Quantum Implementation of LSH

Yujin Oh, Kyungbae Jang, and Hwajeong Seo

Hansung University, Seoul (02876), South Korea

oyj0922@gmail.com, starj1023@gmail.com, hwajeong84@gmail.com

Abstract. As quantum computing progresses, the assessment of cryptographic algorithm resilience against quantum attack gains significance interests in the field of cryptanalysis. Consequently, this paper implements the depth-optimized quantum circuit of Korean hash function (i.e., LSH) and estimates its quantum attack cost in quantum circuits. By utilizing an optimized quantum adder and employing parallelization techniques, the proposed quantum circuit achieves a 78.8% improvement in full depth and a 79.1% improvement in Toffoli depth compared to previous the-state-of art works. In conclusion, based on the implemented quantum circuit, we estimate the resources required for a Grover collision attack and evaluate the post-quantum security of LSH algorithms.

Keywords: Quantum Circuit · Quantum Collision Attack · LSH.

1 Introduction

The advancement of quantum computing presents new challenges and opportunities in cryptography. The parallel processing capability of quantum computers can exponentially enhance the speed of breaking many traditional cryptographic systems. Particularly, it enables finding solutions to classical hard problems such as large-scale factorization and discrete logarithm problems at a much faster rate.

Moreover, the development in quantum computing allows for the discovery and application of new quantum algorithms. These algorithms can be employed to uncover vulnerabilities in currently used classical cryptographic systems. For instance, Shor's algorithm [14] demonstrated the ability to factorize large numbers efficiently, leading to the collapse of security in public-key cryptography schemes such as RSA. Grover's algorithm [7], on the other hand, reduces the time complexity of cryptographic functions such as symmetric key encryption and hash functions.

As quantum computing technology progresses further, there is numerous research underway to assess the security of current cryptographic systems. These studies involve implementing cryptographic systems as quantum circuits and estimating quantum attack costs to assess security strength.

In this paper, we propose a depth-optimized quantum circuit of LSH [11], which is a hash function included as validation subjects in the Korean Cryptographic Module Validation Program (KCMVP). Additionally, based on the quantum circuit, we estimate the cost of collision attack using Grover algorithm for LSH.

2 Background

2.1 Quantum Gates

In this section, we explain the quantum gates to implement our quantum circuit. The Hadamard gate creates superposition states of qubits. The X gate, also known as the Pauli-X gate, operates on a single qubit. It can invert the state of a qubit, transforming $|0\rangle$ state to $|1\rangle$ and $|1\rangle$ state to $|0\rangle$. This operation is similar to the classical NOT gate in traditional computing. The CNOT (Controlled-NOT) gate uses two-qubit gate and performs a NOT operation on the target qubit if the control qubit is in the state$|1\rangle$. If the control qubit is in the $|0\rangle$ state, the target qubit remains unchanged. The Toffoli gate, also known as the CCNOT gate (Controlled-Controlled-NOT), uses two control qubits and one target qubit. The Toffoli gate performs a NOT operation on the target qubit only if both control qubits are in the state $|1\rangle$. Thus, it is similar to the classical AND operation. The Toffoli gate can be decomposed into a combination of gates such as H, CNOT and T gates.

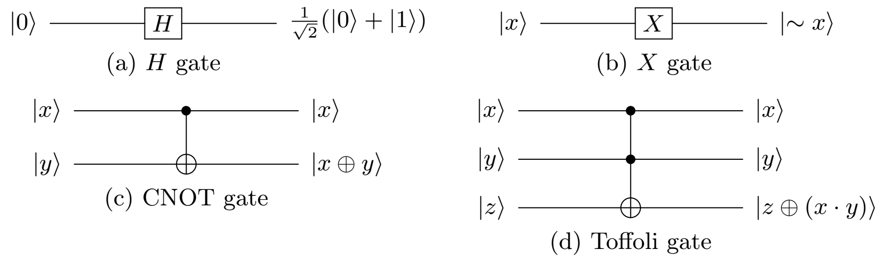

Fig. 1: Quantum gates

2.2 The Grover algorithm

The Grover algorithm can find a solution for the n-qubit data (in a superposition state) with a complexity of $O(\sqrt{2^n})$ (i.e., a square root speedup compared to classical search of $O(2^n)$). As such, Grover's algorithm has an advantage in solving problems with high search complexity. Notably, extensive research has been conducted on Grover's algorithm for block ciphers and hash functions [8,10,13]. Grover's algorithm consists of three major processes; *Input setting, Oracle, Diffusion operator.*

Input setting. H gates are used to prepare an n-qubit in a superposition state ($|\psi\rangle$). As a result, an n-qubit input with a superposition state can represents 2^n cases as probabilities.

