Grover on *SIMON*

Ravi Anand[1], Arpita Maitra[2,3], Sourav Mukhopadhyay[1]

[1] Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur-721302, West Bengal, India
[2] TCG Centre for Research and Education in Science and Technology, Kolkata-700091, West Bengal, India.
[3]CR Rao Advanced Institute of Mathematics, Statistics and Computer Science, Hyderabad, India.

Abstract. For any symmetric key cryptosystem with n-bit secret key, the key can be recovered in $O(2^{n/2})$ exploiting Grover search algorithm, resulting in the effective key length to be half. In this direction, subsequent work has been done on AES and some other block ciphers. On the other hand, lightweight ciphers like $SIMON$ was left unexplored. In this backdrop, we present Grover's search algorithm on all the variants of $SIMON$ and enumerate the quantum resources to implement such attack in terms of NOT, CNOT and Toffoli gates. We also provide the T-depth of the circuits and the number of qubits required for the attack. We show that the number of qubits required for implementing Grover on $SIMON$ $2n/mn$ is $O(2nr + mn)$, where r is the number of chosen plaintext-cipher text pairs. We run a reduced version of $SIMON$ in IBMQ quantum simulator and the 14-qubits processor as well. We found that where simulation supports theory, the actual implementation is far from the reality due to the infidelity of the gates and short decoherence time of the qubits. The complete codes for all version of $SIMON$ have also been presented.

Keywords: Lightweight Cryptography; Quantum Cryptanalysis; Quantum Circuits; Grover's Algorithm; Feistel Ciphers

1 Introduction

The last two decades witnessed an enormous proliferation in the domain of quantum computation and communication. Due to the two pioneering quantum algorithms, Shor's algorithm [25] and Grover's search algorithm [9], the security of currently deployed cryptosystems is under a threat. As a consequence, in recent time, a lot of symmetric constructions are being evaluated in quantum settings. For example, one can mention the key recovery attacks against Even-Mansour constructions and distinguishers against 3-round Feistel constructions [19]. Not only that, the key recovery attacks against multiple encryptions [15], forgery attacks against CBC-like MACs [16] have also been studied. The list is expanding considering Quantum meet-in-the-middle attack on Feistel constructions [12], key recovery attacks against FX constructions [21] etc. Researchers have also tried to convert the existing classical attacks to quantum settings [17,13,16,23].

Very recently, Bonnetain et al. [6] proposed a novel methodology for quantum cryptanalysis on symmetric ciphers. They exploited the algebraic structure of certain classical cryptosystems to bypass the standard quantum superposition queries.

In case of symmetric ciphers or hash function, Grover's algorithm provides a quadratic speed up in exhaustive key search. So a conservative rule of thumb is to double the security parameter, i.e., atleast double the size of the key or double the size of the output of hash function. However, this does not rule out the need of analyzing the cost of Grover's algorithm on symmetric ciphers. In this direction, subsequent efforts have been made to derive cost estimation for applying Grover's search algorithm on all variants of AES [32,10,20,14]. The cost of applying Grover's search algorithm as a pre-image search attack on hash functions has also been studied [2].

Fault tolerant commercialized quantum computers are still elusive. However, several companies are providing simulation facilities through the web. Along with the simulation, IBM provides facility to run the program in their small scale actual quantum processors. Based on this state-of-the-art situation, this is very important to explore actual implementation issues of all these quantum cryptanalysis procedures.

$SIMON$ is a family of lightweight block ciphers released by NSA in June 2013. It is a balanced Feistel structured block cipher. $SIMON$ is optimized for performance in hardware implementations. In October 2018, the $SIMON$ and Speck ciphers have been standardized by ISO as a part of the RFID Air Interface Standard, International Standard ISO/29167-21 (for $SIMON$) and International Standard ISO/29167-22 (for Speck), making them available for use by commercial entities [35]

As these are comparatively new ciphers, quantum cryptanalysis on those ciphers remained unexplored. In this backdrop, in the current effort, we study the cost of Grover search on all the variants of $SIMON$ and try to implement that in publicly available IBM quantum processors. Due to the limitation of the qubits, we could not run the full scale cipher, instead we run the algorithm for a reduced version. We found that whereas simulation meets theory, actual implementation has been masked with error.

Our Contribution. One may argue that it is already well known that Grover search provides quadratic speedup over classical exhaustive key search. In this direction, we like to emphasize that for implementing Grover algorithm on a symmetric cipher, one requires a reversible implementation of that cipher which is a hard task. In this regard, we design the reversible version of all the variants of $SIMON$ [4] so that one can successfully implement Grover oracle and Grover diffusion for key search on all of those variants. We also provide the full implementation code for the cipher in QISKIT [34]. We estimate the resources in terms of NOT, CNOT, and Toffoli gates required to attack the cipher. We provide the T-depth of the circuits and the number of qubits needed to implement the attack, too. We tested our circuits with existing classical test vectors to make sure that the implementations are correct. The code for all the variants is given in [30]. This is because when the full scale quantum computers arrive, one may implement the code immediately. For independent verification of our results with the state-of-the-art IBM quantum simulator and processors, we add all the QASM and QISKIT codes for a reduced $SIMON$. To the best of our knowledge, this is the first full implementation and resource estimation on $SIMON$ in quantum settings.

2 Preliminaries

2.1 Brief Summary on $SIMON$

$SIMON$ is a family of balanced Feistel structured lightweight block ciphers with 10 different block sizes and key sizes (Table 1). The round function used in the Feistel structure of $SIMON$ block ciphers consists of circular shift, bitwise AND and bitwise XOR operations. The state update function is defined as,

$$F(x, y) = (y \oplus S^1(x)S^8(x) \oplus S^2(x) \oplus k, x) \tag{1}$$

The structure of one round $SIMON$ encryption is depicted in Figure 1, where S^j represents a left circular shift by j bits, L_i and R_i are n-bit words which constitutes the state of $SIMON$ at the i-th round and k_i is the round key which is generated by key scheduling algorithm described below.

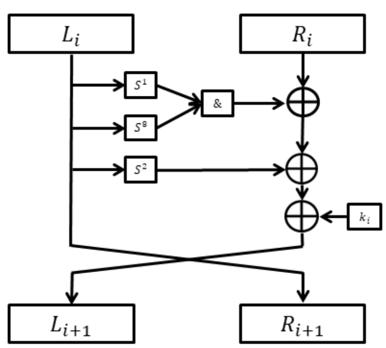


Fig. 1: $SIMON$ round function

The different variants of $SIMON$ are denoted by $SIMON$ $2n/mn$, where $2n$ denotes the block size of the variant, and mn is the size of the secret key.

