New Quantum Search Model on Symmetric Ciphers and Its Applications

Yangru Zheng\(^1\), Juntao Gao\(^1\), and Baocang Wang\(^1\)

School of Telecommunications Engineering, Xidian University, China
jtgao@mail.xidian.edu.cn

Abstract. It has been a long-standing viewpoint that doubling the length of key seeds in symmetric cipher can resist the quantum search attacks. This paper puts forward a quantum key search model to deal with the post-quantum security of symmetric ciphers. The quantum search is performed in the punctured keystream/ciphertext space instead of the key space. On inputting the punctured keystreams/ciphertexts, we rule out the spurious keys and find out the real key via the iterative use of the quantum singular value transformation search algorithm. We find out several parameters, such as the length of the punctured keystream/ciphertext, the iterations, and the error in the search algorithm, and all of them can influence the resulting complexity. When these parameters are chosen properly, a better complexity can be obtained than Grover algorithm.

Our search model can apply to any typical symmetric cipher. To demonstrate the power, we apply our model to analyze block ciphers AES family, stream ciphers Grain-128 and ZUC-like. The resulting complexity is $\widetilde{O}(2^{31.1})$ of AES-128, $\widetilde{O}(2^{32.4})$ of AES-192, $\widetilde{O}(2^{33.0})$ of AES-256, $\widetilde{O}(2^{27.7})$ of Grain-128, $\widetilde{O}(2^{33.3})$ of ZUC-128, and $\widetilde{O}(2^{31.7})$ of ZUC-256.

Our results show that increasing the length of key seeds is not an effective way anymore to resist the quantum search attacks, and it is necessary to propose new measures to ensure the post-quantum security of symmetric ciphers.

Keywords: Stream cipher · Block cipher · post-quantum security · quantum search algorithm.

1 Introduction

The symmetric cipher uses the same key in the encryption and decryption, which consists of block ciphers and stream ciphers.

The stream cipher [1] encryption process consists of an initialization process and a keystream output process. The initialization process doesn’t output any keystream, with input of a fixed-length key seed and an optional initialization vector (IV). And it makes sure that the key seed and initialization vector are sufficiently mixed to make the states on each register more random and to prepare for the keystream output process. The keystream output process starts after the initialization process is completed, and each clock output one symbol (bit,
byte or word), with updating states on each register. When the required length of outputting keystream is in polynomial length, a secure stream cipher can be considered as a pseudo-random function.

The block cipher \[ F : \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}^m \], encrypts a block \( m \) bits of plaintext into the ciphertext of the same length by a key in fixed length of \( n \). Moreover, \( F \) is a keyed function such that, for all keys \( k \), the function \( F_k \) defined by \( F_k(x) \overset{def}{=} F(k, x) \) is a bijection, i.e., a permutation. And the main distinction between block ciphers and pseudo-random permutations is that the former typically only support a specific set of key/block lengths, and in particular do not support arbitrary-length keys.

As for stream ciphers, it is difficult to traverse the full key space for the large size due to the limitation of classical computers, so the distribution of each symbol in the outputting keystream cannot be determined. Grover’s algorithm \[ 2 \] is able to search for \( M \) specific elements in a set with size of \( N \), achieving a squared speedup compared to classical search algorithms. Grover’s algorithm can be illustrated in geometry. In the two-dimensional plane spanned by the target vector and its orthogonal vector, the angle between the current quantum state and the two vectors can be calculated by \( M \) and \( N \). When Grover operator is applied, the current quantum state rotates fixed degree in the above plane. Based on the rotation degree, the number of the applied Grover operator can be calculated, so that the quantum state is rotated near the target vector, where the amplification of the target state is achieved. Grover’s algorithm can attack against stream cipher algorithms with an initialization vector given, by traversing the full key space on a quantum computer and amplifying the amplitude of the quantum state corresponding to the key seed. The complexity is \( O(2^n/2) \), where \( n \) denote the length of the key seed. In above attack, Grover’s algorithm requires that the search target is the unique correct key, which means the attacker must be given a portion of the keystream such that it is uniquely mapped to the correct key by the encryption function (no other spurious keys exist).

The search algorithm by quantum singular value transformation (QSVT) \[ 3 \], combined with quantum signal process, can search for the unknown amount of specific data in the overall \( N \) data. By transforming between the left and right singular spaces and rotating within the spaces, in the Bloch sphere representation, the quantum state keeps spirally approaching and finally converges to the target quantum state. Because of convergence property, the search algorithm by QSVT only needs to know the size \( N \) of the search set. Besides, the amplification of the target state can be achieved even if the search oracle is applied too many times. As for Grover’s algorithm, if only the size \( N \) of the search set is known, the optimal applied number of Grover search oracle can’t be calculated. There is great probability that the Grover search oracle operates too many (or too few) times, so that the total rotation angle in the two-dimensional plane is too large (or too small), and the quantum state deviates from the target state, resulting in a failure. Therefore, without knowing the number of the target in the dataset, the Grover’s algorithm is likely to fail, but the search algorithm by QSVT is still applicable.
In [4], a quantum signal processing framework is proposed, which uses $O(d)$ elementary unitary quantum operations to achieve quantum subsystem's evolutionary transformation by simulation to nearly arbitrary $d$ degree polynomials. The authors in [3] solves the synthesis problem of unitary quantum functions with a full characterization of achievable functions, and efficient techniques for their implementation. In [5], the authors propose a simulation algorithm required at most two auxiliary quits for the time-evolution operator $e^{-i\hat{H}t}$, such that the oracle in the algorithm is parameter-optimal in both asymptotic and non-asymptotic states. The key technology of the algorithm is qubitization, which uses a controlled oracle to embed the hermitian matrix $\hat{H}$ into the $SU(2)$ subspace. Qubitization forms a core tenet of quantum singular value transformation. In [6], the authors elaborate quantum singular value transformation combined with quantum signal process framework [4], and its applications on three central quantum problems, quantum search, factoring, and simulation.