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

Oracle The target function (e.g., block ciphers or hash functions) is placed in the oracle and returns the solution using the superposition state of input. To accomplish this, the target function should be implemented using quantum gates (i.e., a quantum circuit). If the quantum circuit finds a solution for the target function (i.e., if $f(x) = 1$), the amplitude of the specific input in a superposition state changes negatively (see Equation 3).

$$f(x) = \begin{cases} 1 \text{ if } \mathrm{Hash}(x) = \text{target output} \\ 0 \text{ if } \mathrm{Hash}(x) \neq \text{target output} \end{cases} \tag{2}$$

$$U_f(|\psi\rangle |-\rangle) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle |-\rangle \tag{3}$$

Diffusion operator The diffusion operator enhance the probability for measuring the solution returned by the oracle. Due to the fixed design of the diffusion operator and relatively low complexity compared to the oracle, it is often neglected for the cost estimation in Grover's search [6,10,8,12].

2.3 Quantum Collision Search

Grover's search for an n-bit key of block ciphers or a pre-image of an n-bit hash output for hash functions can be approached straightforwardly, as the complexity of $O(n)$ in classical computing is reduced to $O(\sqrt{n})$ in quantum computing. However, quantum collision search for hash functions is more complicated and can be approached in various ways.

There are various quantum collision attack algorithms using Grover's algorithm. Among them, the BHT algorithm [2] has a query complexity of $O(2^{n/3})$. However, this algorithm demands a notably large quantum memory, $O(2^{2n/3})$. Also, Bernstein pointed out in [1] that this algorithm is controversial. Considering these aspects, we employ the CNS algorithm [3], which has a query complexity of $O(2^{2n/5})$ and requires only $O(2^{n/5})$ classical memory. Note that the CNS algorithm can be parallelized to reduce the search complexity of $O(2^{n/5})$. By utilizing 2^s quantum instances in parallel, the search complexity for finding collisions is reduced to $O(2^{\frac{2n}{5}-\frac{3s}{5}})$, with $s \leq \frac{n}{4}$. In [9], the authors defined a parallelization strength of $s = n/6$ to estimate the required quantum resources for finding a collision in the SHA-2 and SHA-3 hash functions. Following this approach, we also define a parallelization strength of $s = n/6$ for finding a collision in the LSH hash functions.

2.4 Description of LSH Hash Function

LSH is a Korean cryptographic hash algorithm included among the validation subjects of the KCMVP. The LSH consists of LSH-256-224, LSH-256-256, LSH-512-224, LSH-512-256. LSH-512-384 and LSH-512-512. LSH-256-n operates based on a 32-word and LSH-512-n operates based on a 64-word. LSH operates in three stages: *Initialization*, *Compression*, and *Finalization*.

Initaillization During the *initialization* process, a given input message undergoes one-zero padding. Following this, the padded input message is divided into 32-word array messages. Additionally, in the initialization, the 16-word array hash chaining variables ($CV^{(i)}$) are set as an initialization vector.

Compression The *compression* function consists of MsgExp and Step (MsgAdd, Mix, and WordPerm) functions. The MsgExp function converts a 32-bit word array message into a 16-word array. The MsgExp function process is as follows in Equation 5.

$$\begin{aligned}
&\mathbf{M}_0^{(i)} \leftarrow (M^{(i)}[0], ..., M^{(i)}[15]),\ \mathbf{M}_1^{(i)} \leftarrow (M^{(i)}[16], ..., M^{(i)}[31]) \\
&\mathbf{M}_j^{(i)} \leftarrow (M_j^{(i)}[0], ..., M_j^{(i)}[15])_{j=2}^{N_s} \\
&M_j^{(i)}[l] \leftarrow M_{j-1}^{(i)}[l] \boxplus M_{j-2}^{(i)}[\tau(l)]\ for\ 0 \le l \le 16
\end{aligned} \tag{4}$$