Block Size ($2n$)	Key Size ($k = mn$)	word size (n)	keywords (m)	Rounds (T)
32	64	16	4	32
48	72,96	24	3,4	36,36
64	96,128	32	3,4	42,44
96	96, 144	48	2,3	52,54
128	128,192,256	64	2,3,4	68,69,72

Table 1: $SIMON$ parameters

Here n can take values from $16, 24, 32, 48$ or 64, and m from $2, 3$ or 4. For each combination of (m, n), the corresponding round number T is adopted.

The key schedule of $SIMON$ has three different procedures depending on the key size. The first m round keys are initialized directly from the main key. The remaining $(T - m)$ round keys are generated by the following procedure:

$$k_{i+m} = \begin{cases} c_i \oplus k_i \oplus S^{-3}(k_{i+1}) \oplus S^{-4}(k_{i+1}) & m = 2, \\ c_i \oplus k_i \oplus S^{-3}(k_{i+2}) \oplus S^{-4}(k_{i+2}) & m = 3, \\ c_i \oplus k_i \oplus S^{-1}(k_{i+1}) \oplus S^{-3}(k_{i+3}) \oplus S^{-4}(k_{i+3}) & m = 4, \end{cases} \tag{2}$$

where c_i are the round dependent constants and $S^{-j}(x)$ denotes right rotation by j times on x.

Encryption: The input to the encryption oracle is a $2n$-bit plaintext block P. This block is divided into $n-$bit subblocks $P = (L_0, R_0)$ which is the initial state of the cipher. The encryption consists of T applications of the round function with the respective round key produced by the key schedule. The ciphertext obtained is a $2n-$bit block $C = (L_{T-1}, R_{T-1})$

Decryption: Decryption of the ciphertext $C = (L_{T-1}, R_{T-1})$ consists of first swapping L and R part of the block cipher, i.e. the input to the decryption oracle is (R_{T-1}, L_{T-1}). Then T round functions with round keys in reverse order (ie. round keys $k_{T-1}, \cdots k_0$) is applied followed by a final swapping of the two subblocks.

Existing cryptanalysis of SIMON: To the best of our knowledge, no successful attack on full-round Simon of any variant is known. As is typical for iterated ciphers, reduced-round variants have been successfully attacked. Some of the results are summarized below Table 2. For a more detailed description of $SIMON$, the readers are referred to [4].

Variant Rounds	attacked	Time complexity	Data complexity	Attack type
Simon32/64	21/32	2^{63}	2^{31}	Integral [27]
Simon48/72	20/36	$2^{59.7}$	2^{48}	Zero Correlation [27]
Simon48/96	21/36	$2^{72.63}$	2^{48}	Zero Correlation [27]
Simon64/96	26/42	$2^{63.9}$	2^{63}	Differential [1]
Simon64/128	26/44	2^{94}	2^{63}	Differential [1]
Simon96/96	35/52	$2^{93.3}$	$2^{93.2}$	Differential [1]
Simon96/144	35/54	2^{101}	$2^{93.2}$	Differential [1]
Simon128/128	46/68	$2^{125.7}$	$2^{125.6}$	Differential [1]
Simon128/192	46/69	2^{142}	$2^{125.6}$	Differential [1]
Simon128/256	46/72	2^{206}	$2^{125.6}$	Differential [1]

Table 2: Summary of existing cryptanalysis results on $SIMON$

2.2 Grover's Algorithm

Grover's algorithm [9] searches through a space of N elements for a solution. We can assume that $N = 2^n$ and each state be represented by the indices in $\{0,1\}^n$. Let us assume that there exists a state y such that

$$f(x) = \begin{cases} 1 & \text{if x=y ,} \\ 0 & \text{otherwise.} \end{cases}$$

We also assume that f is easily and effectively computable. The *oracle*, f is provided as a black box. It finds the state y. Grover's algorithm needs only $O(2^{n/2})$ oracle calls as compared to $O(2^n)$ oracle calls needed classically.

Grover's algorithm can be summarised in the following steps:

1. Apply Hadamard gate on the initial state $|00...0\rangle$ bit by bit to obtain the following superposition
$$|\psi\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle$$
2. The second step make $\lfloor \frac{\pi}{4} 2^{n/2} \rfloor$ calls to Grover's iteration. Grover's iteration comprises of two subroutines.
The first subroutine makes use of the operator U_f which evaluates the Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, which marks the solutions of the search problem, *i.e.* $f(x) = 1$ if and only if the element corresponding to x is a solution. When we apply the Gorver oracle U_f to a state $|x\rangle|z\rangle$, where $|x\rangle$ is a n-qubit state and $|z\rangle$ is a single qubit then it acts as U_f : $|x\rangle|z\rangle \rightarrow |x\rangle|z \oplus f(x)\rangle$. If $|z\rangle$ is chosen to be $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, then we have $U_f : |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \rightarrow (-1)^{f(x)} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ which means that the oracle applies a phase shift only to the solution indices while leaving the remaining indices unaltered. Each call to U_f involves two calls to a reversible implementation of f and one call to a comparision circuit that checks if x is

a solution or not.

The second subroutine implements the transformation $2|0\rangle\langle 0| - I$, also known as the diffusion operator. This routine flips the amplitude of the states about it's mean thus amplifying the amplitude of the solution. This involves single qubit gates and one t-fold controlled NOT gate. So in the second step, the following two steps are repeated $O(2^{n/2})$ times:

(a) For any state $|x\rangle$ in the superposition $|\psi\rangle$, rotate the phase by π radians if $f(x) = 1$ and leave the system unaltered otherwise.

(b) Apply the diffusion operator.

3. Measure the resulting superposition and obtain the solution with the probabilities determined by the amplitudes of the states.

