Quantum Information Processing (2021) 20:394
https://doi.org/10.1007/s11128-021-03322-7

Evaluation of Grover's algorithm toward quantum cryptanalysis on ChaCha

Bhagwan Bathe[1] · Ravi Anand[2] · Suman Dutta[3]

Received: 25 April 2021 / Accepted: 21 October 2021 / Published online: 19 November 2021

Abstract
In this work, we have analyzed ChaCha against Grover's search algorithm. We designed a reversible quantum circuit of ChaCha and then estimated the resources required to implement Grover. We showed that for MAXDEPTH = 2^{40}, the ChaCha20 256-bit key can be recovered using Grover's search algorithm with a gate count of $1.233 \cdot 2^{251}$, which is less than the NIST's requirement of 2^{258}. We also showed that implementing Grover's algorithm greatly depends on the number of rounds in ChaCha. We deduced that ChaCha would require approximately 166 rounds so that implementing a non parallelized Grover would require a 2^{298} gate count. We implemented a ChaCha-like toy cipher in IBMQ simulator and recovered key using Grover's algorithm.

Keywords Quantum algorithm · Grover's algorithm · ChaCha · IBMQ simulator

1 Introduction

The symmetric encryption schemes can be categorized as the schemes which are good for hardware implementations and those which are good for software implementation. ChaCha is an encryption scheme which is good for software implementation. It is a general consensus that implementing a software-based cipher in hardware requires

✉ Ravi Anand
ravianandsps@gmail.com

Bhagwan Bathe
bathebn@barc.gov.in

Suman Dutta
sumand.iiserb@gmail.com

[1] Bhabha Atomic Research Centre (CI), Homi Bhabha National Institute, Mumbai 400094, India

[2] Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur 721302, India

[3] Department of Computer Science, Indian Statistical Institute Kolkata, Kolkata 700108, India

more amount of resources than a hardware-based cipher. However, if we want to study the security of a software-based cipher against quantum algorithms, we have to estimate the cost of implementing the cipher in quantum computers.

ChaCha [7] was designed by Bernstein in 2008 as a refinement of the stream cipher Salsa20. Salsa is one of the selected ciphers in the eSTREAM software portfolio. Compared to Salsa, in ChaCha the diffusion per round has been increased to achieve a high performance during software implementations. The inexpensive operations and its relatively fewer resource requirements made it easier to implement on different architectures. ChaCha also has a better privacy as compared to Salsa in terms of side-channel attack. It has a simple and fast key setup procedure, which made ChaCha widely acceptable in the community.

Recent attacks have shown several problems with CBC-mode cipher suites in TLS and DTLS as well as issues with the stream cipher RC4. Because of several attacks at the security conferences, Google stopped using ciphers like SSLv3 and RC4. The stream cipher ChaCha started getting renewed interest when Google decided to implement ChaCha20 for symmetric encryption and Poly1305 for authentication (MAC) in OpenSSL and NSS in March 2013.

Since its inception, several works have been done in the direction of classical cryptanalysis of ChaCha. In FSE 2008, Aumasson et al. [5] first used probabilistic neutral bits (PNBs) technique to break ChaCha6 and ChaCha7. In 2013, Shi et al. [19] improved the result by considering a new type of distinguishers, called column chaining distinguishers (CCD). In WCC 2015, Maitra et al. [16] provided interpretation based on Fisher's 2006 result by including the key bits in the PNB set which results lesser probability for distinguishing. Cryptanalysis on reduced round of ChaCha by exploiting specific chosen IVs corresponding to the secret key was done by Maitra [15] in 2016. Further work of Choudhuri et al. [9] improved the result by considering hybrid model for the evaluation of the differential cryptanalysis of ChaCha and claimed that ChaCha12 with 256-bit keys is secure against certain kinds of differential cryptanalysis. Improved algorithm, for the construction of PNBs and modified complexities for the cryptanalysis of ChaCha were given by Dey et al. [11]. All these studies clearly suggest that higher rounds of ChaCha (for example ChaCha20) are absolutely secure against the classical attacks. Although there have been a lot of studies regarding the classical cryptanalysis of ChaCha, surprisingly not enough work has been done in the direction of quantum cryptanalysis of the same.

We would also like to bring attention to the work done by Aumasson [Subsection 5.2, [6]] where the author has stated: "Grover's search (and other quantum cryptanalysis techniques) is mostly independent of the number of rounds ..." and has discussed how some of the ciphers, including ChaCha has a very large margin of security in even while considering quantum attacks. So, in our work, we completely appreciate the point raised by the author in [6] and accept that quantum computers are still elusive, and at the same time, there are several papers [1–3,10,12,14,17,18] where the resources and gate counts are understood based on Grover's algorithm as well as the metrics provided by NIST [21]. Further, a cipher with a lower amount of hardware will be considered as more vulnerable in the Quantum paradigm because the quantum computers are in the evolving stage, and thus technology will move toward building more efficient quantum computers over the next few decades, and ciphers with less amount of hardware will attract sharper cryptanalysis in limited resource if we use Grover directly.

So, there is a requirement to study ciphers under the recommendation provided by NIST and based on these observations we find that ChaCha requires 166 rounds to be secure against the recommendations provided by NIST, (Subsection 2.4). We also show that with MAXDEPTH=2^{40}, ChaCha20 key can be recovered using the Grover's search algorithm with a gate count of $1.233 \cdot 2^{251}$, which less than the NIST's requirement of 2^{258}. This paper could serve as a starting point for estimating gate counts for the quantum implementations of software-based stream ciphers like ChaCha.

Although an error-free, commercial quantum computer is elusive at the present time frame, we have implemented a toy version of the circuit in the IBMQ simulator and also recovered the secret key using Grover's algorithm on this toy version.

1.1 Brief description of ChaCha stream cipher

ChaCha maps a 512-bit input block to a 512-bit output key stream. The state is built by arranging data in a 4×4 matrix M where each element is a 32-bit word. M is initialized as:

$$M = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ cnt & iv_0 & iv_1 & iv_2 \end{pmatrix}. \tag{1}$$

The initial state M consists of 128-bit predefined constants c_0, c_1, c_2, c_3, 32-bit block counter cnt, 96-bit nonce iv_0, iv_1, iv_2 and 256-bit key $k_0, k_1, \cdots k_7$. In case the key length is 128-bit the key is copied in both second and third rows, i.e., $(k_0, k_1, k_2, k_3) = (k_4, k_5, k_6, k_7)$.