***Step* function** The *step* function is composed of MsgAdd, Mix, and WordPerm functions. The MsgAdd inputs are $CV^{(i)} = T[0], ..., T[15]$ and $M_j^{(i)} = (M_j^{(i)}[0], ..., M_j^{(i)}[15])_{j=2}^{N_s}$. The MsgAdd process is MSGADD(T,M) $\leftarrow (T[0] \oplus M[0], ..., T[15] \oplus M[15])$. The Mix function updates the 16-word $T = T[0], ..., T[15]$. In this function, the 16-word array T is split into upper eight words and lower eight words, which are then used as input. The operation of the Mix function involves modular addition, XOR, and left rotation. The process of the Mix function is shown in Figure 2. The WordPerm function is defined by $X = (X[0], ...X[15]) \leftarrow (X[\sigma(0)], ...X[\sigma(15)])$.

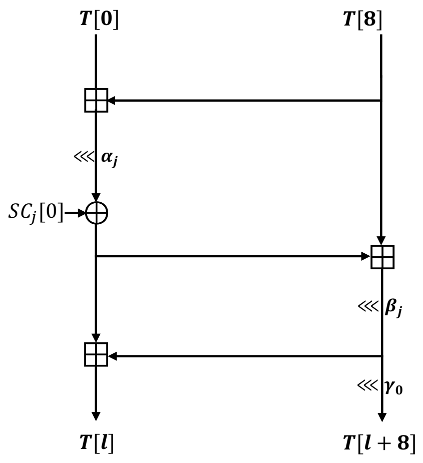

Fig. 2: Mix function

Finallization The finalization function produces an n-bit hash value, denoted as h, obtained from the final chaining variable. The finalization process is as follows:

$$\begin{aligned} &\mathbf{h} \leftarrow (CV^t[0] \oplus CV^t[8], ..., CV^t[7] \oplus CV^t[15]), \\ &h \leftarrow (h[0] \,||\, ... \,||\, h[w-1])_{[0:n-1]} \end{aligned} \tag{5}$$

3 Quantum Circuit Implementation of LSH

This section describe our quantum circuit implementation of LSH. Our main focus is to optimize the circuit depth for the efficiency of the Grover collision attack. 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.

3.1 Quantum adder for Optimizing the Depth

To implement the MsgExp function and Mix function, we use a quantum adder. Quantum adders can indeed be designed in various ways, and the choice depends on optimization techniques. Commonly used types of quantum adders include the ripple-carry adder (RCA) and the carry-lookahead (CLA) adder.

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. This sequential operation leads to a high depth of the adder, as each stage depends on the carry-out from the previous stage.

On the contrary, the CLA adder accelerates addition by pre-computing carry values for each stage. It adds extra circuits to calculate carry values in advance, determining whether a carry will occur at each stage. This pre-calculation is processed in parallel, speeding up the overall addition process and reducing the depth of the quantum circuit.

A previous work used a Cuccaro adder [4], an improved ripple-carry adder. This adder is implemented in-place operation and requires only one ancilla qubit, $(2n-3)$ Toffoli gates, $(5n-7)$ CNOT gates, and achieves a circuit depth of $(2n+2)$.

In our case, we utilize a Draper adder [5], which is a carry-lookahead adder. This adder can be implemented both in-place and out-of-place. Table 1 compares the resource estimation for 32-bit adders used in LSH-256-n application. In Table 1, the out-of-place Draper adder requires approximately twice the depth compared to the in-place adder, necessitating 32-bit output qubits for each addition. With a total of 544 adders, 17,408 (544×32) qubits are garbage qubits. Hence, we opt for the in-place Draper adder. By using Draper in-place adders, we can reuse all ancilla qubits (53 qubits) except for the input and output qubits in other operations. Although it entails a higher depth than the out-of-place adder, we employ adders in parallel within each function (described in Section 3.2 and Section 3.3). This allows us to conserve 17,408 qubits instead of allowing for a higher depth of about 400.

