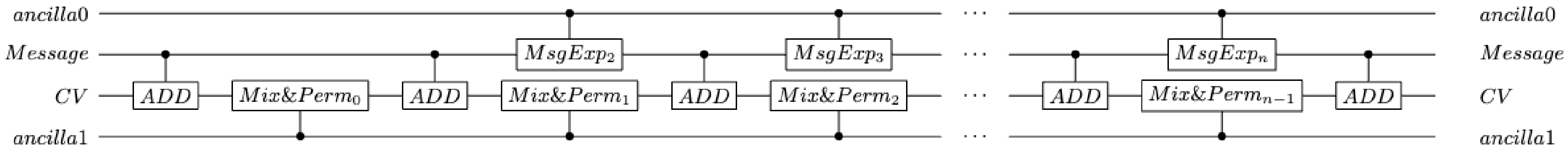


Fig. 5: Proposed parallel Compression function architecture

[17] Song, G., Jang, K., Kim, H., Seo, H.: A parallel quantum circuit implementations of lsh hash function for use with grover's algorithm. Applied Sciences 12(21), 10891 (2022) 8, 9, 11

Proposed Method

- **Combined Architecture of Compress Function**
  - **By allocating two sets of ancilla qubits**

    → the even-round Mix with the odd-round MsgExp

    and the odd-round Mix with the even-round MsgExp in parallel.
  - Only **the depths of the Mix functions are estimated**

    →They have a **higher depth** compared to the MsgExp function.
  - The parallel process demonstrates **lower depth** compared to processing them sequentially.

Function	Operation	#CNOT	#Toffoli	Toffoli depth	#Qubit	Depth
Compression	Sequential	139,776	260,096	2,266	2,384	2,873
	Parallel	139,776	260,096	**1,716**	2,808	**2,198**

※: Estimation of undecomposed resources

Algorithm 1: Quantum circuit implementation of Compress function.

Input: M_{even}, M_{odd} CV, α, β, SC, $ancilla_0$, $ancilla_1$

Output: M_{even}, M_{odd}, CV, 424 qubit array-$ancilla_0$, 848 qubit array-$ancilla_1$

$CV \leftarrow$ MsgAdd(M_{even}, CV)
$CV \leftarrow$ Mix(CV, α_{even}, β_{even}, SC, $ancilla_0$)
$CV \leftarrow$ WordPerm(CV)

$CV \leftarrow$ MsgAdd(M_{odd}, CV)
$CV \leftarrow$ Mix(CV, α_{odd}, β_{odd}, SC, $ancilla_0$) ▷ Parallelization 1
$CV \leftarrow$ WordPerm(CV)

for $1 \leq i \leq 13$ **do**
  $M_{even} \leftarrow$ MsgExp(M_{even}, M_{odd}, $ancilla_1$) ▷ Parallelization 1
  $CV \leftarrow$ MsgAdd(M_{even}, CV)
  $CV \leftarrow$ Mix(CV, α_{even}, β_{even}, SC, $ancilla_0$) ▷ Parallelization 2
  $CV \leftarrow$ WordPerm(CV)

  $M_{odd} \leftarrow$ MsgExp(M_{even}, M_{odd}, $ancilla_1$) ▷ Parallelization 2
  $CV \leftarrow$ MsgAdd(M_{odd}, CV)
  $CV \leftarrow$ Mix(CV, α_{odd}, β_{odd}, SC, $ancilla_0$) ▷ Parallelization 1
  $CV \leftarrow$ WordPerm(CV)
end for

$M_{even} \leftarrow$ MsgExp(M_{even}, M_{odd}, $ancilla_1$) ▷ Parallelization 1
$CV \leftarrow$ MsgAdd(M_{even}, CV)

return CV

Performance

- **Estiamation of quantum resources required for LSH**
  - For LSH-256-n and LSH-512-n, all resource costs except for the X gates are identical, respectively.

    → We will only compare LSH-256-256 and LSH-512-512.
  - Applying the Draper adder further **increases the qubit** usage, but it significantly **reduces the full depth**.
  - For the trade-off, we report the TD-M, FD-M, TD^2-M, FD^2-M cost. (TD: Toffoli depth, FD: Full depth, M: qubit)

    → **Our proposed quantum circuit achieves the optimized performance across all trade-off metrics.**

Table 6: Quantum resources required for implementations of LSH.

	Cipher	Source	#CNOT	#1qCliff	#T	Toffoli depth (TD)	#Qubit (M)	Full depth (FD)	TD-M	FD-M	TD^2-M	FD^2-M
Same adder	LSH-256-256	[17]	545,536	187,813	437,248	6,283	1,552	50,758	$1.16 \cdot 2^{23}$	$1.17 \cdot 2^{26}$	$1.78 \cdot 2^{35}$	$1.82 \cdot 2^{41}$
		Ours-CDKM	545,536	187813	437,248	**4,758**	1,560	**38,483**	$\mathbf{1.77 \cdot 2^{22}}$	$\mathbf{1.79 \cdot 2^{25}}$	$\mathbf{1.03 \cdot 2^{35}}$	$\mathbf{1.05 \cdot 2^{41}}$
		Ours-Draper	1,700,608	306,947	1,820,672	**1,716**	2,808	**13,647**	$\mathbf{1.15 \cdot 2^{22}}$	$\mathbf{1.14 \cdot 2^{25}}$	$\mathbf{1.93 \cdot 2^{32}}$	$\mathbf{1.90 \cdot 2^{38}}$
Same adder	LSH-512-512	[17]	1,203,760	418,369	966,000	13,875	3,088	111,532	$1.28 \cdot 2^{25}$	$1.28 \cdot 2^{28}$	$1.08 \cdot 2^{39}$	$1.09 \cdot 2^{45}$
		Ours-CDKM	1,203,760	418,369	966,000	**10,500**	3,096	**84,451**	$\mathbf{1.94 \cdot 2^{24}}$	$\mathbf{1.95 \cdot 2^{27}}$	$\mathbf{1.24 \cdot 2^{38}}$	$\mathbf{1.26 \cdot 2^{44}}$
		Ours-Draper	4,030,000	736,569	2,614,473	**2,028**	5,832	**17,385**	$\mathbf{1.41 \cdot 2^{23}}$	$\mathbf{1.51 \cdot 2^{26}}$	$\mathbf{1.40 \cdot 2^{34}}$	$\mathbf{1.60 \cdot 2^{40}}$