The core function is defined by iterating several rounds on the input blocks, where each round consists of four parallel quarterround operations. Each of quarterround(a, b, c, d) consists of four ARX rounds, each of which comprises of addition modulo 2^n ($\boxplus$), cyclic left rotation and XOR ($\oplus$) operation (one each) as given below:

$$\begin{cases} a = a \boxplus b; \quad d = d \oplus a; \quad d = d \lll 16; \\ c = c \boxplus d; \quad b = b \oplus c; \quad b = b \lll 12; \\ a = a \boxplus b; \quad d = d \oplus a; \quad d = d \lll 8; \\ c = c \boxplus d; \quad b = b \oplus c; \quad b = b \lll 7; \end{cases} \tag{2}$$

Each *columnround* consists of four *quarterrounds* on each of the four columns of the state matrix and each *diagonalround* consists of four *quarterrounds* on each of the four diagonals. In ChaCha$R$, $R/2$ *columnround*, and $R/2$ *diagonalround* are applied alternatively to the initial state (total R many rounds, where R could be 20, 12).

In each of the odd rounds, we first apply quarterround to all the four columns in the following order:

$$\begin{cases} quarterround(x_0, x_4, x_8, x_{12}); \\ quarterround(x_1, x_5, x_9, x_{13}); \\ quarterround(x_2, x_6, x_{10}, x_{14}); \\ quarterround(x_3, x_7, x_{11}, x_{15}); \end{cases}$$

This is a complete *columnround*. In each of the even rounds, we apply quarterround to all the four diagonals in the following order:

$$\begin{cases} quarterround(x_0, x_5, x_{10}, x_{15}); \\ quarterround(x_1, x_6, x_{11}, x_{12}); \\ quarterround(x_2, x_7, x_8, x_{13}); \\ quarterround(x_3, x_4, x_9, x_{14}); \end{cases}$$

This describes a complete *diagonalround*.

Let the state after R rounds be M^R. A keystream block of 16 words or 512 bits is obtained as $Z = M \boxplus M^R$.

1.2 Stream cipher key recovery using Grover

Let for any key, $K = \{0, 1\}^k$ and initial value, $IV = \{0, 1\}^{96}$, $\mathcal{S}_{K,IV} = ks$, be the 512-bit keystream generated by the ChaCha stream cipher $\mathcal{S}$. For a known keystream ks generated by a secret key K_0 we can apply Grover's search for key recovery as follows:

1. Construct a Boolean function f which takes K, IV as input and satisfies

$$f(K) = \begin{cases} 1 & \text{if } \mathcal{S}_{K,IV} = ks \\ 0 & \text{otherwise} \end{cases}$$

2. Initialize the system by making a superposition of all possible keys of equal amplitudes.

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

   (a) For any state $|K_j\rangle$ in the superposition $|\mathcal{K}\rangle$, rotate the phase by π radians if $f(K_j) = 1$ and leave the system unaltered otherwise.
   (b) Apply the diffusion operator.

3. Iterate $2(a), (b)$ for $O(2^{k/2})$ times.

4. Measure the system and observe the state $K = K_0$ with probability atleast $(\frac{1}{2})$, where K_0 is the secret key.

In most cases, it is considered that the IV is known to the adversary so the first step can be modified so that the function f takes only K as input and satisfies $f(K) = 1$, if $\mathcal{S}_K$ generates the given keystream ks.

From [8], we know that when measuring the k key qubits after iterating the Grover's oracle for $\iota \geq 0$ iterations, then the success probability $p(\iota)$ of obtaining the correct key K_0 is given by $p(\iota) = \sin^2((2\iota + 1)\theta)$, where $\sin^2(\theta) = 1/2^k$. Since $1/2^k$ is negligibly small, we have $\theta \approx \sin(\theta) = \sqrt{1/2^k}$. The probability $p(\iota)$ is close to 1 if $\iota \approx \frac{\pi}{4\theta}$. Hence after approximately $\frac{\pi}{4}2^{k/2}$ iterations, we obtain the correct key with a very high probability.

1.3 Resource estimation under a depth limit

In this work, we are focused on estimating the cost of implementing Grover's search algorithm on ChaCha under NIST's MAXDEPTH. Grover's full algorithm parallelizes very badly. So, we use the inner parallelization as described by Kim, Han, and Jeong [13]. In inner parallelization, the seacrh space is divided into disjoint subsets and each subset is assigned to a different machine. Since each machine's search space is smaller, the required number of iterations is smaller.

The original search space has size 2^k, where k is the key size. Let us assume that we use a Grover's oracle $\mathcal{O}$ such that a single Grover's iteration costs $\mathcal{O}_g$ gates and has a depth $\mathcal{O}_d$. Also, assume that $M = 2^m$ be the number of machines that are used in parallel by dividing the search space into M disjoint sets.

Since the search space is now reduced to 2^{k-m} for each machine, hence we can expect that one of the machines will recover the correct key after approximately $\iota_m = \frac{\pi}{4}2^{\frac{k-m}{2}}$ iterations, with a very high success probability. Then, the total depth of ι_m Grover iterations will be

$$D_m = \iota_m \times \mathcal{O}_d \approx \frac{\pi}{4}2^{\frac{k-m}{2}}\mathcal{O}_d \tag{3}$$

Each of the M machines will be using $\iota_m \times \mathcal{O}_g$ gates for ι_m iterations and thus the total gate cost over all M machines will be

$$G_m = 2^m \iota_m \times \mathcal{O}_g \approx \frac{\pi}{4}2^{\frac{k+m}{2}}\mathcal{O}_g \tag{4}$$

Now let us fix the depth limit to MAXDEPTH. Then from Eq. 3, we will have

$$\begin{aligned} \mathsf{MAXDEPTH} &= \frac{\pi}{4}2^{\frac{k-m}{2}}\mathcal{O}_d \\ \implies M = 2^m &= \left(\frac{\pi}{4}\right)^2 2^k \frac{\mathcal{O}_d^2}{\mathsf{MAXDEPTH}^2} \end{aligned} \tag{5}$$

Using this value in Eq. 4, we obtain the total gate cost under the MAXDEPTH restriction as

$$G_{MD} = \left(\frac{\pi}{4}\right)^2 2^k \frac{\mathcal{O}_g \mathcal{O}_d}{\mathsf{M}AXDEPTH} \tag{6}$$

We will use these values to compute the cost of implementing Grover's search on ChaCha under NISTs MAXDEPTH limit in Sect. 2.3.4.