Table 1: 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

3.2 Parallel addition of MsgExp and Mix Functions

In the MsgExp function, 16 adders are needed to update $\mathbf{M}_j^{(i)}$. According to Section 3.1, we can initially allocate 53 ancilla qubits and reuse them throughout. However, in this scenario, 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. To process 16 adders in parallel in the MsgExp function, 848 (16×53) ancilla qubits are required.

Similarly, in the Mix function, 24 (8×3) adders are used and 8 out of the 24 adders can be operated simultaneously. In other words, 8 adders can be processed in parallel, and this parallel addition is repeated a total of 3 times. In this scenario, the ancilla qubits used in the MsgExp function can be reused. Therefore, there is no need to allocate additional ancilla qubits for the adders in the Mix function. As a result, 848 ancilla qubits are initially allocated at once.

However, due to the reuse of qubits, the depth may increase (the description continues in Section 3.3).

Table 2 shows the comparison of quantum resources required for MsgExp and Mix function. The parallel operations greatly reduce the toffoli depth and full detph compared to the sequential operations.

Table 2: 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**

※: Estimation of undecomposed resources

3.3 Combined Architecture of Compress Function

Within the Compression 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 independently. While this architecture can reduce the number of qubits, it increases the circuit depth due to the sequential operations of high complexity (as shown in Figure 3). To optimize the circuit depth, we execute the MsgExp function and Mix function in parallel by allocating additional ancilla qubits as shown in Figure 4. This parallel execution method allows us to effectively reduce the overall circuit depth, improving efficiency.

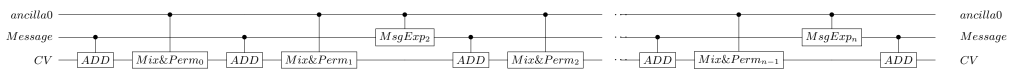

Fig. 3: Compression function in [15]

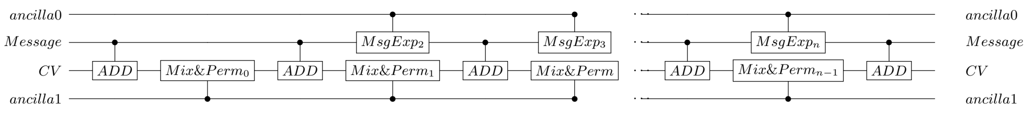

Fig. 4: Proposed Compression Function Architecture

Figure 5 shows our proposed Compression function. Specifically, the i-th Mix function in Mix function and the $i+1$-th Message Expansion 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. Thus, we initially allocate 1,272 ancilla qubits at once and reuse them each round. Algorithm 1 represents the overall process of Compress function. By allocating two sets of ancilla qubits, we can parallelize the even-round Mix function with the odd-round MsgExp function, and the odd-round Mix function with the even-round MsgExp function.

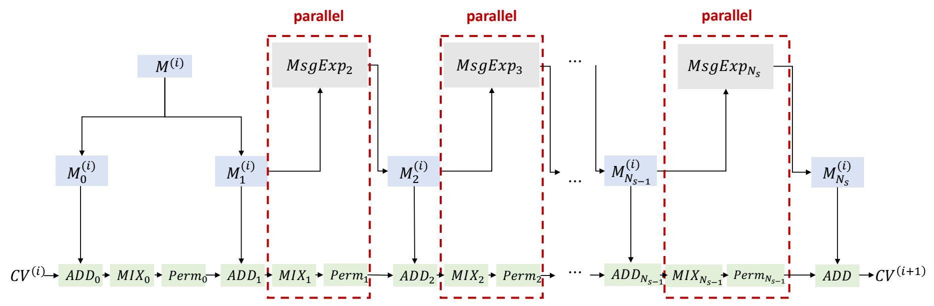

Fig. 5: Proposed Compression function Architecture

Table 3 shows the comparision of quantum resources required for Compression function. In parallel process, only the depth of the Mix functions is estimated because it has a higher depth compared to the MsgExp function. Consequently, the process of the Compression and the Mix functions in parallel demonstrates lower depth compared to processing them sequentially.

Table 3: Comparison of quantum resources required for the Compression function.