As for block ciphers, Grover’s algorithm is widely used in key search model. In [16], the authors apply Grover oracle in the AES key search. For detail, given a small number of plaintext pairs, the key of AES is searching by Grover’s algorithm. And the AES attacking quantum circuit is designed with minimum qubits required and other quantum resources optimized, which has been adopted by the National Institute of Standards and Technology (NIST). In [17], the authors design an invertible quantum circuit for AES-128 algorithm with the same condition of minimum qubits, which decreases the number of quantum gates as well. The main optimized point is the quantum realization of S-box, which achieves the affine transformation in the S-box over finite field $GF(2^8)$ by Itoh-Tsujii algorithm [18]. In [13], the authors optimize the AES attacking quantum circuit based on [16], by searching two pairs of plaintext and ciphertext simultaneously in a quantum circuit, and prove that AES’s quantum security is weaker than it NIST declares. Besides, they determine the relation between the key length and the pairs, and design the LowMC attacking quantum circuit in the same way. In [15], the off-line Simon algorithm is applied to the 2XOR-Cascade construction, and the attacking complexity is beyond the quadratic speedup of quantum search algorithm.

This paper searches in keystream/ciphertext space rather than key space. When the keystream/ciphertext space size is larger than the key space size, both Grover’s algorithm and search algorithm by QSVT have squared acceleration effect. So consider a keystream/ciphertext space with much smaller size than key space. A short keystream/ciphertext may correspond to lots of keys, where a correct key and some spurious keys exist. At this time, Grover’s algorithm is not suitable, because of the unknown relation between key and keystream/ciphertext. Hence, using the search algorithm by QSVT [3], we propose a search model applicable to the quantum security analysis of symmetric ciphers. Firstly, we design a quantum algorithm to find out the specific bits of keystreams uniformly distributed on 0-1 space to generate a pseudo-random function, only for stream ciphers. And then, we design the single-round key search algorithm combined with the search oracle by QSVT, according to the opera-
tion rules of typical symmetric cipher. At last, based on the pseudo-randomness, run the single-round key search algorithm for \( r \) rounds, and return the correct key seed. Our search model attaches importance on how to obtain the key seed using quantum algorithms, so ultimately the attacking complexity against the symmetric cipher is our focus.

To verify the validity, we use our search model and instantiate it on block cipher AES family, stream ciphers Grain-128 and ZUC-like. The security of block cipher AES family is one of the most important issues in cryptanalysis. And these two stream ciphers are chosen for the representativity, and they both output the keystream after the initialization process. The keystream of the Grain type is one bit per symbol, and the ZUC type is one word per symbol.

The block cipher AES \cite{19}, released by NIST in 2001, is intended to replace DES as the widely used standard. In \cite{20}, the author publishes a probabilistic mixture-differential distinguisher on five rounds as well as a key-recovery attack on six rounds from \cite{21}, which costs \( 2^{72.8} \) required chosen plaintexts, \( 2^{105} \) time complexity and \( 2^{33} \) memory.

The stream cipher Grain is one of the hardware implementation-oriented stream ciphers solicited by the eSTREAM project. And the Grain-128 algorithm \cite{7} is proposed as an improvement on the Grain v0 algorithm. The main classical attack algorithm against Grain-128 is chosen IV attack. Itai Dinur and others propose a dynamic cubic attack \cite{8}, which recovers the key seed belonging to a large subset of \( 2^{-10} \) of the key space. For \( 2^{118} \) key seeds applicable to Grain-128 stream cipher, an attacker can obtain \( 2^{15} \) of improvement than the exhaustive search.

The stream cipher ZUC \cite{9} is identified as a next-generation international standard for LTE by the Third Generation Partnership Project Protocol (3GPP) in September 2011, and is established as a national standard in October 2016. The ZUC algorithm absorbs the advantages of cycle sequence generated by linear feedback shift register, bit-reorganization of Feistel structure with the nonlinearity, and S-box with the nonlinearity and strong diffusivity. The current classical attack algorithms are not effective in breaking the ZUC algorithm, so it is challenging to attack the ZUC algorithm.

By analyzing the number of quantum gates and the number of possible erroneous keys, we obtain optimal computational complexity of \( \tilde{O}(2^{27.7}) \) for Grain-128, \( \tilde{O}(2^{33.3}) \) for attacking ZUC-128, \( \tilde{O}(2^{34.7}) \) for attacking ZUC-256, \( \tilde{O}(2^{31.1}) \) for AES-128, \( \tilde{O}(2^{32.4}) \) for AES-192, and \( \tilde{O}(2^{33.0}) \) for AES-256.

Therefore, the attacking complexity in the quantum computing environment is much smaller than the classical attack. It indicates that two typical types of stream ciphers and a representative block cipher have some security risks in the quantum computing environment. Furthermore, increasing the key length of Grain-like or ZUC-like stream ciphers and AES family block cipher is no longer an effective method for enhancing the security to resist our search model.
2 Preliminaries

2.1 Symbol Description

The symbols and their representative meanings are shown in the Table 1.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Representative Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>key seed</td>
</tr>
<tr>
<td>$n$</td>
<td>key seed length</td>
</tr>
<tr>
<td>$v$</td>
<td>initialization vector/plaintext block</td>
</tr>
<tr>
<td>$m$</td>
<td>initialization vector/block length</td>
</tr>
<tr>
<td>$z$</td>
<td>punctured keystream/ciphertext</td>
</tr>
<tr>
<td>$s$</td>
<td>punctured keystream/ciphertext length</td>
</tr>
<tr>
<td>$r$</td>
<td>the iteration</td>
</tr>
<tr>
<td>$K_i$</td>
<td>key space</td>
</tr>
<tr>
<td>$E_v$</td>
<td>encryption function of symmetric ciphers with vector $v$</td>
</tr>
<tr>
<td>$p_s$</td>
<td>the puncture function</td>
</tr>
<tr>
<td>$g_{v,s}$</td>
<td>the compound function</td>
</tr>
<tr>
<td>$C_{g_{v,s}}$</td>
<td>the complexity of quantum circuit in realizing function $g_{v,s}$</td>
</tr>
<tr>
<td>$C_{block}$</td>
<td>the complexity of block cipher quantum circuit</td>
</tr>
<tr>
<td>$Init(n)$</td>
<td>the complexity of full rounds initialization process</td>
</tr>
<tr>
<td>$Output(n,s)$</td>
<td>the complexity of outputting s-bit keystream(without initialization)</td>
</tr>
</tbody>
</table>

2.2 Grover’s Algorithm

Grover’s algorithm solves the problem of searching some specific elements in the set $S = \{1, \ldots, N\}$. As for the function $f : \{1, \ldots, N\} \rightarrow \{0, 1\}$, Grover’s algorithm can find elements $\{\alpha\}$, for which $f(\alpha) = 1$. And for other elements $x \in \{1, \ldots, N\}\backslash\{\alpha\}$, $f(x) = 0$.