Contribution and Organization The main contributions of this work are as follows.

- In Sect. 2, we provide the reversible quantum circuits of ChaCha. We estimate the cost of applying Grover for key recovery in Table 3 and then estimate it under the NISTs MAXDEPTH constraint in Table 4. We find that an attack using the Grover's key search algorithm has a gate count complexity $1.233 \cdot 2^{251}$ for 20 rounds ChaCha and $1.859 \cdot 2^{249}$ for 12 rounds ChaCha, with MAXDEPTH = 2^{40}. As an immediate observation, we find the relation between the number of rounds and the gate count estimation for applying Grover's algorithm.
- In Sect. 3, we implement the Grover's search algorithm on a toy version of ChaCha in the IBMQ environment and successfully recovered the desired key. The necessary circuit representations are also given using Qiskit.

In Sect. 4, we conclude the paper.

All the codes for the results on the toy cipher are provided in a public repository [23] for independent verification of our results.

2 Resource estimation of ChaCha

We construct a reversible quantum circuit for implementation of the cipher. We then provide the resource estimation for this construction in terms of number of qubits, Toffoli gates, CNOT gates and the depth of the circuit in Table 1.

We have assumed full parallelism while constructing the circuits, i.e., any num ber of gates can be applied in the circuit simultaneously if these gates act on dis joint set of qubits. The circuits were implemented in IBMQ simulator and this compiler allows us to compute the depth of the circuit automatically. It has been generally observed that the depth of two circuits applied in series is less than or equal to the sum of the depth of the individual circuits.

2.1 Quantum implementation of XOR, rotation, and addition modulo 2^n

The ARX ciphers makes use of the following operations

- bitwise XOR, $\oplus$
- left and right circular shifts (rotations), S^j and S^{-j}, respectively, by j bits, and
- addition modulo 2^n, $\boxplus$.

Component-wise XOR of two n-bit numbers a and b can be implemented using n many *CNOT* gates in a quantum circuits i.e. $\{b = a \oplus b\} = CNOT\{a_i, b_i\}$ for $i \in \{0, 1, \ldots, n-1\}$.

Rotation of any number can be implemented by swapping values across wires which can be done by simply keeping track of the necessary free rewiring. So, in the estimations, these gates are disregarded as free.

For addition we use the circuit described in [20] as it uses no ancilla. To implement $a = a \boxplus b$ where $a = a_0a_1 \ldots a_{n-1}$ and $b = b_0b_1 \ldots b_{n-1}$ are two n-bit numbers, the circuit construction is described in Algorithm 1.

Algorithm 1 Implementing addition modulo 2^n [20]

```
1: procedure $\mathcal{ADD}(a, b)$:  $a = a \boxplus b$
2:     for $i \leftarrow 1, n-1$ do
3:         $CNOT\ \{b_i, a_i\}$
4:     end for
5:     for $i \leftarrow n-1, 1$ do
6:         $CNOT\ \{b_i, b_{i+1}\}$
7:     end for
8:     for $i \leftarrow 0, n-1$ do
9:         $Toffoli\ \{a_i, b_i, b_{i+1}\}$
10:    end for
11:    for $i \leftarrow n-1, 1$ do
12:        $CNOT\ \{b_i, a_i\}$
13:        $Toffoli\ \{a_{i-1}, b_{i-1}, b_i\}$
14:    end for
15:    for $i \leftarrow 1, n-2$ do
16:        $CNOT\ \{b_i, b_{i+1}\}$
17:    end for
18:    for $i \leftarrow 0, n-1$ do
19:        $CNOT\ \{b_i, a_i\}$
20:    end for
21: end procedure
```

To implement addition modulo 2^n, we only need to remove one *CNOT* gate and one *Toffoli* gate associated with a_n.

Proposition 1 [20] *Addition modulo* 2^n *requires* $5n - 6$ *CNOT gates,* $2n - 2$ *Toffoli gates.*

The circuit for addition modulo 2^n, with $n = 3$ and $a = 001, b = 010$ has been implemented in IBMQ, as shown in Fig. 1.

2.2 Quantum circuit for the ChaCha

We now describe the construction of a reversible quantum circuit for ChaCha with key size 256 and rounds $R = [12, 20]$. As described above, ChaCha has two major components: the quarterround routine and the last step in which the initial state is added to the state we get after the R round of updates. We have given the circuit for full ChaCha for R rounds.

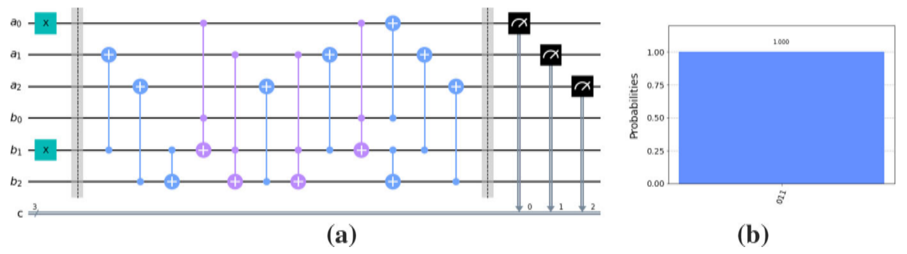


Fig. 1 **a** Circuit for implementing $a = a \boxplus b$. **b** Result obtained after measurement when inputs were $a = 001, b = 010$

2.2.1 Circuit for *quarterround*

The quarterround(a, b, c, d) routine is defined as

$$\begin{aligned} a &= a \boxplus b; & d &= d \oplus a; & d &= d \lll 16; \\ c &= c \boxplus d; & b &= b \oplus c; & b &= b \lll 12; \\ a &= a \boxplus b; & d &= d \oplus a; & d &= d \lll 8; \\ c &= c \boxplus d; & b &= b \oplus c; & b &= b \lll 7; \end{aligned}$$

where a, b, c, d are n-bit numbers. So we need $4n$ qubits to store the values of a, b, c, d. The circuit can be constructed as described in Algorithm 2 and in Fig. 2.