[17] Song, G., Jang, K., Kim, H., Seo, H.: A parallel quantum circuit implementations of lsh hash function for use with grover's algorithm. Applied Sciences 12(21), 10891 (2022) 8, 9, 11

Evaluation

- **Grover collision search**
  - To estimate the collision attack cost for LSH, we adopt the **CNS algorithm**.
  - The CNS algorithm has the complexity of $\boldsymbol{O}(\boldsymbol{2}^{\frac{2n}{5}-\frac{3s}{5}})$ $(s \leq \frac{n}{4})$.
  - We set $\boldsymbol{s} = \frac{n}{6}$ to define suitable criteria for NIST post-quantum security levels, following that approach[9].
  - The quantum attack cost for LSH is approximately $\boldsymbol{2} \times \boldsymbol{2}^{(\frac{2n}{5}-\frac{3s}{5})} \times$ **quantum circuit resources**, excluding qubits.

Table 7: Costs of the Grover's collision search for LSH.

Cipher	#Gate (G)	Full depth (FD)	T-depth (Td)	#Qubit (M)	G-FD	FD-M	Td-M	FD^2-M	Td^2-M
LSH-256-224	$1.65 \cdot 2^{89}$	$1.5 \cdot 2^{81}$	$1.51 \cdot 2^{80}$	$1.72 \cdot 2^{48}$	$\mathbf{1.23 \cdot 2^{171}}$	$1.29 \cdot 2^{130}$	$1.3 \cdot 2^{129}$	$1.95 \cdot 2^{211}$	$1.97 \cdot 2^{209}$
LSH-256-256	$1.25 \cdot 2^{99}$	$1.13 \cdot 2^{91}$	$1.14 \cdot 2^{90}$	$1.08 \cdot 2^{54}$	$\mathbf{1.42 \cdot 2^{190}}$	$1.23 \cdot 2^{145}$	$1.24 \cdot 2^{144}$	$1.41 \cdot 2^{236}$	$1.42 \cdot 2^{234}$
LSH-512-224	$1.96 \cdot 2^{90}$	$1.91 \cdot 2^{81}$	$1.78 \cdot 2^{80}$	$1.79 \cdot 2^{49}$	$\mathbf{1.87 \cdot 2^{172}}$	$1.71 \cdot 2^{131}$	$1.6 \cdot 2^{130}$	$1.64 \cdot 2^{213}$	$1.43 \cdot 2^{211}$
LSH-512-256	$1.49 \cdot 2^{100}$	$1.45 \cdot 2^{91}$	$1.35 \cdot 2^{90}$	$1.13 \cdot 2^{55}$	$\mathbf{1.07 \cdot 2^{192}}$	$1.64 \cdot 2^{146}$	$1.53 \cdot 2^{145}$	$1.18 \cdot 2^{238}$	$1.03 \cdot 2^{236}$
LSH-512-384	$1.96 \cdot 2^{138}$	$1.91 \cdot 2^{129}$	$1.78 \cdot 2^{128}$	$1.42 \cdot 2^{76}$	$\mathbf{1.87 \cdot 2^{268}}$	$1.36 \cdot 2^{206}$	$1.27 \cdot 2^{205}$	$1.3 \cdot 2^{336}$	$1.13 \cdot 2^{334}$
LSH-512-512	$1.29 \cdot 2^{177}$	$1.26 \cdot 2^{168}$	$1.17 \cdot 2^{167}$	$1.79 \cdot 2^{97}$	$\mathbf{1.63 \cdot 2^{345}}$	$1.13 \cdot 2^{266}$	$1.05 \cdot 2^{265}$	$1.43 \cdot 2^{434}$	$1.24 \cdot 2^{432}$

[9] Jang, K., Lim, S., Oh, Y., Kim, H., Baksi, A., Chakraborty, S., Seo, H.: Quantum implementation and analysis of sha-2 and sha-3. Cryptology ePrint Archive (2024) 4, 9

Conclusion

- We focused on **optimizing the depth** of quantum circuits for Korean cryptographic hash function LSH.
- We utilize **optimized quantum adders** and **parallelization.**
- Our implementation of LSH achieves a significant **depth** improvement of over **78.8%** and a **Toffoli depth** improvement of **79.1%** compared to previous work.
- Through the depth-optimized implementation, we also obtain the optimized quantum resources of **Grover collision attack** for LSH.
- If NIST defines criteria for hash functions, we will compare our results with those criteria.

Q & A