Set an operator $U_f : |x\rangle|y\rangle \rightarrow |x\rangle|y + f(x)\rangle$. If $x = \alpha$, $U_f|x\rangle|y + f(x)\rangle = |\alpha\rangle|y + 1\rangle$. Else, $U_f|x\rangle|y + f(x)\rangle = |x\rangle|y\rangle$. Specifically, if $|y\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, $U_f|x\rangle|y\rangle = (-1)^{f(x)}|x\rangle|y\rangle$. Else if $|y\rangle = |\phi\rangle$ is quantum state with arbitrary length,

$$U_f|\alpha\rangle|\phi\rangle = \sin(\theta)|\alpha\rangle|U(\phi)\rangle + \cos(\theta)|\psi^+\rangle,$$

where $(|\alpha\rangle|\alpha\rangle \otimes I) |\psi^+\rangle = 0$.

Define Grover operator $G = (2|\alpha\rangle\langle\alpha| - I)U_f$. We can get the relation

$$G^t|\alpha\rangle|\phi\rangle = \sin[(2t + 1)\theta]|\alpha\rangle|U(\phi)\rangle + \cos[(2t + 1)\theta]|\psi^+\rangle.$$

We apply above operators to the quantum state $|s\rangle$, which means $|s'\rangle = G|s\rangle$. As is shown in Fig. 1, the above process can be seen as a rotation of $2\theta$ degree on the two-dimension plane, which is spanned by the superposition state corresponding to the special vector and its orthonormal state.
When the goal is searching $M$ elements out of $N$, then
\[ \sin^2(\theta) = \frac{M}{N}, \quad 0 < \theta \leq \frac{\pi}{2}. \]

If $M \ll N$, $\theta \approx \sin(\theta) = \sqrt{M/N}$. We can successfully measure the special vectors $\{\alpha\}$ with probability of $\frac{1}{M}$, after applying $\left\lfloor \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rfloor$ operator $G$.

Above all, Grover algorithm can achieve quadratic acceleration compared to classical unordered database search algorithms.

### 2.3 Quantum Signal Processing

Set a quantum state $\ket{\psi} = \cos \frac{\theta}{2} \ket{0} + e^{i\phi} \sin \frac{\theta}{2} \ket{1}$, where $\theta \in [0, \pi]$, $\phi \in [0, 2\pi]$. The parameters $\theta$ and $\phi$ can locate a point in Bloch sphere [10], as shown in Fig. 2.

Quantum signal processing (QSP) is built on the idea of interleaving two kinds of single-qubit rotations: a signal rotation operator $W$, and a signal processing rotation operator $S$. These rotation operations are about different axes through the Bloch sphere. For instances, $W(a) = \begin{bmatrix} \frac{a}{\sqrt{1 - a^2}} & i \sqrt{1 - a^2} \\ i \sqrt{1 - a^2} & \frac{a}{\sqrt{1 - a^2}} \end{bmatrix}$ is an $x$-rotation of $\theta = -2 \cos^{-1}(a)$ degree, and $S(\phi) = e^{i\phi Z}$ is a $z$-rotation of $-2\phi$ degree.

**Definition 1.** [11] For a tuple of phases $\vec{\phi} = (\phi_0, \phi_1, \cdots, \phi_d) \in \mathbb{R}^{d+1}$, the QSP operation sequence $U_{\vec{\phi}}$ is defined as

\[ U_{\vec{\phi}} = e^{i\phi_0 Z} \prod_{k=1}^{d} W(a) e^{i\phi_k Z}. \]

Based on the definition 1, we have the following theorem:
Theorem 1. [11] The QSP sequence $U_{\phi} \rightarrow$ produces a matrix which may be expressed as a polynomial function of $a$:

$$e^{i\phi Z} \prod_{k=1}^{d} (W(a)e^{i\phi Z}) = \begin{bmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{bmatrix},$$

for $a \in [-1, 1]$, and $\phi$ exists for any polynomial $P, Q$ in $a$ such that:
1. $\deg(P) \leq d$, $\deg(Q) \leq d - 1$.
2. $P$ has parity $d \mod 2$, $Q$ has parity $(d - 1) \mod 2$.
3. $|P(a)|^2 + (1 - a^2)|Q(a)|^2 = 1$.

The authors in [4] indicate that, Remez-type exchange algorithm can compute a $\phi$ that produces a good approximation to any feasible polynomials $P$ and $Q$.

2.4 Search Algorithm by QSVT

Apply an operator $U$ to initial state $|B_0\rangle$. Our goal is to search target state $|A_0\rangle$ among states $U|B_0\rangle$.

Let $a = \langle A_0|U|B_0\rangle$, and $a \neq 0$(If $a = 0$, searching set $U|B_0\rangle$ doesn’t contain target state $|A_0\rangle$). If $a$ is known, Grover’s algorithm can be used for amplitude amplification. Else, if $a$ is unknown but the low bound of $a$, the search algorithm by quantum singular value transformation(QSVT) can solve the problem. The search algorithm by QSVT is elaborated as following:

Let $|A_{\perp}\rangle = \frac{1}{N} (I - |A_0\rangle\langle A_0|)U|B_0\rangle$, where $N$ is the normalization factor needed to make $|A_{\perp}\rangle$ a unit vector. And $U|B_0\rangle = a|A_0\rangle + \sqrt{1 - a^2}|A_{\perp}\rangle$, $U|B_{\perp}\rangle = -a|A_{\perp}\rangle + \sqrt{1 - a^2}|A_0\rangle$. Besides, the singular value decomposition is

$$U = a (|A_0\rangle\langle B_0| - |A_{\perp}\rangle\langle B_{\perp}|) + \sqrt{1 - a^2} (|A_{\perp}\rangle\langle B_0| + |A_0\rangle\langle B_{\perp}|),$$
where \( |A_0\rangle, |A\rangle \) are left singular vectors, and \( |B_0\rangle, |B\rangle \) are right singular vectors.