Algorithm 2 Implementing quarterround

```
1: procedure QUARTERROUND($a, b, c, d$)
2:     $\mathcal{ADD}(a, b)$                    ▷ $\mathcal{ADD}$ as desribed in Algorithm 1
3:     for $i \leftarrow 0, n-1$ do
4:         $CNOT\ \{a_i, d_i\}$
5:     end for
6:     $\mathcal{RL}_{16}(d)$                    ▷ $\mathcal{RL}_n(x) \rightarrow$ left rotate $x$ by $n$ bits
7:     $\mathcal{ADD}(c, d)$
8:     for $i \leftarrow 0, n-1$ do
9:         $CNOT\ \{c_i, b_i\}$
10:    end for
11:    $\mathcal{RL}_{12}(b)$
12:    $\mathcal{ADD}(a, b)$
13:    for $i \leftarrow 0, n-1$ do
14:        $CNOT\ \{a_i, d_i\}$
15:    end for
16:    $\mathcal{RL}_8(d)$
17:    $\mathcal{ADD}(c, d)$
18:    for $i \leftarrow 0, n-1$ do
19:        $CNOT\ \{c_i, b_i\}$
20:    end for
21:    $\mathcal{RL}_7(b)$
22: end procedure
```

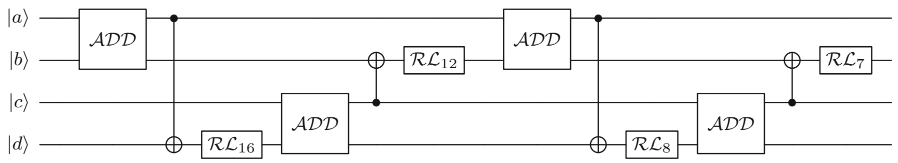


Fig. 2 Circuit for quarterround(a, b, c, d), where $|a\rangle$, $|b\rangle$, $|c\rangle$, $|d\rangle$ are quantum states of size n. $\mathcal{ADD}$ is the adder circuit described in Algorithm 1. $\mathcal{RL}_n$ stands for rotate-left by n bits

Proposition 2 *The circuit for one quarterround routine requires* $4 * (5 * 32 - 6) + 4 * 32 = 744$ *CNOT gates,* $4*(2*32-2) = 248$ *Toffoli gates. One round of ChaCha has 4 quarterround routines, and so requires* $4*744 = 2976$ *CNOT gates and* $4*248 = 992$ *Toffoli gates.*

2.2.2 Circuit for the last step

The last step of ChaCha is to add the initial state with the final state obtained after R rounds. For this, we need 16×32 qubits which stores a copy of the initial state, which can be done using $CNOT$ gates,

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix} = \begin{pmatrix} c_0 \oplus x'_0 & c_1 \oplus x'_1 & c_2 \oplus x'_2 & c_3 \oplus x'_3 \\ k_0 \oplus x'_4 & k_1 \oplus x'_5 & k_2 \oplus x'_6 & k_3 \oplus x'_7 \\ k_4 \oplus x'_8 & k_5 \oplus x'_9 & k_6 \oplus x'_{10} & k_7 \oplus x'_{11} \\ cnt \oplus x'_{12} & iv_0 \oplus x'_{13} & iv_1 \oplus x'_{14} & iv_2 \oplus x'_{15} \end{pmatrix}. \tag{7}$$

The values of $x_i's$ in the right matrix of Eq. 7 are initialized to zero and so the matrix on the left is a copy of the initial state. This matrix is added to the final state matrix to generate the keystream.

Proposition 3 *The last step requires* 16 $\mathcal{ADD}$ *operations, and so the number of CNOT gates and Toffoli gates required for this step, are* 2464 *and* 992 *respectively.*

2.2.3 Circuit for full ChaCha

To implement ChaCha we need 16 quantum states of size 32, and another 16 quantum states of size 32 to store the copy of initial state for the last step. The circuit for full ChaCha is described in Fig. 3.

Proposition 4 *The complete circuit for ChaCha R requires* $[(R \times 2976) + 2464]$ *CNOT gates and* $[(R \times 992) + 992]$ *Toffoli gates.*

2.3 Resource estimation

2.3.1 Cost of implementing ChaCha

We first estimate the cost implementing ChaCha as a circuit. We have not included the number of NOT gates required to initialize the states in our estimates as it depends on