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

4 Performance & Evaluation

In this section, we provide an estimated quantum resources and the costs of Grover collision attack of our LSH quantum circuit implementation comparing the previous work. For LSH-256-n, the only differences lie in the constant value and the hash length, while the overall operation remains identical. Therefore,

Algorithm 1: Quantum circuit implementation of Compress function.

```
Input: $M_{even}$, $M_{odd}$ $CV$, $\alpha$, $\beta$, $SC$, $ancilla_0$, $ancilla_1$
Output: $M_{even}$, $M_{odd}$, $CV$, 424 qubit array-$ancilla_0$, 848 qubit array-$ancilla_1$

 1: $CV \leftarrow$ MsgAdd($M_{even}$, $CV$)
 2: $CV \leftarrow$ Mix($CV$, $\alpha_{even}$, $\beta_{even}$, $SC$, $ancilla_0$)
 3: $CV \leftarrow$ WordPerm($CV$)

 4: $CV \leftarrow$ MsgAdd($M_{odd}$, $CV$)
 5: $CV \leftarrow$ Mix($CV$, $\alpha_{odd}$, $\beta_{odd}$, $SC$, $ancilla_0$)          ▷ Parallelization 1
 6: $CV \leftarrow$ WordPerm($CV$)

 7: for $1 \leq i \leq 13$ do
 8:     $M_{even} \leftarrow$ MsgExp($M_{even}$, $M_{odd}$, $ancilla_1$)          ▷ Parallelization 1
 9:     $CV \leftarrow$ MsgAdd($M_{even}$, $CV$)
10:     $CV \leftarrow$ Mix($CV$, $\alpha_{even}$, $\beta_{even}$, $SC$, $ancilla_0$)     ▷ Parallelization 2
11:     $CV \leftarrow$ WordPerm($CV$)

12:     $M_{odd} \leftarrow$ MsgExp($M_{even}$, $M_{odd}$, $ancilla_1$)           ▷ Parallelization 2
13:     $CV \leftarrow$ MsgAdd($M_{odd}$, $CV$)
14:     $CV \leftarrow$ Mix($CV$, $\alpha_{odd}$, $\beta_{odd}$, $SC$, $ancilla_0$)       ▷ Parallelization 1
15:     $CV \leftarrow$ WordPerm($CV$)
16: end for

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

19: return $CV$
```

all estimated resources, excluding X gates, remain the identical. Similarly, the same applies to LSH-512-n. Therefore, we will only compare LSH-256-256 and LSH-512-512.

Table 4 shows the comparison of the decomposed quantum resources required for implementations of LSH. In [15], the decomposed quantum resources were not provided, so we estimate the quantum resources based on the undecomposed resources provided in the paper.

As shown in Table 4, we can observe that our implementation, which applies Cuccaro adders (the same adder as [15]) and parallelization method utilizes 8 more qubits compared to [15]. However, it reduces the full depth by approximately 12,000. Additionally, applying the Draper adder further increases the qubit usage, but it significantly reduces the full depth. To assess the trade-off between qubits and depth, we provide metrics TD-M, FD-M, TD^2-M and FD^2-M. As a result, our proposed quantum circuit achieves the optimized performance across all trade-off metrics.

Based on the estimated resources of the LSH quantum circuit, we can estimate the cost of collision attacks on LSH. To estimate the collision attack cost for LSH, we adopt the CNS algorithm described in Section 2.3. The CNS algorithm has the complexity of $\frac{2n}{5} - \frac{3s}{5}$ $(s \leq \frac{n}{4})$. According to [9], they set $s = \frac{n}{6}$ to define suitable criteria for NIST post-quantum security levels, and we follow that approach. Furthermore, since most of the quantum resources are used in implementing the target cipher in the quantum circuit, the overhead of the diffusion operator can be considered negligible compared to the oracle. Additionally, the Grover oracle consists of LSH quantum circuit twice consecutively. The first constructs the encryption circuit, and the second operates the encryption circuit in reverse to return to the state before encryption. As a result, the oracle necessitates twice the cost of implementing the quantum circuit, excluding qubits. Consequently, the cost of Grover's search for LSH is approximately $2 \times 2^{(\frac{2n}{5} - \frac{3s}{5})}$ $\times$ Table 4, as shown in Table 5. Since the cost of collision attacks varies depending on the input and output lengths, we present the resource costs for all LSH parameters.