Thus, the block encoding of the operator \( U \) is

\[
U = \begin{bmatrix} a & \sqrt{1-a^2} \\ \sqrt{1-a^2} & -a \end{bmatrix}.
\]

The search algorithm by QSVT can measure the target state \( |A_0\rangle \) with the probability approximate to \( 1(poly(a) \rightarrow 1) \), combined quantum signal processing in a way of applying a sequence of rotation operators and operator \( U \). Based on Theorem 1, Theorem 2 can be deduced:

**Theorem 2.**[11] Given a unitary \( U \), its inverse \( U^\dagger \), and operator \( A_\phi = e^{i\phi|A_0\rangle\langle A_0|} \), \( B_\phi = e^{i\phi|B_0\rangle\langle B_0|} \),

\[
\langle A_0 | \prod_{k=1}^{d/2} U B_{\phi_{2k-1}} U^\dagger A_{\phi_{2k}} | B_0 \rangle = poly(a),
\]

where \( poly(a) \) is a polynomial in \( a = \langle A_0 | U | B_0 \rangle \) of degree at most \( d \), satisfying the conditions on \( P \) from Theorem 1.

As shown in Fig. 3, the geometry representation of Theorem 2 is as below:

Let \( H_A = \text{span}(|A_0\rangle, |A\rangle) \) and \( H_B = \text{span}(|B_0\rangle, |B\rangle) \) denote two invariant subspaces separately spanned by left/right singular vectors.

1. The operator \( U \) maps vectors in space \( H_B \) to vectors in space \( H_A \) with a rotation.
2. The operator \( A_\phi \) works as a rotation around vector \( |A_0\rangle \) with certain degree, and the operator \( B_\phi \) works around vector \( |B_0\rangle \).
3. The operator \( U^\dagger \) maps vectors in space \( H_A \) to vectors in space \( H_B \) with a rotation.

Theorem 2 indicates that, search algorithm by QSVT combined with quantum signal proceeding, makes the final vector gradually converge on target vector \( |A_0\rangle \) in a way of transforming between two singular vector spaces and rotations around singular vectors.

As for Grover’s algorithm, on condition that only the low bound of \( a = \langle A_0 | U | B_0 \rangle \) is known, we try to increase the number of applied Grover operator to solve the problem. Specifically, if the low bound of \( a \) works as parameter in Grover’s algorithm but not exact value of \( a \), the number of applied Grover operator can be much more than the optimal number corresponding to Grover’s algorithm with exact value of \( a \). As shown in Fig. 1, when the number of Grover operator is more than excepted, vector rotates a lager angle in total in the plane, and the probability \( p = |\langle a \otimes \phi|S^fU_j|s\rangle|^2 \) no longer tends to 1. In result, the final vector is diverging from target vector, which fails amplitude amplification.

In a word, compared to Grover’s algorithm, search algorithm by QSVT doesn’t have the problem that too many rotation operators fail amplitude amplification because of convergence.
In Theorem 2, the optimal function for $poly(a)$ is sign function

$$\Theta(x - c) = \begin{cases} 
-1 & x < c \\
0 & x = c \\
1 & x > c 
\end{cases}$$

And sign function $\Theta(x - c)$ can be estimated with arbitrary precision by finding a polynomial approximation to gauss error function $erf \left( k \left[ x - c \right] \right)$, for large enough $k$. Particularly, a degree $d = O \left( \frac{1}{\sqrt{\log \left( \frac{1}{\varepsilon} \right)} \varepsilon} \right)$ odd polynomial $P_{\varepsilon, \Delta}^{\Theta}(x - c)$ can be computed, where $\varepsilon \in \left( 0, \frac{\sqrt{2}}{e\pi} \right)$, and such that

1. $|P_{\varepsilon, \Delta}^{\Theta}(x - c)| \leq 1$, for $x \in [-1, 1]$.
2. $|\Theta(x - c) - P_{\varepsilon, \Delta}^{\Theta}(x - c)| \leq \varepsilon$, for $x \in [-1, 1] \setminus (c - \frac{\Delta}{2}, c + \frac{\Delta}{2})$.

All in all, $P_{\varepsilon, \Delta}^{\Theta}(x - c)$ can $\varepsilon$-approximate sign function, as shown in Fig. 4.
Let $N$ denote the searching set’s size, and $|a| = |\langle A_0 | U | B_0 \rangle| \geq \frac{1}{\sqrt{N}}$. In Fig. 4, for an arbitrary value $|a| \geq \frac{1}{2} \sqrt{\frac{1}{N}}$, $P_{\varepsilon, \Delta}(a) = 1$, when $\Delta / 2 \leq \frac{1}{2} \frac{1}{\sqrt{N}}$. Thus, we get the Theorem 3.

**Theorem 3.** [11] Given unitary operators $U$, $U^\dagger$, and rotation operators $A_\phi = e^{i \phi A_0} A_0$, $B_\phi = e^{i \phi B_0} B_0$,

$$\langle A_0 \right| \left[ \prod_{k=1}^{d/2} U B_{2k-1} U^\dagger A_{2k} \right] U | B_0 \rangle = P_{\varepsilon, \Delta}^\theta (x - c),$$

where $P_{\varepsilon, \Delta}^\theta (x - c)$ is a polynomial with degree at most $d$, satisfying the conditions on polynomial $P$ in Theorem 1, $\Delta \leq \frac{2}{\sqrt{N}}$, and $d = \mathcal{O} \left( \frac{1}{\Delta} \log \left( \frac{1}{\varepsilon} \right) \right) = \mathcal{O} \left( \sqrt{N} \log (1/\delta) \right)$.

And the odd polynomial $P_{\varepsilon, \Delta}^\theta (x)$ is an approximation of $\Theta(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$

satisfying the conditions in Theorem 1 with $\text{poly}(a) = \langle + | U \phi | + \rangle$. At this point, the operator sequence $U_{\phi} = (P_{\varepsilon, \Delta}^\theta)^{(SV)}(W) \approx \Theta^{(SV)}(W)$.

In summary, search algorithm by QSVT is as below:

---

**Algorithm 1: Unstructured Search Algorithm by QSVT**[11]

**Input:** Access to a controlled version of the oracle $U$ which bit-flips an auxiliary qubit when given an unknown target state $|m\rangle$, an error tolerance $\delta = 2\varepsilon$, and a $\Delta / 2 \leq 1/\sqrt{N}$.

**Output:** The flagged state $|m\rangle$.

1. Use QSVT to construct the operator $(P_{\varepsilon, \Delta}^\theta)^{(SV)}(W)$, where $W$ is the block encoding of $U$.

2. Apply $(P_{\varepsilon, \Delta}^\theta)^{(SV)}(W)$ to the uniform superposition. If the auxiliary is measured as $|+\rangle$, then $|m\rangle$ remains in the register. Else, repeat the above process.