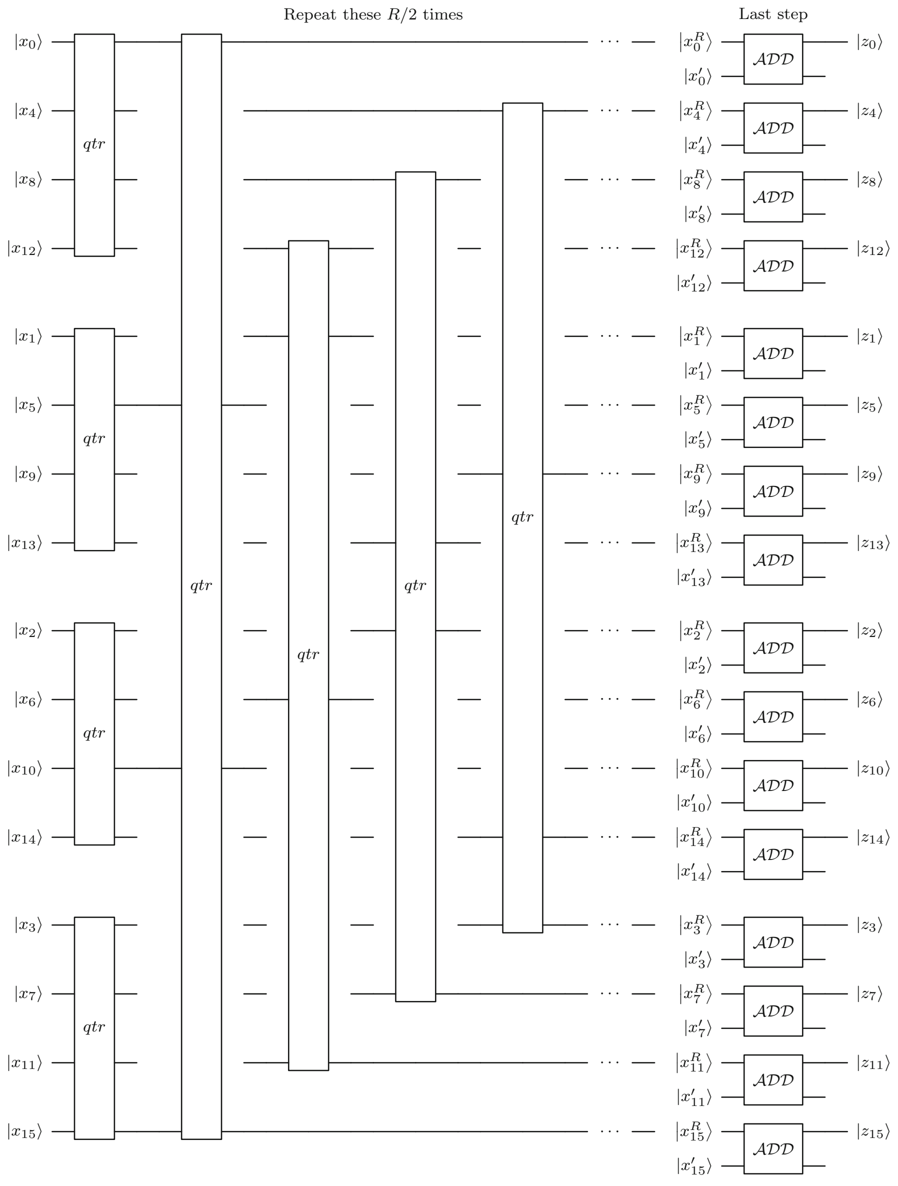


Fig. 3 Circuit for ChaCha, where $R = [20, 12]$

the given constant and nonce. Table 1 gives the cost estimates of implementing both the ChaCha variants.

Table 1 Cost of implementing ChaCha variants

	# *CNOT*	# *Toffoli*	# qubits	Full depth
ChaCha 12	38176	12896	1024	7643
ChaCha 20	61984	20832	1024	12635

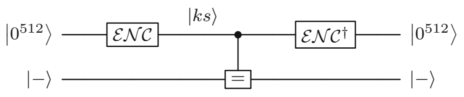


Fig. 4 Grover oracle for stream ciphers. The (=) operator compares the output of the $\mathcal{ENC}$ with the given keystreams and flips the target qubit if they are equal

2.3.2 Cost of Grover's oracle

The Grover's oracle consists of producing a superposition of keystreams that can be generated from all possible keys and given constants and IV. Then the keystream is matched to the given keystream and the target qubit is flipped if the keystream matches. So, this oracle consists of the circuit of the cipher that produces the keystream and then this circuit needs to be uncomputed so that the next iteration can be run. We denote the cipher instance as $\mathcal{ENC}$. Uncomputation of $\mathcal{ENC}$ is denoted by $\mathcal{ENC}^\dagger$. The oracle is constructed as shown in Fig. 4

The oracle in Fig. 4 is different from the oracle generally constructed for block ciphers, as described in [12, Figure 6]. However, we can show that the key retrieved from this oracle which uses only one instance of the cipher, instead of multiple cipher instances used for block ciphers, is unique.

Uniqueness of the Key Retrieved: Let us first discuss the scenario for a block cipher. We assume a block cipher that has been initialized with a key K of size k and has a block size of k bits to be a pseudo random permutation (PRP) $C_K : \{0,1\}^k \rightarrow \{0,1\}^k$ that takes a message M of size k bit and outputs a cipher text of the same size. Then if we generate t-blocks of cipher text corresponding to a message, then we have the following collision probability

$$\underset{K\neq K'}{Pr}\Big(C_K(M_1)||C_K(M_2)\ldots||C_K(M_t)$$
$$= C_{K'}(M_1)||C_{K'}(M_2)\ldots||C_{K'}(M_t)\Big) \approx 2^k - 1\prod_{i=1}^{t}\frac{1}{2^k - i - 1}.$$

Even if we set $t = 2$ then we have a negligibly low probability of collision $\mathcal{O}\left(2^{-k}\right)$.

Now let us consider the case of stream ciphers. Suppose a stream cipher has a key of size k bits. Then we can design the following function $\hat{C}_\rho(K) : \{0,1\}^k \rightarrow \{0,1\}^\rho$ which takes in the key K and outputs ρ bits of keystream. Then we can safely assume

$$\underset{K\neq K'}{Pr}\left(\hat{C}_\rho(K) = \hat{C}_\rho(K')\right) \approx \frac{2^k - 1}{2^\rho}.$$

Even if we set $\rho = k + c$ for some constant c, the collision probability is approximately $\mathcal{O}\left(2^{-c}\right)$. Then even for $c = k$ we have a very low probability of collision and thus less false positive. Hence we can say that the oracle described in Fig. 4 will retrieve the required key uniquely.

We now describe the process of computing the cost of the Grover's oracle. It has generally been assumed that the logical T gate is significantly more expensive than the Clifford group gates on the surface code and so while estimating the resources we provide the costs for T gates only as well as the cost of circuit considering all gates equally. While computing the T-depth of a circuit, we assume that all gates have a depth of 0 except the T gate, which is assigned a depth of 1 and for the full depth we assign a depth of 1 to all the gates.

We decompose the Toffoli gates into the set of Clifford+T gates using the decomposition provided by [4] that requires 7 T gates and 8 Clifford gates, a T depth of 4 and total depth 8.

We estimate the cost of constructing the Grover's oracle in terms of number of Clifford and T gates, T depth and the full depth (D). The number of Clifford gates required for the oracle of stream ciphers is the number of the Clifford gates required for the cipher used in the oracle. The total number of Clifford gates is computed as

$$2 \cdot \text{ \# Clifford gates for } \{(\mathcal{ENC})\}. \tag{8}$$

The Grover's oracle consists of comparing the 512-bit outputs of the cipher with the given 512-bit keystreams. This can be done using (512)-controlled $CNOT$ gates (we neglect some NOT gates which depend on the given cipher-texts). Following [22], we estimate the number of T gates required to implement a t-fold controlled $CNOT$ gates as $(32 \cdot t - 84)$. The total number of T gates is computed as

$$(32 \cdot 512 - 84) + 2 \cdot \text{ \# } T \text{ gates for } \{(\mathcal{ENC})\}. \tag{9}$$

To estimate the full depth and the T-depth we only consider the depths of the cipher instances. The depth of the Grover's oracle is

$$\text{The depth of the oracle} = 2 \cdot (\text{Depth of } \mathcal{ENC}). \tag{10}$$

Using Eqs. 8, 9, 10 the cost estimates for the Grover's oracle are presented in Table 2.