2.3 Block Cipher Key Search Using Grover

Let E be a block cipher with block size n and key size k. For any key $K \in \{0,1\}^k$, let $E_K(M)$ be the encryption of plaintext M under the key K. For a given plaintext-ciphertext pair (M, C) with $C = E_K(M)$, we can apply Grover's algorithm to determine the key K [29]. The steps involved are:

1. Define a Boolean function f for Grover's oracle which takes the key K as input,

$$f(K) = \begin{cases} 1 & \text{if } E_{K_0}(M) = C, \\ 0 & \text{otherwise.} \end{cases}$$

2. Initialize the system by making a superposition of all the possible keys with same amplitude,

$$|\mathcal{K}\rangle = \frac{1}{2^{K/2}} \sum_{j=0}^{2^K - 1} |K_j\rangle .$$

3. Iterate $2(a), (b)$ as described in Section 2.2 for $O(2^{K/2})$ times.
4. Measure the system and observe the state $K = K_0$ with probability atleast $(\frac{1}{2})$.

As a matter of fact, there may be more than one key that satisfies $C = E_{K_0}(M)$. To ensure that the key obtained is unique we may require more than one plaintext-ciphertext pairs under the same key. Let us consider that we have r such pairs (M_i, C_i). In this case the Boolean function for Grover's oracle would be defined as,

$$f(K) = \begin{cases} 1 & \text{if } E_K(M_i) = C_i, 0 \leq i \leq r, \\ 0 & \text{otherwise.} \end{cases}$$

2.4 Attack Model

We mount known plaintext attack using Grover's algorithm [9] for all the variants of $SIMON$. We consider that the adversary has access to certain pairs of plaintexts and corresponding ciphertexts. Then he finds the secret key using Grover's search using quantum resources. In this regard, we need to design a quantum circuit for all the variants of $SIMON$. In the following section, we describe the circuit.

3 Quantum Circuit for $SIMON$

In this section, we develop a reversible quantum circuit for $SIMON$. We analyze our circuits based on the number of qubits, NOT gates, $CNOT$ gates, and *Toffoli* gates. Grover search will be executed on this circuit under the known plaintext attack model, i.e. when pairs of plaintext and the corresponding ciphertext are already known.

The circuits described in this section are implemented in QISKIT [34]. The circuit is reversibly computable and needs no ancilla qubits. We also estimate the T-depth of the circuit.

The internal state size of $SIMON$ varies from 32 bits to 128 bits as described in Table 1. $SIMON$ consists of two subroutines, the round function, and the key expansion. We describe both these routines first and then show how they can be used simultaneously in the whole cipher construction.

3.1 Circuit for Round Update Function

The round function F is defined as,

$$F(x, y) = (y \oplus S^1(x)S^8(x) \oplus S^2(x) \oplus k, x),$$

where $S^i(x)$ denotes left rotation by i times on x.

Now, we assume that we have k-qubits reserved for the key, K and, n-qubits each for L and R. Let (L_0, R_0) be the initial state and the state propagate as $(L_0, R_0), (L_1, R_1), (L_2, R_2), \cdots, (L_j, R_j)$ in j rounds.

Now, due to the construction of $SIMON$, we can write

$$\begin{aligned} R_2(i) = L_1(i) \; &= R_0(i) \;\oplus\; K_0(i) \;\oplus\; L_0((i+1) mod(n/2)) \;\&\; L_0((i+8) mod(n/2)) \\ &\oplus\; L_0((i+2) mod(n/2)), 0 \leq i \leq (n/2), \end{aligned}$$

Note that here i denotes the position of the bit in L and R of a round. If we consider two round $SIMON$, then each bit of R_2 will be the XORing of each bit of R_0, $F(L_0)$ and K_0, where $F(x) = S^1(x)S^8(x) \oplus S^2(x)$. Similarly, each bit of L_2 will be the XORing of each bit of L_0, $F(R_2)$ and K_1. So the qubits reserved for R_0 can be used to store the values for R_2. Similarly the qubits reserved for L_0 can be used to store the value of L_2; hence, no need for extra qubits.

Each bit of R_2, i.e., $R_2(i)$, is computed using the following three steps:

1. $Toffoli(L_0((i+1)mod(n/2)), L_0((i+8)mod(n/2)), R_0(i))$,
2. $CNOT(L_0((i+2)mod(n/2)), R_0(i))$,
3. $CNOT(K_0(i), R_0(i))$.

L_2 can be implemented in similar way. Proceeding sequentially, we can build a circuit for as many rounds as required.

Now, it is easy to calculate that for 1 round we require n *Toffoli* gates and $2n$ $CNOT$ gates. So, for j rounds we need jn *Toffoli* gates and $2jn$ $CNOT$ gates.

Let us now define three functions $\mathcal{F}, \mathcal{G}$ and $\mathcal{H}$, shown below in Figure 2, for easy understanding of the circuit construction.

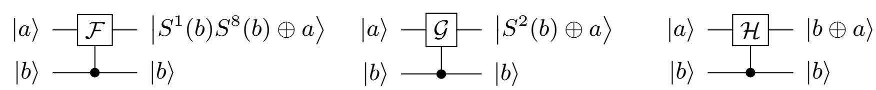


Fig. 2: Subrotines comprising the round function F

Here $|a\rangle = |a_1 a_2 ... a_n\rangle$, $|b\rangle = |b_1 b_2 ... b_n\rangle$ are n length quantum states and $(+)$ is addition modulo n, i.e. $(i+8) = (i+8)mod(n)$.

The circuit for the two rounds is shown Figure 3, where K_j, L_j, R_j represent quantum states of size n for a round j.

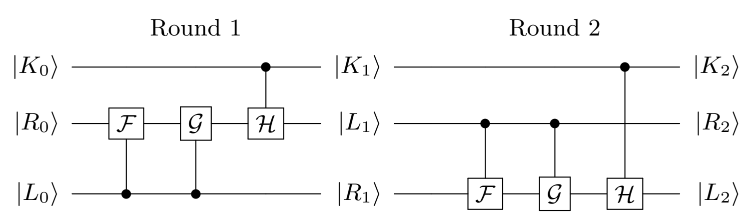


Fig. 3: Circuit for two rounds

Now, consider the circuit in Figure 4. This is the IBMQ [33] implementation of the circuit for two rounds of a reduced version of $SIMON$. The assumed state size is 16 and the key size is 16 and $m = 2$. The state is split into L, R each of size 8 and the key is split into two round keys (k_0, k_1) also of size 8. The state update function is assumed to be $F(x, y) = (y \oplus S^1(x)S^4(x) \oplus S^2(x) \oplus k, x)$. The circuit here describes the two rounds of the cipher, i.e. if we measure the L and R states, we would get the values of the state after two rounds. We can extend these circuits for more than two rounds described earlier.