Algorithm 1 succeeds in the probability of at least $1-\delta$, and costs 1 extra auxiliary qubit with complexity of

$$\tilde{O} \left( \frac{1}{\Delta} \log \left( \frac{1}{\delta} \right) \right) = \tilde{O} \left( \sqrt{N} \log (1/\delta) \right),$$

which obeys well-established quantum lower bounds on the hardness of unstructured search in [23].
New Quantum Search Model on Symmetric Ciphers and Its Applications

3 Quantum Key Search Model for Symmetric Ciphers

In this section, we design a general search model which is applicable to search the key both in block ciphers and stream ciphers. Because of the pseudo-randomness, the procedure in searching the block cipher key is simpler than stream cipher’s.

3.1 General Key Search Model

Our search model is taking advantage of pseudo-randomness, part of the security of symmetric ciphers.

Preset The construction of a pseudo-random function includes three steps.

Firstly, we set the encryption function of symmetric cipher,

\[ E_v(\cdot) : \{0, 1\}^n \rightarrow \{0, 1\}^h, \]

where \( n \) represents the key length, \( k \) represents the key, and vector \( v \) has length of \( m \) bits. Specifically, for block ciphers, vector \( v \) represents the plaintext block, vector \( z \) represents the corresponding ciphertext, \( h \) and \( m \) represents the block length (\( h = m \)). As for stream ciphers, vector \( v \) represents the initialization vector with length of \( m \), and vector \( z \) represents the corresponding \( h \)-bit keystream.

Secondly, we design a puncture function \( p_s(\cdot) : \{0, 1\}^h \rightarrow \{0, 1\}^s \),

\[ p_s((z_1, \cdots, z_h)) = (z_{i_1}, \cdots, z_{i_s}), \]

where \( \{i_1, \cdots, i_s\} \subset \{1, \cdots, h\} \), and for \( 1 \leq j < k \leq s, i_j < i_k \). In this way, define the punctured keystream/ciphertext \( z' \), which is punctured by function \( p_s(\cdot) \), i.e. \( z' = p_s(z) = (z_{i_1}, \cdots, z_{i_s}) \).

At last, in this way, we define a compound function \( g_{v,s} : \{0, 1\}^n \rightarrow \{0, 1\}^s \),

\[ g_{v,s}(k) = p_s \circ E_v(k). \]

By a delicate selection of \( i_1, \cdots, i_s \) (proper design of function \( p_s \)), \( g_{v,s} \) can be guaranteed as a pseudo-random function, which our search model take use of. And we assume the function \( g_{v,s} \) can be achieved in the oracle \( O_{g_{v,s}} \). We discuss two designs of puncture function \( p_s \) separately for block ciphers and stream ciphers in next section.

Key Search Model Design Based on the pseudo-randomness of function \( g_{v,s} \), we design a multi-round key search model (as Algorithm 3) with \( r \) pairs of vector \( v \) and \( z \), as a chosen IV/plaintext attack. Besides, we design a single-round key search algorithm (as Algorithm 2) with input of a pair \((v_i, z_i)\) as a subroutine in multi-round key search model. Apart from that, the single-round key search algorithm can work as a dependent algorithm to search the key. But to ensure the key’s uniqueness and correctness, this algorithm has the same or
even worse effect than Grover’s algorithm. However, working as a subroutine, the total complexity will outperform the Grover’s algorithm.

The following is the detail of single-round key search algorithm.

1. Single-round Key Search Model

Suppose an attacker want to get the key \( k^* \) of symmetric cipher. The attacker can choose an initialization vector/plaintext \( v_i \), and get the \( s \)-bit punctured keystream/ciphertext \( z_i = (z_{i,1}, z_{i,2}, \ldots, z_{i,s}) \), corresponding to \( (k^*, v_i) \). And then, search \( z_i \) in punctured keystream/ciphertext space \( Z = \{ z | g_{v_i,s}(k) = z, k \in \{0, 1\}^n \} \) by Algorithm 1 to measure \( k^* \) in full key space, where \( g_{v_i,s}(k^*) = z_i \).

Noticed, we don’t use search algorithm in key space but keystream/ciphertext space. If \( s \geq n \), searching space \( Z \) has the almost same size of the key space where the key \( k \) in. Else, the size of searching space \( Z \) is \( |g_{v_i,s}(\{0, 1\}^n)| \leq 2^s \).

The reasonable value range of \( s \) is discussed later. The design of single-round search algorithm is as following: (quantum circuit is shown in Fig. 5).

<table>
<thead>
<tr>
<th>Algorithm 2: Single-round Key Search Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> The ( m )-bit vector ( v_i ) and ( s )-bit vector ( z_i = (z_{i,1}, z_{i,2}, \ldots, z_{i,s}) ).</td>
</tr>
<tr>
<td><strong>Output:</strong> The ( n )-bit key.</td>
</tr>
<tr>
<td>1 Prepare the state (</td>
</tr>
<tr>
<td>2 Add ( s ) qubits in the third register ( Z ), where (</td>
</tr>
<tr>
<td>3 Apply ( n ) Hadamard gates to ( n ) qubits in the first register ( K ), where (</td>
</tr>
<tr>
<td>4 Load the vector ( v_i ) in the second register ( V ), where ( \frac{1}{\sqrt{2^n}} \sum_{k \in {0, 1}^n}</td>
</tr>
<tr>
<td>5 Apply the oracle ( O_{g_{v_i,s}} ) and store the output vector ( z = g_{v_i,s}(k) ) on register ( Z ), where ( \frac{1}{\sqrt{2^n}} \sum_{k \in {0, 1}^n}</td>
</tr>
<tr>
<td>6 Add one auxiliary qubit stored in register ( flag ), and initialize the state to (</td>
</tr>
<tr>
<td>7 Apply the search algorithm by QSVT to amplitude the amplification of (</td>
</tr>
<tr>
<td>8 Measure register ( flag ), where the quantum state collapses to ( \frac{1}{\sqrt{</td>
</tr>
<tr>
<td>9 Uncompute the step 5 and 4, where ( \frac{1}{\sqrt{</td>
</tr>
<tr>
<td>10 Measure and return the state in register ( K ).</td>
</tr>
</tbody>
</table>

Here are some implementation details about Algorithm 2.
a. The search algorithm in step 7 is Algorithm 1, where the parameter $\Delta$ is related to $s$ and $n$.
b. The oracle $O_{g,v,s}$ in step 5, which achieves function $g_{v,s}(k)$, and takes a key $k$ as input, vector $v_i$ as parameter, and $s$-bit vector $z = g_{v,s}(k)$ as output.
c. In step 9, the uncomputing operation aims for rolling back the state on registers to recover the original key and other all-zero state for the measurement or the loading of vector $v$ in next round search (Algorithm 3).