Table 2 Cost of the Grover's oracle for ChaCha

	# Clifford gates	# T gates	Gate Cost($\mathcal{O}_g$)	T-depth	Full depth($\mathcal{O}_d$)	# qubits
ChaCha 12	282688	188972	471660	23808	54878	1025
ChaCha 20	457280	300076	757356	39680	90718	1025

Table 3 Cost estimates of Grover's algorithm with $\frac{\pi}{4}2^{k/2}$ oracle iterations for ChaCha

k	Cipher	# Clifford gates	# T gates	G	T-depth	Full depth (D)	# qubits
256	ChaCha 12	$1.694 \cdot 2^{145}$	$1.132 \cdot 2^{145}$	$1.413 \cdot 2^{146}$	$1.141 \cdot 2^{142}$	$1.315 \cdot 2^{143}$	1025
	ChaCha 20	$1.370 \cdot 2^{146}$	$1.798 \cdot 2^{145}$	$1.135 \cdot 2^{147}$	$1.902 \cdot 2^{142}$	$1.087 \cdot 2^{144}$	1025
128	ChaCha 12	$1.694 \cdot 2^{81}$	$1.108 \cdot 2^{81}$	$1.401 \cdot 2^{82}$	$1.141 \cdot 2^{78}$	$1.315 \cdot 2^{79}$	1025
	ChaCha 20	$1.370 \cdot 2^{82}$	$1.774 \cdot 2^{81}$	$1.128 \cdot 2^{83}$	$1.902 \cdot 2^{78}$	$1.087 \cdot 2^{80}$	1025

G is the addition of # Clifford and # T gates

2.3.3 Cost of exhaustive key search

Using the estimates in Table 2, of the Grover's oracle, we provide the cost estimates for the full exhaustive key search on ChaCha in Table 3. We consider $\frac{\pi}{4}2^{k/2}$ iterations of the Grover's operator. We have not included the cost of putting all key bits in superposition using Hadamard gates.

2.3.4 Cost of Grover's search under NISTs MAXDEPTH limit

In this work we assume that an adversary is bounded by a constraint on the depth of the circuit that (s)he can use for Grover. NIST suggests a parameter MAXDEPTH as such a bound and the plausible values range from 2^{40} to 2^{96}.

From [21]

1. In, Page 16, it is stated: "In particular, NIST will define a separate category for each of the following security requirements (listed in order of increasing strength): 1) Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128-bit key (e.g., AES128)."
2. In Page 18, it is stated: "NIST provides the estimates for optimal key recovery for AES256 as 2^{298}/MAXDEPTH."

So, for a MAXDEPTH of 2^{40}, a cipher can be considered VULNERABLE if there exists any attack with gate count $< 2^{258}$ for ciphers with key size 256.

The cost estimates provided by NIST is deduced from the results in [17]. In Table 4, we provide the gate counts for Grover's search on both the ciphers under the constraint of MAXDEPTH. These counts are computed using Eq. 6.

Table 4 Cost of Grover's search on the ciphers with key size 256 under MAXDEPTH

	G_{MD} for MAXDEPTH			$G_{MD} *$ MAXDEPTH
	2^{40}	2^{64}	2^{96}	
NIST[21]	2^{258}	2^{234}	2^{202}	2^{298}
ChaCha 12	$1.859 \cdot 2^{249}$	$1.859 \cdot 2^{225}$	$1.859 \cdot 2^{193}$	$1.859 \cdot 2^{289}$
ChaCha 20	$1.233 \cdot 2^{251}$	$1.233 \cdot 2^{227}$	$1.233 \cdot 2^{195}$	$1.233 \cdot 2^{291}$
AES[12]	$1.39 \cdot 2^{245}$	$1.39 \cdot 2^{221}$	$1.39 \cdot 2^{190}$	$\approx 2^{285}$

2.4 Are the number of rounds enough?

From above descriptions we can calculate that one round of state update requires the following resources: 10912 Clifford gates, 6944 T gates and has depth 2240. So, for R rounds the circuit will have $R \cdot (10912 + 6944)$ number of gates and depth $(R \cdot 2240)$. The last step of adding the initial state to the state obtained after R rounds requires $(10400 + 6944)$ gates and depth 559.

It should be noted here that we do not assume full parallelism and have not considered some extra overheads which might be required while running the full Grover's iterations and also have ignored multiplication by $\frac{\pi}{4}$. These assumptions will, however, amount to only a difference of constant amount.

Let us consider that 2^x is the security parameter which is the minimum amount of gate counts required to mount a quantum attack to call a cipher secure. Then the following should be satisfied:

$$\begin{aligned}
&2^x \leq G \cdot D \\
&\Rightarrow 2^x \leq (2^{128} \cdot 2 \cdot (R \cdot (10912 + 6944) + (10400 + 6944))) \\
&\quad \cdot (2^{128} \cdot 2 \cdot (R \cdot 2240 + 559)) \\
&\Rightarrow 2^x \leq 2^{258} \cdot (R \cdot 17856 + 17344)(R \cdot 2240 + 559) \\
&\Rightarrow 2^{x-258} \leq (R \cdot 17856 + 17344)(R \cdot 2240 + 559).
\end{aligned}$$

Now we can make the following observations:

1. The NIST's requirement is $x = 298$. For $R = 166$, we have $(R \cdot 17856 + 17344)(R \cdot 2240 + 559) \approx 1.009 \cdot 2^{40}$, which means the number of rounds ChaCha must have, to be secure, is atleast 166.
2. From [17], we have $x = 296$, and for $R = 83$, we have $(R \cdot 17856 + 17344)(R \cdot 2240 + 559) \approx 1.017 \cdot 2^{38}$, which means the number of rounds ChaCha must have, to be secure, is 83.
3. From [14], we have $x = 291$, and for $R = 15$, we have $(R \cdot 17856 + 17344)(R \cdot 2240 + 559) \approx 1.134 \cdot 2^{33}$, which means the number of rounds ChaCha must have, to be secure, is 15.
4. If we consider the most recent resource count of implementing Grover's algorithm on AES described in [12], then $x = 285$. For $R = 2$, we have $(R \cdot 17856 + 17344)(R \cdot 2240 + 559) \approx 1.992 \cdot 2^{27}$, which means the number of rounds ChaCha must have, to be secure, is 2.

We can see here that depending on the security parameters, the round numbers varies from 2 to 166.