One should note that this implementation works for all variants of $SIMON$ except $SIMON$ 128/192. The problem arises for the $SIMON$ 128/192 as the number of rounds is 69. This can be solved by implementing the last round such that the state R is modified according to the state update function and L state is left as it is. Then we apply a swap function on the states L and R, which increase the number of $CNOT$ gates in the circuit by an amount of $64 \times 3 = 192$.

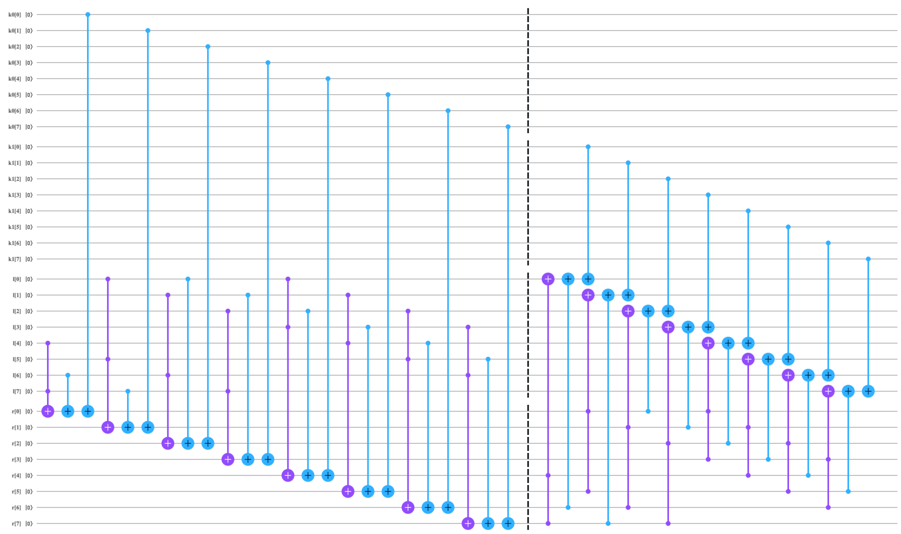


Fig. 4: Circuit for two rounds of reduced version of $SIMON$

3.2 Key Expansion

The key expansion routine is linear and invertible. It is defined in Eqn. 2. We can implement an in-place construction of key expansion without the use of any ancilla qubits. We here assume that the round constants are implemented in the circuits using an adequate number of NOT gates. This number will depend on the round constant values. Here we show the construction for all the three cases $m = 2, 3, 4$.

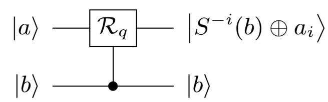
Let us define a subroutine $\mathcal{R}_q$ on two states $|a\rangle = |a_1 a_2 ... a_n\rangle$ and $|b\rangle = |b_1 b_2 ... b_n\rangle$ such that

$$\mathcal{R}_q(a, b) = (a \oplus S^{-i}(b), b)$$

That is, the state $|b\rangle$ remains unchanged and each qubit in $|a\rangle$ gets modified as $|a_j\rangle = CNOT\ (b_{(j-i)mod(n)}, a_j)$. So each application of $\mathcal{R}_q$ on $|a\rangle, |b\rangle$ involves n $CNOT$ gates, where n is the size of $|a\rangle, |b\rangle$.

For $m = 2$, we have two key words k_0, k_1 which are used for the first and second round of encryption. k_0, k_1 are states of size n, so we need $2n$ qubits to store this value.

The third round key k_2 can be computed on the same qubits which will store k_0.



Each bit $k_2(j)$ can be computed from (k_0, k_1) by applying the following three steps:

1. $CNOT(k_1(j-3)mod(n/2), k_0(j))$,
2. $CNOT(k_1(j-4)mod(n/2), k_0(j))$,
3. $NOT(k_0(j))$ only if the value of round constant rc(j) is 1.

where $0 \leq j \leq (n/2)$. The first and the second step is represented by $\mathcal{R}_3$ and $\mathcal{R}_4$ in Figure 5 respectively.
After k_2 has been computed, we can compute each bits $k_3(j)$ applying the following steps:

1. $CNOT(k_2(j-3)mod(n/2), k_1(j))$,
2. $CNOT(k_2(j-4)mod(n/2), k_1(j))$,
3. $NOT(k_1(j))$ only if the value of round constant rc(j) is 1.

where $0 \leq j \leq (n/2)$. Similarly, we can proceed to compute the further round keys sequentially.

For $m = 3$ we have three key words k_0, k_1, k_2 each of size n. The round keys for further round can be computed as explained above for $m = 2$ and the details are shown in Figure 6.

For $m = 4$ we have three key words k_0, k_1, k_2, k_3 each of size n. For the extra round keys we will require an extra step. $k_4(j)$ is computed applying the following steps:

1. $CNOT(k_1(j-1)mod(n/2), k_0(j))$,
2. $CNOT(k_3(j-3)mod(n/2), k_0(j))$,
3. $CNOT(k_3(j-4)mod(n/2), k_0(j))$,
4. $NOT(k_0(j))$ only if the value of round constant rc(j) is 1

where $0 \leq j \leq (n/2)$. The first step is the extra step required for $m = 4$. The circuit is described in Figure 7.

It can be easily calculated that for 1 round of key expansion we need mn $CNOT$ gates and n' NOT gates (where $0 \leq n' \leq n$ depending on the number of 1's in the round constant). In the complete implementation of $SIMON$, $(T-m)$ round-keys are generated as the first m key words are used as first m round keys. So, for $(T-m)$ rounds of key expansion we need $(T-m)(mn)$ $CNOT$ gates and $(T-m)n'$ NOT gates.

Consider the circuit in Figure 8. This circuit represents reduced version of key expansion of $SIMON$ with key words 2, as defined in Eqn. 2. The two round constants are assumed to be $c_0 = 11111111$ and $c_1 = 01001101$, which are implemented in the circuit by using NOT gates. Barrier separates k_3 from k_4.

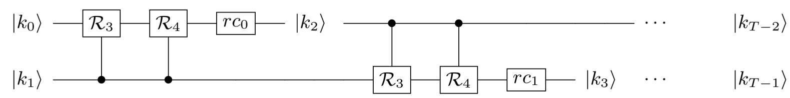


Fig. 5: Circuit for key expansion with 2 keywords. rc_i represents the round dependent constants

.