In order to measure the key $k^*$, we have to choose $s > n$, if run Algorithm 2 only once. Let \( C_{g,v,s} \) denote the gate count of oracle $O_{g,v,s}$. The complexity of Algorithm 2 is
\[
\tilde{O} \left( \sqrt{2^n} \log(1/\delta) \cdot C_{g,v,s} \right),
\]
for the condition that $s > n$, $\Delta \leq 2/\sqrt{2^n}$. And it is larger than $2^2$. Consequently, the designer can extend the key’s length to resist the attack from Algorithm 2.

Hence, we design a multi-round key search model by QSVT, based on Algorithm 2. In brief, we shrink the length of punctured keystream/ciphertext to reduce searching complexity each round, and repeat Algorithm 2 $r$ times.

2. Multi-round Key Search Model

Algorithm 3 reduces the complexity of search algorithm in a smaller searching space of $s$-bit punctured keystream/ciphertext $z$, which in return has a better effect than quadratic speedup. The quantum circuit of Algorithm 3 is shown in Fig. 6.

**Algorithm 3:** Multi-round Key Search Model

**Input:** The $r$ pairs of $m$-bit vector $v_i$ and $s$-bit vector $z_i$, $i = 1, \ldots, r$.

**Output:** The $n$-bit key.

1. Run step 1 to step 9 of Algorithm 2 with inputs of vector $v_1$ and $z_1$. The state in registers $K,V,Z$ is $\frac{1}{\sqrt{|K|}} \sum_{k \in K} |k\rangle |0\rangle^{\otimes(m+s)}$, where $K_1 = \{k|g_{v_1,s}(k) = z_1\}$. Quantum state collapses because of the measurement on the register flag.
2. For $i = 2, \ldots, r$, each time repeat step 4 to step 9 of Algorithm 2 with inputs of vector $v_i$ and $z_i$.
3. Measure and return the state in register $K$, which belongs to key space $K_r = \{k|g_{v_i,s}(k) = z_i, i = 1, \ldots, r\}$.

Set the key space $K_i = \{k|g_{v_i,s}(k) = z_j, 1 \leq j \leq i, k \in \{0,1\}^n\}, 1 \leq i \leq r$. For a convenient expression, set $K_0 = \{0,1\}^n$. In this way, the alternation of key space in each round of Algorithm 3 is
\[
K_0 \xrightarrow{g_{v_1,s}(\cdot)=z_1} K_1 = \{k|g_{v_1,s}(k) = z_1\} \xrightarrow{g_{v_2,s}(\cdot)=z_2} K_2 = \{k|g_{v_2,s}(k) = z_2, g_{v_1,s}(k) = z_1\} \xrightarrow{g_{v_3,s}(\cdot)=z_3} \cdots \xrightarrow{g_{vr,s}(\cdot)=z_r} K_r = \{k|g_{v_i,s}(k) = z_i, i = 1, \ldots, r\}.
\]

Hence, $K_{i+1} = \{k|g_{v_{i+1},s}(k) = z_{i+1}, k \in K_i\}, 0 \leq i \leq r - 1$. 


In particular, the alternation of key space each round is \( K_i \xrightarrow{\text{search}} K_{i+1} \). It consists of two parts round by round, amplification of amplitude of the state corresponding to \( \{ k \in K_{i+1} | k \in K_i \} \) (step 7 in Algorithm 2), and elimination the state corresponding to \( \{ k \notin K_{i+1} | k \in K_i \} \) (step 8 in Algorithm 2).

a. Amplification of the state corresponding to \( k \in K_{i+1} \).

Focus on the quantum state before searching process (step 7 in Algorithm 2),

\[
|\psi_i\rangle = \frac{1}{\sqrt{|K_i|}} \left( \sum_{k \in K_{i+1}} |k\rangle|v_{i+1}'\rangle|z_{i+1}\rangle|+\rangle + \sum_{k \notin K_{i+1}} |k\rangle|v_{i+1}'\rangle|g_{v_{i+1},s}(k)\rangle|-\rangle \right)
\]

And by the definition of \( K_{i+1} \), \( \forall k \notin K_{i+1}, \langle z_{i+1}|g_{v_{i+1},s}(k)\rangle = 0 \).

Set \( |\alpha_{i+1}\rangle = \frac{1}{\sqrt{|K_{i+1}|}} \sum_{k \in K_{i+1}} |k\rangle|v_{i+1}'\rangle \), we can get

\[
|\psi_i\rangle = \sqrt{|K_{i+1}|} |\alpha_{i+1}\rangle|z_{i+1}\rangle|+\rangle + \frac{1}{\sqrt{|K_i|}} \sum_{k \notin K_{i+1}} |k\rangle|v_{i+1}'\rangle|g_{v_{i+1},s}(k)\rangle|-\rangle
\]

For the pseudo-randomness of \( g_{v_{i+1},s} \),

\[
\forall z \in \{0,1\}^s, \Pr(g_{v_{i+1},s}(k) = z | k \in K_i) = \frac{1}{2^s}.
\]

(2)
Set a random variable $X$ represents the number of elements in $K_{i+1}$. By formula (2),
\[ X \sim B\left(|K_i|, \frac{1}{2^s}\right), \]
where $E(X) = \frac{|K_i|}{2^s}$, $D(X) = \frac{|K_i|}{2^s} \left(1 - \frac{1}{2^s}\right)$. For the large $|K_i|$ and small $\frac{1}{2^s}$, $X$ approximately follows the normal distribution. Hence,
\[ \frac{X - E(X)}{\sqrt{D(X)}} \sim N(0, 1). \]

And the 95.45% confidence interval is
\[ -2 \leq \frac{X - E(X)}{\sqrt{D(X)}} \leq 2, \]
i.e.,
\[ \frac{1}{2^s} \left(1 - 2 \sqrt{\frac{2^s - 1}{|K_i|}}\right) \leq \frac{X}{|K_i|} = \frac{|K_{i+1}|}{|K_i|} \leq \frac{1}{2^s} \left(1 + 2 \sqrt{\frac{2^s - 1}{|K_i|}}\right). \]