Table 4: 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
LSH-256-256	[15]	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**	$\mathbf{1.15 \cdot 2^{22}}$	$\mathbf{1.14 \cdot 2^{25}}$	$\mathbf{1.93 \cdot 2^{32}}$	$\mathbf{1.90 \cdot 2^{38}}$
LSH-512-512	[15]	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**	$\mathbf{1.41 \cdot 2^{23}}$	$\mathbf{1.51 \cdot 2^{26}}$	$\mathbf{1.40 \cdot 2^{34}}$	$\mathbf{1.60 \cdot 2^{40}}$

Table 5: 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	$1.96 \cdot 2^{138}$	$1.91 \cdot 2^{129}$	$1.78 \cdot 2^{128}$	$1.42 \cdot 2^{76}$	$\mathbf{1.87 \cdot 2^{268}}$	$1.36 \cdot 2^{206}$	$1.27 \cdot 2^{205}$	$1.3 \cdot 2^{336}$	$1.13 \cdot 2^{334}$

5 Conclusion

In this work, we focused on optimizing the depth of quantum circuits for the Korean cryptographic hash function LSH. To optimize the depth, we use optimized quantum adders and parallelization. Our quantum circuit implementation of LSH achieves a significant improvement in depth over 78.8% compared to the approach presented in [15]. Additionally, the Toffoli depth sees an enhancement of more than 79.1%.

Through the depth-optimized implementation, we also obtain the optimized quantum resources of Grover collision attack for LSH. Although NIST provide the post-quantum security level and quantum attack costs for symmetric key ciphers, they do not provide the specific quantum cost for hash functions. If NIST defines criteria for hash functions, we will compare our results with those criteria.

References

1. Bernstein, D.J.: Cost analysis of hash collisions: Will quantum computers make sharcs obsolete. SHARCS **9**, 105 (2009) 3
2. Brassard, G., Hoyer, P., Tapp, A.: Quantum algorithm for the collision problem. arXiv preprint quant-ph/9705002 (1997) 3
3. Chailloux, A., Naya-Plasencia, M., Schrottenloher, A.: An efficient quantum collision search algorithm and implications on symmetric cryptography. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23. pp. 211–240. Springer (2017) 3
4. Cuccaro, S., Draper, T., Kutin, S., Moulton, D.: A new quantum ripple-carry addition circuit. arXiv (2008), https://arxiv.org/pdf/quant-ph/0410184.pdf 6
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) 6
6. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: Quantum resource estimates. In: Takagi, T. (ed.) Post-Quantum Cryptography. pp. 29–43. Springer International Publishing, Cham (2016) 3
7. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219 (1996) 1
8. Jang, K., Baksi, A., Kim, H., Song, G., Seo, H., Chattopadhyay, A.: Quantum analysis of AES. Cryptology ePrint Archive, Paper 2022/683 (2022), https://eprint.iacr.org/2022/683, https://eprint.iacr.org/2022/683 2, 3
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) 3, 10
10. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing Grover Oracles for quantum key search on AES and LowMC. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 280–310. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_10, https://doi.org/10.1007/978-3-030-45724-2_10 2, 3
11. Kim, D.C., Hong, D., Lee, J.K., Kim, W.H., Kwon, D.: Lsh: A new fast secure hash function family. In: Information Security and Cryptology-ICISC 2014: 17th International Conference, Seoul, South Korea, December 3-5, 2014, Revised Selected Papers 17. pp. 286–313. Springer (2015) 1
12. Liu, Q., Preneel, B., Zhao, Z., Wang, M.: Improved quantum circuits for AES: Reducing the depth and the number of qubits. Cryptology ePrint Archive, Paper 2023/1417 (2023), https://eprint.iacr.org/2023/1417, https://eprint.iacr.org/2023/1417 3

13. Rahman, M., Paul, G.: Grover on katan: Quantum resource estimation. IEEE Transactions on Quantum Engineering **3**, 1–9 (2022) 2
14. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. IEEE (1994) 1
15. 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) 7, 9, 10