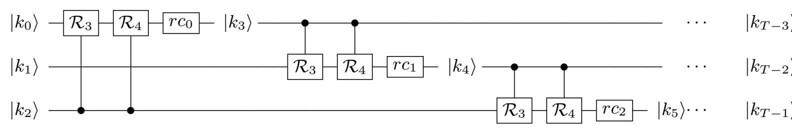


Fig. 6: Circuit for key expansion with 3 keywords. rc_i represents the round dependent constants

3.3 Circuit for $SIMON$

We implement $SIMON$ as a reversible circuit as reversibility is necessary for the cipher to be useful as a subroutine in Grover search. Using the circuits developed for round function and key expansion we can now construct the circuit for full round $SIMON$.

The input to the circuit is the key K and the plaintext split into two halves L_0, R_0. The output of the circuit is the ciphertext L_{T-1}, R_{T-1}, where T is the number of rounds. The size of K, L, R are mn, n, n respectively, where m is either 2 or 3 or 4. In Figure 9, we draw the circuit considering $m = 2$. Similar construction can be made for variants with $m = 3$ and $m = 4$. $\mathcal{U}$, is the round update function which consists of the three subroutines $\mathcal{F}, \mathcal{G}, \mathcal{H}$ and $\mathcal{KE}$ is the key expansion routine described in 3.2.

Figure 10 gives an implementation of the reduced version of $SIMON$, with two key words. First, we run the circuit in IBMQ Simulator [33]. We consider the key size and state size to be 6 and the number of rounds to be 4. The state update function is defined as $((L_{j+1}, R_{j+1}) = (R_j \oplus (S^1(L_j) \& S^2(L_j)) \oplus S^0(L_j) \oplus k_j), L_j)$ and the key expansion is defined as $k_{j+2} = c_j \oplus k_j \oplus S^{-1}(k_{j+1}) \oplus S^{-2}(k_{j+1})$ where c_j are round dependent constants $[0, 0, 1]$ and $[0, 0, 1]$ for the third and fourth round respectively. Let the plaintext be $L_0 = [0, 1, 1], R_0 = [1, 0, 1]$ and the key words be $k_0 = [0, 0, 1], k_1 = [1, 1, 0]$. Then after four rounds the ciphertext will be $L_4 = [0, 1, 1], R_4 = [1, 1, 1]$. In IBMQ circuit, one should read the cipher text as $L_4 = [L_4(0), L_4(1), L_4(2)]$ and $R_4 = [R_4(0), R_4(1), R_4(2)]$. In Figure 11 the output is shown as $[R_4(2), R_4(1), R_4(0), L_4(2), L_4(1), L_4(0)]$.

Then, we run the circuit (Figure 10) in ibmq_melbourne, the 14 qubits actual processor. In contrary to the simulation (Figure 11), we observed a huge error in the result. Figure 12 shows the histogram we have obtained after running the circuit in ibmq_melbourne for 1024 shots. The reason behind this is the infidelity of the gates used in the circuit and the decoherence time of the qubits used in the actual processor, which has been pointed out in several works like [11,26] and also in the IBMQ's official site.

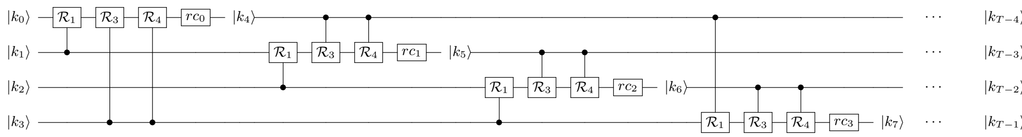


Fig. 7: Circuit for key expansion with 4 keywords. rc_i represents the round dependent constants

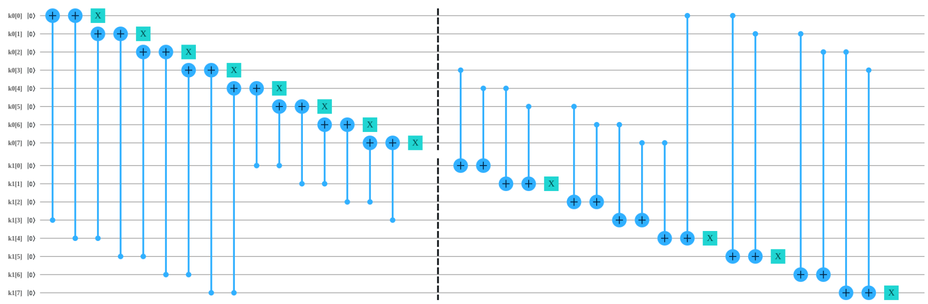


Fig. 8: Key expansion for $m = 2$ and key size 16 split into two round keys of size 8 each.

3.4 Grover's oracle

In this subsection, we will discuss the implementation of Grover search on a block cipher under known plain text attack. Let r pairs of plaintext and ciphertext be sufficient to successfully extract the unique solution. In this regard, we have to design an oracle that encrypts the given r plaintexts under the same key and then computes a Boolean value which determines if all the resulting ciphertexts are equal to the given classical ciphertexts. This can be done by running the block cipher circuit r times in parallel. Then the resultant ciphertexts are compared with the classical ciphertexts. The target qubit will be flipped if the ciphertexts match. This is called Grover oracle. In Figure 13, the construction of such an oracle is given for two instances of plaintext-ciphertext pairs considering $SIMON$ block cipher.

We implement the above idea, i.e., Grover oracle for our reduced version of $SIMON$ (Figure 10) in the IBMQ simulator. We have not run the circuit for Grover in ibmq_melbourne, as the output of the reduced $SIMON$ in this processor has already been masked with a huge amount of error (Figure 12).