Set $|K_i| = 2^{s+h}$. There are two conditions.

i. If $h \geq 0$, then
\[ \frac{1}{2^s} \left(1 - 2 \sqrt{\frac{2^s - 1}{|K_i|}}\right) > \frac{1}{2^s} \left(1 - 2 \sqrt{\frac{2^s}{|K_i|}}\right) = \frac{1}{2^s} \left(1 - \frac{1}{2^s - 1}\right) \geq \frac{1}{2^{s+1}}(h \geq 4) \]
works with condition that $h \geq 4$. Thus, if the number $|K_i|$ of elements in key space satisfy
\[ |K_i| \geq 2^{s+4}, \] (3)
then the search algorithm by QSVT can function well with a low bound of $2^{\frac{s+4}{2}}$.

ii. If $h < 0$, then
\[ \frac{1}{2^{s+1}} < \frac{1}{2^s} \leq \frac{X}{|K_i|} = \frac{|K_{i+1}|}{|K_i|}. \]

In the last searching round, the size of punctured vector space must be smaller than $2^s$, while the percentage of the target vector in searching space is more than $\frac{1}{2^s}$. However, for the convergence of search algorithm by QSVT, it still works well.

In summary, if $h \geq 4$ or $h < 0$, then
\[ \frac{1}{2^{s+1}} < \sqrt{\frac{|K_{i+1}|}{|K_i|}}, \text{i.e.,} \]
\[ \frac{1}{2^{s+1}} < \sqrt{\frac{|K_{i+1}|}{|K_i|}}. \]
We take this common low bound into search algorithm by QSVT, i.e., put \( \Delta/2 \leq 2^{-\frac{s+1}{2}} \) into formula (1), and get the one round search complexity

\[ \tilde{O} \left( 2^{\frac{s+1}{2}} \cdot \log \left( \frac{1}{\delta} \right) C_{g,s} \right). \]  

(4)

b. Elimination of the state corresponding to \( k \in K_i - K_{i+1} \).

After the search process (step 7 in Algorithm 2), we get the quantum state

\[ |\psi'\rangle = a_1 \sum_{k \in K_{i+1}} |k'\rangle |v'_{i+1}\rangle |z_{i+1}\rangle |+\rangle + a_2 |\psi^\perp\rangle |-\rangle \]

\[ = \left( a_1 \sqrt{|K_{i+1}|} \right) \left( \frac{1}{\sqrt{|K_{i+1}|}} \sum_{k \in K_{i+1}} |k'\rangle |v'_{i+1}\rangle |z_{i+1}\rangle \right) |+\rangle + a_2 |\psi^\perp\rangle |-\rangle, \]

where \( a_1 \rightarrow \frac{1}{\sqrt{|K_{i+1}|}} \) and \( a_2 \rightarrow 0 \).

So it’s of nearly 1 possibility to measure \(|+\rangle\) on register \( \text{flag} \). As Simon’s algorithm shows, the measurement on register \( \text{flag} \) triggers the collapse of quantum state components on register \( K, V \) and \( Z \), which means only the state components corresponding to \(|+\rangle\) remains. In this way, the state on registers is

\[ |\psi''\rangle = \frac{1}{\sqrt{|K_{i+1}|}} \sum_{k \in K_{i+1}} |k'\rangle |v'_{i+1}\rangle |z_{i+1}\rangle |+\rangle. \]

The major purpose of measurement on register \( \text{flag} \) is to avoid that the wrong keys re-entry the following search rounds. Hence, in next round of searching, it is needed to prepare a new auxiliary qubit again.

All in all, the multi-round key search model costs

\[ n + m + s + r + q \]

qubits, and by formula (4), its complexity is

\[ \tilde{O} \left( r \cdot 2^{\frac{s+1}{2}} \cdot \log \left( \frac{1}{\delta} \right) C_{g,v} \right), \]  

(5)

where \( n \) represents key length, \( m \) represents the length of vector \( v \), \( s \) represents the length of vector \( z \), \( r \) represents the number of auxiliary qubits or the iteration, \( q \) represents the number of extra qubits in the quantum realization of symmetric cipher, and \( C_{g,v} \) represents the complexity of oracle \( O_{g,v} \).

**Parameter Selection**

In this section, we elaborate the selection rules of parameter \( s \) and \( r \), to guarantee two points. First is that search algorithm by QSVT can work well with parameter \( \Delta/2 \leq 2^{-\frac{s+1}{2}} \). And the second point is that the returning key \( k \in K_r \) is unique and correct.
1. Make sure search algorithm by QSVT work well each round.
   In previous, we take $2^{-\frac{s^2}{2}}$ as the common low bound with the condition of formula (3).
   Let’s analyze the last but one round, $K_{r-2} \xrightarrow{\text{search}} K_{r-1}$, where $|K_{r-2}| > 2^s > |K_{r-1}|$. Obeying the formula (3), we have to make sure
   $$|K_{r-2}| \geq 2^{s+4}.$$  
   Noticed that in the $(r-2)_{th}$ round, $K_{r-3} \xrightarrow{\text{search}} K_{r-2}$. Because of formula (3),
   $$|K_{r-2}| = |K_0| \frac{|K_1| \cdots |K_{r-2}|}{|K_0|} \geq |K_0| \left( \frac{1}{2s+1} \right)^{r-2} = 2^{n-(s+1)(r-2)}.$$ 
   In this way, $n - (s + 1)(r - 2) \geq s + 4$, i.e.,
   $$n - sr + s - r - 2 \geq 0,$$
   which means if parameter $s$ and $r$ satisfy the formula (6), the search algorithm by QSVT can work well each round.
   2. Guarantee the returning key is unique and correct.
   According to paper [13], we give the following analysis to make sure that the measurement is the correct key.
   Let $k'$ denote the $n$-bit key, $k^*$ denote the correct key, and $(v, z)$ denotes a pair of vectors, where $(v, z)$ is (initialization vector, punctured keystream) for stream ciphers, and (plaintext block, punctured ciphertext) for block ciphers.

   **Definition 2.** For $r$ pairs $(v_1, z_1) \cdots, (v_r, z_r)$, if the key $k'$ satisfies $g_{v_i, s}(k') = g_{v_i, s}(k^*) = z_i$, $i = 1, \cdots, r$ and $k' \neq k^*$, then $k'$ is called spurious key.