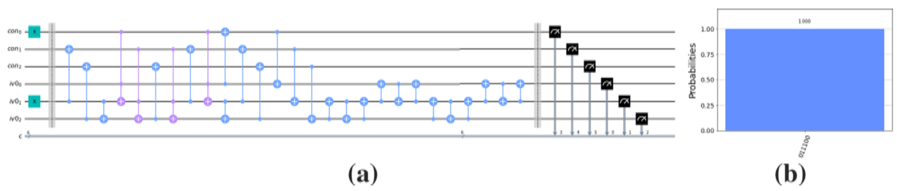


Fig. 5 **a** Circuit for implementing quarterround$(con, iv0)$. **b** Measurement for inputs, $con = 001, iv0 = 010$. Read the output (011100) as $(con, iv0)$

3 Implementing Grover's search on a ChaCha like toy ciper

In this section, we implement a ChaCha like toy cipher in the IBMQ interface. We show that the above mentioned construction of the cipher can be used to construct a toy cipher and successfully extract the key using the Grover's algorithm. To find the secret key, we have assumed that we are given a keystream and the IV.

3.1 Toy ChaCha like cipher

Now we give an implementation of a toy ChaCha like cipher in Qiskit.

Assume, the initial state is

$$M = \begin{pmatrix} con & iv1 \\ iv0 & key \end{pmatrix} = \begin{pmatrix} 001 & 100 \\ 010 & 111 \end{pmatrix}. \tag{11}$$

The quarterround(a, b) is defined as

$$quarterround(a, b) : a = a \boxplus b; b = b \oplus a; b \lll 2.$$

The circuit for quarterround$(con, iv0)$ as implemented in IBMQ is shown in Fig. 5. Then we apply a column round and a diagonal round as follows:

$$\begin{cases} quarterround(con, iv0); \\ quarterround(iv1, key); \\ quarterround(key, con); \\ quarterround(iv0, iv1); \end{cases} \tag{12}$$

and we obtain the state

$$M^f = \begin{pmatrix} con^f & iv1^f \\ iv0^f & key^f \end{pmatrix} = \begin{pmatrix} 011 & 010 \\ 111 & 101 \end{pmatrix}. \tag{13}$$

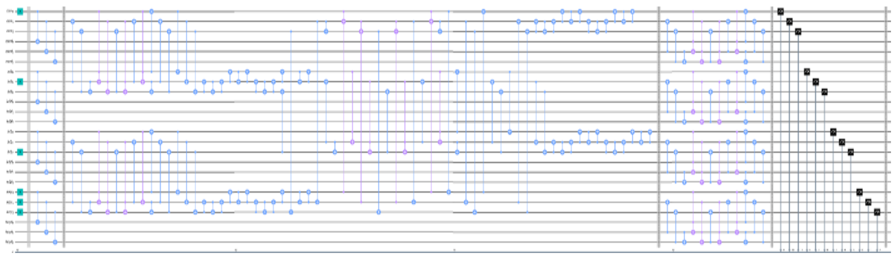
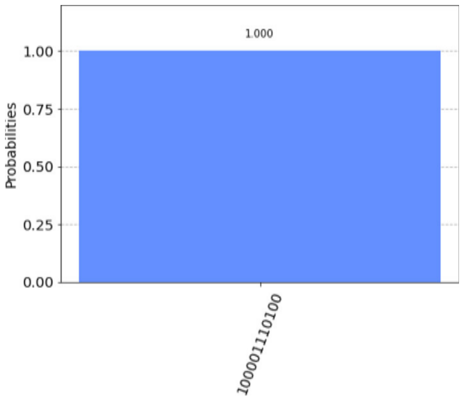


Fig. 6 The circuit of toy ChaCha with the initial state defined in Eq. 11

Fig. 7 The output of the circuit implemented in Fig. 6. Read the output (100001110100) as $(con, iv0, iv1, key)$



And finally the block of key stream is obtained as $Z = M \boxplus M^f$. For the initial state as defined in Eq. 11, the key stream obtained is:

$$Z = \begin{pmatrix} 100 & 110 \\ 001 & 100 \end{pmatrix}. \tag{14}$$

Figure 6 gives an implementation of the reduced ChaCha described above in IBMQ.
Figure 7 shows the output.

3.2 Grover's search on toy ChaCha

For ChaCha the initial state is $M = \begin{pmatrix} 001 & 100 \\ 010 & 111 \end{pmatrix}$, where the key is 111 and $Z = \begin{pmatrix} 100 & 110 \\ 001 & 100 \end{pmatrix}$ is the keystream obtained. Applying Grover should return the key, $key = 111$. Figure 8 shows the outcome of applying Grover on (M, Z).

3.3 Implementation in NISQ device (ibmq_16_melbourne quantum computer)

We implement the above described toy cipher in the real quantum computer, ibmq_16_melbourne, provided by IBMQ. Since the maximum number of qubits avail-

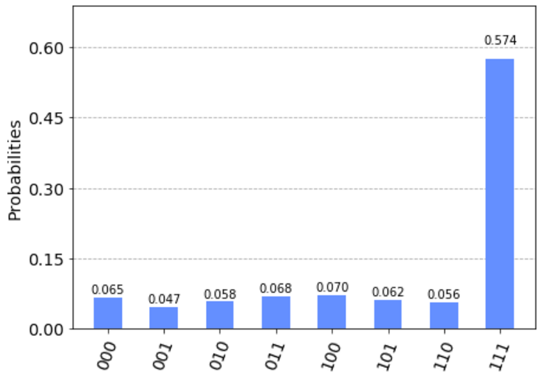


Fig. 8 Histogram obtained after running Grover's search on the reduced ChaCha described in Fig. 6

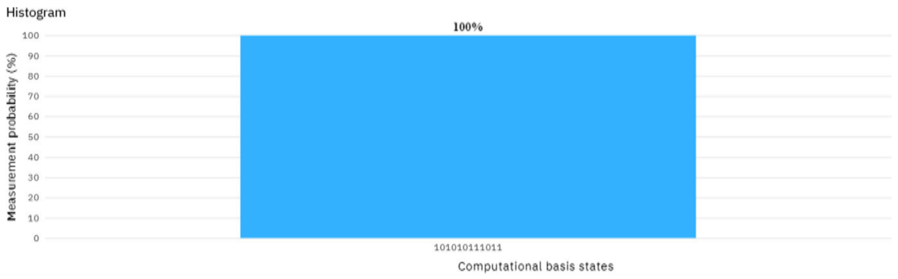


Fig. 9 Histogram obtained by running the toy cipher in the simulator

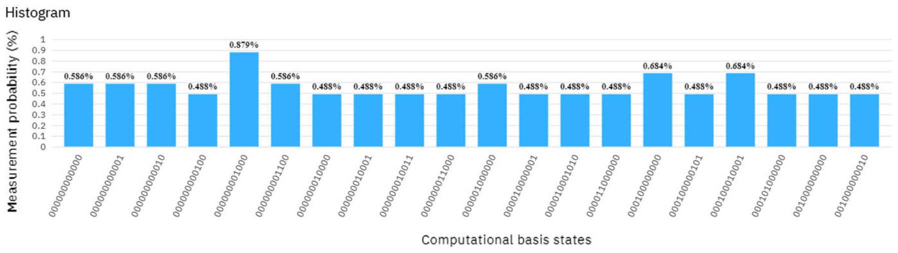


Fig. 10 Histogram obtained by running the toy cipher in ibmq_16_melbourne quantum computer

able is 15, so we do not implement the last step, i.e., addition of the last round, which requires 12 more qubits. The comparison between the results obtained in the simulator and actual machine shows how a real threat from quantum computers is still distant dream.