Now, in this case, the plaintext is $M = [0, 1, 1, 1, 0, 1]$ and the key is $K = [0, 0, 1, 1, 1, 0]$. Then after four rounds the ciphertext will be $C = [0, 1, 1, 1, 1, 1]$. In theory, this key $K = [0, 0, 1, 1, 1, 0]$ will be obtained as the output of Grover oracle. Figure 14a shows the outcome of Grover applied to this circuit. Note that in the histogram two peaks appear; one for the exact key K and another for an unknown key $K' = [1, 1, 1, 0, 0, 0]$. This is because, the unknown key K' also

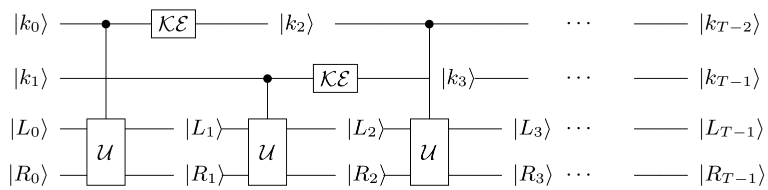


Fig. 9: The circuit for implementing $SIMON$ with two key words. .

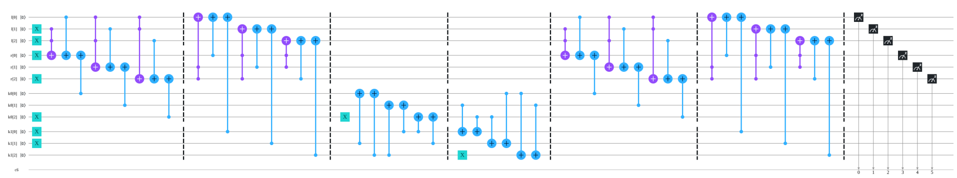


Fig. 10: Circuit for 4 rounds reduced $SIMON$ with two key words

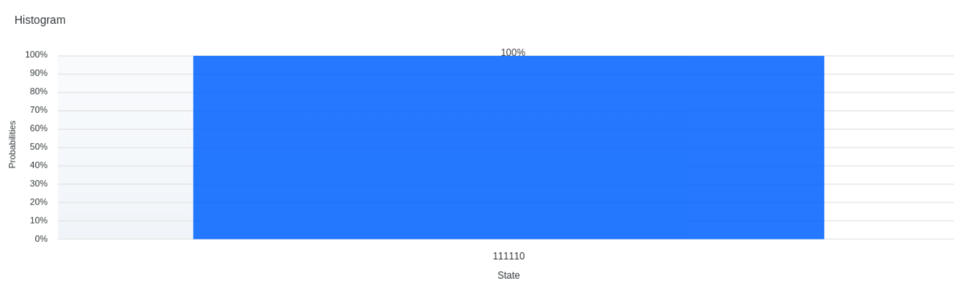


Fig. 11: Measurement of circuit in Figure 10 in IBMQ Simulator. As expected the output after 4 rounds is [011111].

encrypts the plain text M onto the cipher text C. To find the unique key, we use another plaintext-ciphertext pair, $M_1 = [0,0,1,1,0,1], C_1 = [1,1,0,0,1,1]$ under the same key. Figure 14b shows the outcome of Grovers applied to this circuit. In this histogram also we got two peaks. Similar to 14a, one peak is for the exact key K and another is for an unknown key $K'' = [0,0,1,0,0,1]$ as K'' encrypts M_1 to C_1 too. Note that the key common to both the pairs is $K = [0,0,1,1,1,0]$ which is the exact key. The full implementation is described in [30]. The functions Grover_oracle and Grover_Diffusion used in our code is taken from [18] by importing the file Our_Qiskit_Functions.

3.5 Resource Estimation

Cost of implementing $SIMON$ We first estimate the cost of implementing $SIMON$ as a circuit including the key expansion as well as the round function. As discussed above the round constants are implemented to the key expansion function using an adequate number of NOT gates. We have not included the

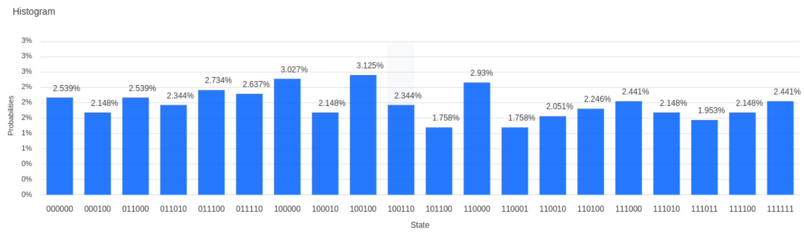


Fig. 12: Measurement of circuit in Figure 10 in ibmq_melbourne, the 14 qubits actual processor.

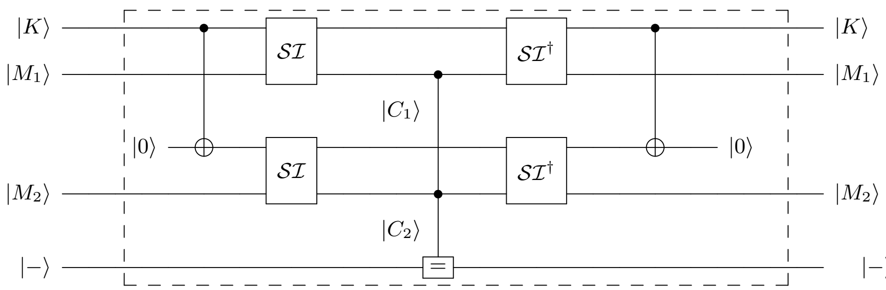
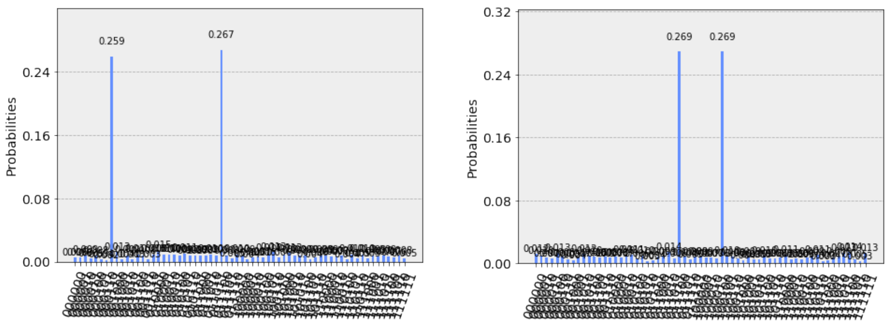


Fig. 13: Grover's oracle for $SIMON$ using two plaintext-ciphertext pairs. Here, $\mathcal{SI}$ represents the $SIMON$ encryption as described in 3.3. The $(=)$ operator compares the output of the $\mathcal{SI}$ with the given ciphertexts and flips the target qubit if they are equal.

number of NOT required to initialize the plaintext in our estimates as it depends on the given plaintext. Table 3 gives the cost estimates of implementing all $SIMON$ variants.