Figure 9 shows the output of implementing the toy cipher in the simulator for 1000 shots.

Figure 10 shows the output of implementing the toy cipher in the real quantum computer for 1000 shots. We can see here that the output of the real quantum computer is completely random and the actual output is far from the reality due to the infidelity of the gates and short decoherence time of the qubits.

The situation in Fig. 10 shows that the threat due to a real quantum computer is still a distant dream. However, it is worth analyzing ciphers considering that the future with better working quantum computers is not very far.

4 Conclusion

In this work, we have constructed quantum reversible circuits for ChaCha and then estimated the resources required to implement the Grover's search algorithm to find the key. We have studied the cost of implementing Grover's key search algorithm on the cipher under the NISTs MAXDEPTH constraint and find that the ciphers fail to satisfy the security constraints. We have also observed that the number of rounds does affect the effectiveness of the Grover's algorithm. Our estimates are not optimal as we have made some assumptions and we firmly believe that these estimates can be made tighter under different set of assumptions. Also future works on various quantum error correcting schemes may provide a significantly optimized results in reality.

Funding Not Applicable.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Anand, R., Maitra, A., Mukhopadhyay, S.: Grover on SIMON. Quantum Inf. Process. **19**(9), 1–17 (2020). https://doi.org/10.1007/s11128-020-02844-w
2. Anand, R., Maitra, A., Mukhopadhyay, S.: Evaluation of quantum cryptanalysis on SPECK. In: Bhargavan, K., Oswald, E., Prabhakaran, M. (eds.) Progress in Cryptology INDOCRYPT 2020. LNCS, vol. 12578. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-65277-7_18
3. Anand, R., Maitra, S., Maitra, A., Mukherjee, C.S., Mukhopadhyay, S.: Resource estimation of Grovers-kind quantum cryptanalysis against FSR based symmetric ciphers. Cryptology ePrint Archive, Report 2020/1438, https://eprint.iacr.org/2020/1438 (2020)
4. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(6), 818–830 (2013). https://doi.org/10.1109/TCAD.2013.2244643
5. Aumasson, J.P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New features of Latin dances: analysis of Salsa, ChaCha, and Rumba. In: Nyberg, K. (ed.) Fast Software Encryption. FSE 2008. Lecture Notes in Computer Science, vol. 5086. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-71039-4_30
6. Aumasson, J.P.: Too much crypto. IACR Cryptol. ePrint Arch., p.1492 (2019). https://eprint.iacr.org/2019/1492
7. Bernstein, D.J.: ChaCha, a variant of Salsa20. In: Workshop Record of SASC, vol. 8, pp. 3–5 (2008)
8. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik: Progress of Physics **46**(4–5), 493–505 (1998)
9. Choudhuri, A., Maitra, S.: Differential cryptanalysis of salsa and ChaCha: An evaluation with a hybrid model. Cryptology ePrint Archive: Report 2016/377, (2016)
10. Davenport, J.H., Pring, B.: Improvements to quantum search techniques for block-ciphers, with applications to AES. In: Selected Areas in Cryptography-SAC (2020)
11. Dey, S., Sarkar, S.: Improved analysis for reduced round Salsa and Chacha. Discrete Appl. Math. **227**, 58–69 (2017). https://doi.org/10.1016/j.dam.2017.04.034

12. Jaques, S., Naehrig, M., Roetteler, M. and Virdia, F.: Implementing Grover oracles for quantum key search on AES and LowMC. Advances in Cryptology—EUROCRYPT 2020, volume 12106 of Lecture Notes in Computer Science, pp. 280–310. Springer (2020)
13. Kim, P., Han, D., Jeong, K.C.: Time-space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2. Quantum Inf. Process. **17**(12), 1–39 (2018)
14. Langenberg, B., Pham, H., Steinwandt, R.: Reducing the cost of implementing the advanced encryption standard as a quantum circuit. IEEE Trans. Quantum Eng. **1**, 1–12 (2020). https://doi.org/10.1109/TQE.2020.2965697
15. Maitra, S.: Chosen IV cryptanalysis on reduced round ChaCha and Salsa. Discrete Appl. Math. **208**, 88–97 (2016)
16. Maitra, S., Paul, G., Meier, W.: Salsa20 cryptanalysis: new moves and revisiting old styles. In: The 9th International Workshop on Coding and Cryptography 2015 WCC2015, Anne Canteaut, Gaëtan Leurent, Maria Naya-Plasencia, Paris, France. ffhal-01276506f (2015)
17. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: quantum resource estimates. In: Takagi, T. (ed.) Post-quantum Cryptography. PQCrypto. Lecture Notes in Computer Science, vol. 9606, pp. 29–43. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29360-8_3
18. Ramos-Calderer, S., Bellini, E., Latorre, J.I., Manzano, M., Mateu, V.: Quantum search for scaled hash function preimages. arXiv preprint arXiv:2009.00621. (2020)
19. Shi, Z., Zhang, B., Feng, D., Wu, W.: Improved key recovery attacks on reduced-round Salsa20 and ChaCha. In: Information Security and Cryptology—ICISC, pp. 337–351 (2012)
20. Takahashi, Y., Tani, S., Kunihiro, N.: Quantum addition circuits and unbounded fan-out. Quantum Inf. Comput. **10**(9), 872–890 (2010)
21. NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016). https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf
22. Wiebe, N., Roetteler, M.: Quantum arithmetic and numerical analysis using repeat-until-success circuits. Quantum Inf. Comput. **16**(1–2), 134–178 (2014)
23. https://github.com/raviro/chacha