Cost of Grover oracle Following [14] we assume that $r = \lceil k/(2n) \rceil$ known plaintext-ciphertext pairs are sufficient to give us a unique solution, where $2n$ is the block size and $k = mn$ is the key size of the cipher. Here, one should mention that if $r = \lceil k/(2n) \rceil$ is an integer, then consider its next integer. Grover's oracle consists of comparing the $2n$-bit outputs of the $SIMON$ instances with the given r ciphertexts. This can be done using a $(2n \cdot r)$-controlled $CNOT$ gates (we neglect some NOT gates which depend on the given ciphertexts). Following [28], we estimate the number of T gates required to implement a t-fold controlled NOT gates as $(32 \cdot t - 84)$.

We use the decomposition of *Toffoli* gates to 7 T-gates plus 8 Clifford gates, a T-depth of 4 and total depth of 8 as in [3]. To estimate the full depth and the T-depth we only consider the depths of the $SIMON$ instances ignoring the multi controlled NOT gate used in comparing the ciphertexts. We also need $(2 \cdot (r-1) \cdot k)$ $CNOT$ gates to make the input key available to all the $SIMON$



(a) **(b)**

Fig. 14: Histogram obtained after running Grover's on the reduced $SIMON$ described in Figure 10.

$SIMON$ $2n/mn$	# NOT	# $CNOT$	# *Toffoli*	# qubits	depth
$SIMON$ 32/64	448	2816	512	96	946
$SIMON$ 48/72	792	3312	864	120	1062
$SIMON$ 48/96	768	4800	864	144	1597
$SIMON$ 64/96	1248	5184	1344	160	1674
$SIMON$ 64/128	1216	7396	1408	192	2643
$SIMON$ 96/96	2400	9792	2496	192	4785
$SIMON$ 96/144	2448	10080	2592	240	3282
$SIMON$ 128/128	4224	17152	4352	256	8427
$SIMON$ 128/192	4224	17472	4416	320	5656
$SIMON$ 128/256	4352	26624	4608	384	8848

Table 3: Cost of implementing $SIMON$ variants

instances in the oracle. The total number of Clifford gates is the sum of the Clifford gates used in the implementation of $SIMON$ and the $(2 \cdot (r-1) \cdot k)$ $CNOT$ gates needed for input key. The cost estimates for all $SIMON$ variants are presented in Table 4

Cost of exhaustive key search Using the estimates in Table 4 of Grovers oracle for the various variants, we provide the cost estimates for the full exhaustive key search on all variants in Table 5. We consider $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ iterations of Grovers operator. As in [20] we do not consider the depth of implementing the two multi-controlled NOT gates while calculating the T-depth and overall depth.

$SIMON$ $2n/k$	r	# Clifford gates	# T gates	T-depth	full depth	# qubits
$SIMON$ 32/64	3	19840	24492	12288	27180	161
$SIMON$ 48/72	2	16560	27180	13824	28440	169
$SIMON$ 48/96	3	33792	40812	20736	45860	241
$SIMON$ 64/96	2	25620	41644	21504	44988	224
$SIMON$ 64/128	3	52184	65196	33792	74994	321
$SIMON$ 96/96	2	48768	75948	39936	89028	289
$SIMON$ 96/144	2	50400	78636	41472	86104	337
$SIMON$ 128/128	2	85760	129964	69632	151564	385
$SIMON$ 128/192	2	87168	131756	70656	146272	449
$SIMON$ 128/256	3	186880	205740	110592	246624	641

Table 4: Cost of Grover oracle

$SIMON$ $2n/k$	# Clifford gates	# T gates	T-depth	Full depth	# qubits
$SIMON$ 32/64	$1.35 \cdot 2^{45.5}$	$1.27 \cdot 2^{46}$	$1.18 \cdot 2^{45}$	$1.05 \cdot 2^{46.3}$	161
$SIMON$ 48/72	$1.01 \cdot 2^{49.65}$	$1.03 \cdot 2^{50.45}$	$1.01 \cdot 2^{49.4}$	$1.05 \cdot 2^{50.37}$	169
$SIMON$ 48/96	$1.02 \cdot 2^{62.66}$	$1.02 \cdot 2^{63.05}$	$1.01 \cdot 2^{61.97}$	$1.02 \cdot 2^{63.11}$	241
$SIMON$ 64/96	$1.02 \cdot 2^{62.27}$	$1.01 \cdot 2^{63.08}$	$1.10 \cdot 2^{61.9}$	$1.07 \cdot 2^{63}$	224
$SIMON$ 64/128	$1.03 \cdot 2^{79.27}$	$1.02 \cdot 2^{79.7}$	$1.06 \cdot 2^{78.6}$	$1.03 \cdot 2^{79.8}$	321
$SIMON$ 96/96	$1.02 \cdot 2^{63.2}$	$1.04 \cdot 2^{63.85}$	$1.02 \cdot 2^{62.9}$	$1.02 \cdot 2^{64}$	289
$SIMON$ 96/144	$1.05 \cdot 2^{87.2}$	$1.06 \cdot 2^{87.9}$	$1.22 \cdot 2^{86.7}$	$1.03 \cdot 2^{88}$	337
$SIMON$ 128/128	$1.03 \cdot 2^{80}$	$1.14 \cdot 2^{80.5}$	$1.17 \cdot 2^{79.51}$	$1.12 \cdot 2^{80.7}$	385
$SIMON$ 128/192	$1.04 \cdot 2^{112}$	$1.17 \cdot 2^{112.5}$	$1.19 \cdot 2^{111.51}$	$1.08 \cdot 2^{112.7}$	449
$SIMON$ 128/256	$1.05 \cdot 2^{145.1}$	$1.11 \cdot 2^{145.2}$	$1.07 \cdot 2^{144.3}$	$1.12 \cdot 2^{145.4}$	641

Table 5: Cost estimates of Grovers algorithm with $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ oracle iterations with a success probability negligibly close to 1.

4 Conclusion

In this work we presented an implementation of Grovers search algorithm on $SIMON$. We first provided a reversible circuit of all variants of $SIMON$. Then these circuits were used to estimate the cost of attacking $SIMON$ using Grover's algorithm. The plausible values the overall circuit depth suggested by NIST [31] is between 2^{40} and 2^{96} and we assume that this depth is an upper bound on the total depth of a quantum attack. The overall circuit depth of all the implementations presented in this work except for SIMON 128/192 and SIMON 128/256 lie in this range. This work is aimed at using the minimal number of qubits, in future it would be interesting to re-estimate the cost by reducing the depth of the implementations at the expense of some extra qubits by using other choices of decomposition of *Toffoli* gates, e.g., as in [24] which has a T-depth of 1 and an overall depth of 7 but uses 18 Clifford gates and 4 ancilla qubits.

References

1. Abed F., List E., Lucks S., Wenzel J. (2015) Differential Cryptanalysis of Round-Reduced Simon and Speck. In: Cid C., Rechberger C. (eds) Fast Software Encryp-

tion. FSE 2014. Lecture Notes in Computer Science, vol 8540. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-46706-0_27
2. Amy, M., Di Matteo, O., Gheorghiu, V., Mosca, M., Parent, A. and Schanck, J., 2016, August. Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In International Conference on Selected Areas in Cryptography (pp. 317-337). Springer, Cham.
3. Amy, M., Maslov, D., Mosca, M. and Roetteler, M., 2013. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 32(6), pp.818-830.
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B. and Wingers, L., 2015, June. The SIMON and SPECK lightweight block ciphers. In Proceedings of the 52nd Annual Design Automation Conference (pp. 1-6).
5. Biryukov, A., Leurent, G. and Perrin, L., 2015, August. Cryptanalysis of Feistel networks with secret round functions. In International Conference on Selected Areas in Cryptography (pp. 102-121). Springer, Cham.
6. Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y. and Schrottenloher, A., 2019, December. Quantum Attacks without Superposition Queries: the Offline Simon's Algorithm. In International Conference on the Theory and Application of Cryptology and Information Security (pp. 552-583). Springer, Cham.
7. Boyer, M., Brassard, G., Hyer, P. and Tapp, A., 1998. Tight bounds on quantum searching. Fortschritte der Physik: Progress of Physics, 46(4?5), pp.493-505.
8. Brassard, G., Hoyer, P., Mosca, M. and Tapp, A., 2002. Quantum amplitude amplification and estimation. Contemporary Mathematics, 305, pp.53-74.
9. Grover, L.K., 1996, July. A fast quantum mechanical algorithm for database search. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (pp. 212-219).
10. Grassl, M., Langenberg, B., Roetteler, M. and Steinwandt, R., 2016, February. Applying Grover's algorithm to AES: quantum resource estimates. In Post-Quantum Cryptography (pp. 29-43). Springer, Cham.
11. Harper, R. and Flammia, S.T., 2019. Fault-tolerant logical gates in the ibm quantum experience. Physical review letters, 122(8), p.080504.
12. Hosoyamada, A. and Sasaki, Y., 2018, September. Quantum Demiric-Selcuk meet-in-the-middle attacks: applications to 6-round generic Feistel constructions. In International Conference on Security and Cryptography for Networks (pp. 386-403). Springer, Cham.
13. Hosoyamada, A. and Sasaki, Y., 2018, April. Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In Cryptographer'sTrack at the RSA Conference (pp. 198-218). Springer, Cham.
14. Jaques, S., Naehrig, M., Roetteler, M. and Virdia, F., 2020, May. Implementing Grover oracles for quantum key search on AES and LowMC. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 280-310). Springer, Cham.
15. Kaplan, M., 2014. Quantum attacks against iterated block ciphers. arXiv preprint arXiv:1410.1434.
16. Kaplan, M., Leurent, G., Leverrier, A. and Naya-Plasencia, M., 2016, August. Breaking symmetric cryptosystems using quantum period finding. In Annual International Cryptology Conference (pp. 207-237). Springer, Berlin, Heidelberg.
17. Kaplan, M., Leurent, G., Leverrier, A. and Naya-Plasencia, M., 2015. Quantum differential and linear cryptanalysis. IACR Transactions on Symmetric Cryptology (2016): 71-94.

18. Koch, D., Wessing, L. and Alsing, P.M., 2019. Introduction to Coding Quantum Algorithms: A Tutorial Series Using Qiskit. arXiv preprint arXiv:1903.04359.

19. Kuwakado, H. and Morii, M., 2012, October. Security on the quantum-type Even-Mansour cipher. In 2012 International Symposium on Information Theory and its Applications (pp. 312-316). IEEE.
20. Langenberg, B., Pham, H. and Steinwandt, R., 2019. Reducing the cost of implementing AES as a quantum circuit. Cryptology ePrint Archive, Report 2019/854.
21. Leander, G. and May, A., 2017, December. Grover meets Simon–quantum attacking the FX-construction. In International Conference on the Theory and Application of Cryptology and Information Security (pp. 161-178). Springer, Cham.
22. Rivest, R.L., Shamir, A. and Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), pp.120-126.
23. Santoli, T. and Schaffner, C., 2016. Using Simon's algorithm to attack symmetric-key cryptographic primitives. arXiv preprint arXiv:1603.07856.
24. Selinger, P., 2013. Quantum circuits of T-depth one. Physical Review A, 87(4), p.042302.
25. Shor, P.W., 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review, 41(2), pp.303-332.
26. Tannu, S.S. and Qureshi, M.K., 2019, April. Not all qubits are created equal: a case for variability-aware policies for NISQ-era quantum computers. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (pp. 987-999).
27. Wang, Q., Liu, Z., Varici, K., Sasaki, Y., Rijmen, V. and Todo, Y., 2014, December. Cryptanalysis of reduced-round SIMON32 and SIMON48. In International Conference on Cryptology in India (pp. 143-160). Springer, Cham.
28. Wiebe, N. and Roetteler, M., 2014. Quantum arithmetic and numerical analysis using Repeat-Until-Success circuits. arXiv preprint arXiv:1406.2040.
29. Yamamura, A. and Ishizuka, H., 2000. Quantum cryptanalysis of block ciphers. Algebraic systems, formal languages and computations. RIMS Kokyuroku 1166 : 235-243.
30. https://github.com/raviro/quantsimon
31. https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf
32. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
33. https://quantum-computing.ibm.com/
34. https://qiskit.org/
35. https://en.wikipedia.org/wiki/$SIMON$_(cipher)