   It’s of great possibility for the existence of spurious keys because of the short length of vector $z$.
   By the pseudo-randomness of function $g_{v, s}(\cdot)$,
   $$\forall k' \in \{0,1\}^n, \quad \Pr_{k' \neq k^*} (g_{v_i, s}(k') = g_{v_i, s}(k^*)) = \frac{1}{2^s}.$$ 
   Given $r$ pairs of $(v_1, z_1), \cdots, (v_r, z_r)$,
   $$p = \Pr_{k' \neq k^*} (g_{v_i, s}(k^*) = g_{v_i, s}(k'), i = 1, \cdots, r) = \prod_{j=0}^{r-1} \frac{1}{2^s - j}.$$ 
   For the condition
   $$r^2 \ll 2^s,$$
   we have
   $$p = \prod_{j=0}^{r-1} \frac{1}{2^s - j} \approx 2^{-rs}.$$ (8)
Set the spurious key set as $\{k'|k' \neq k^*, (g_{v,s}(k^*), i = 1, \cdots , r)\}$. By formula (8), $|SK| \approx (2^n - 1)2^{-rs}$.

Let random variable $X$ be the number of spurious $|SK|$, where $X$ follows the binomial distribution, and

$$\Pr(X = \alpha) = C^\alpha_{2^n-1}p^\alpha(1-p)^{2^n-1-\alpha}.$$ 

By formula (8),

$$\Pr(X = 0) = C^0_{2^n-1}p^0(1-p)^{2^n-1} = (1-p)^{2^n-1} \approx e^{-2^n-rs}.$$ 

Hence, it is $p \approx e^{-2^n-rs}$ probability to return a unique and correct key.

In order to guarantee the success probability and use fewer pairs $(v, z)$, we make the parameter $r$ satisfy

$$r = \left\lfloor \frac{n}{s} \right\rfloor + 1 \quad (9)$$

Above all, if the parameters $r$ and $s$ satisfy the formulas (7) and (9), we can guarantee that the probability of returning a unique key is $p \approx e^{-2^n-rs}$.

All in all, combined with formulas (6), (7) and (9), we get the selection rules

$$\begin{cases} 
    n - s\left\lfloor \frac{n}{s} \right\rfloor - \left\lfloor \frac{n}{s} \right\rfloor - 3 \geq 0 \\
    1000 \cdot r^2 \leq 2^s \\
    r = \left\lfloor \frac{n}{s} \right\rfloor + 1
\end{cases},$$

where we replace formula (7) with $1000 \cdot r^2 \leq 2^s$ for simplicity.

Take three values of $n$, the length of key, where $n = 128$, $n = 192$ and $n = 256$, we get the optimal solutions in Table 2.

<table>
<thead>
<tr>
<th>Key length $n$</th>
<th>Punctured length $s$</th>
<th>Iterations $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>192</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>256</td>
<td>20</td>
<td>13</td>
</tr>
</tbody>
</table>

### Table 2. Solutions with Different Values of Key Length

#### 3.2 Quantum Key Search Model for Block Ciphers

In this section, we construct a pseudo-random function $g_{v,s}$ for block ciphers.

Concluded from the structure of block ciphers, we can make the assumption.

**Assumption 1** The encryption function of block cipher is a strong pseudo-random function.
Set the encryption function of block cipher as
\[ E_v : \{0, 1\}^n \rightarrow \{0, 1\}^h, \; k \mapsto z', \]
where \( k \) denotes the key of \( n \) bits, \( v \) denotes the plaintext block of \( m \) bits, and \( z' \) denotes corresponding \( h \)-bit ciphertext \((h = m)\). By Assumption 1, function \( E_v(\cdot) \) is a pseudo-random function.

Set the puncture function
\[ p_s((x_1, \cdots, x_h)) = (x_1, \cdots, x_s), \]
where \( s \leq h \). In this way, we define the first \( s \) bits of ciphertext as punctured ciphertext.

Define a compound function \( g_{v,s} : \{0, 1\}^n \rightarrow \{0, 1\}^s \)
\[ g_{v,s}(k) = p_s \circ E_v(k) = p_s(z') = z. \]
It is easy to conclude that function \( g_{v,s}(\cdot) \) is a strong pseudo-random function as well. By the definition and property of \( g_{v,s}(\cdot) \), we can map each keys \( k \) to the punctured ciphertext \( z \) uniformly, which means
\[ \forall z \in \{0, 1\}^s, \; \Pr_{k \in \{0, 1\}^n}(g_{v,s}(k) = z) = \frac{1}{2^s}. \]

It’s easy to conclude that the operation for achieving puncture function \( p_s \) only takes negligible constant gates. Hence, the complexity \( C_{g_{v,s}} \) of oracle \( O_{g_{v,s}} \) is almost equal to the complexity \( C_{\text{block}} \) of encryption in block ciphers, i.e., \( C_{g_{v,s}} = C_{\text{block}} \).

### 3.3 Quantum Key Search Model for Stream Ciphers

In this section, we construct a pseudo-random function \( g_{v,s} \) for stream ciphers.

Different from block ciphers, there is no proof that the encryption function of stream ciphers is pseudo-random either.

So the construction of function \( p_s \) is our main problem, which extracts out some special bits of keystream to make \( g_{v,s} \) a pseudo-random function. We design a quantum keystream bit-wise distribution algorithm to solve it, as Algorithm 4 shows.

**Preprocessing Algorithm** We set \( h \) boolean functions
\[ Out_i(k, v) : \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}, (k, v) \mapsto z_i \]
for a large enough \( h \), and \( i = 1, \cdots, h \), which maps from the key \( k \) and initialization vector \( v \) to the \( i_{th} \) bits of keystream \( z \). And the main purpose of Algorithm 4 is to find out whether boolean function \( Out_i \) is balanced or not.
Algorithm 4: Quantum Keystream Bit-wise Distribution Algorithm

**Input:** The parameter $h$.
**Output:** The distribution of bits in keystream.

1. Run Algorithm 4 $c$ times, where $c$ is constant. Make a note of each measurement on register $Z$, $z_i = (z_{i,1}, \cdots, z_{i,h})$, $i = 1, \cdots, c$.
2. Let $z^* = (z_{1}^*, \cdots, z_{h}^*)$ denote distributions of each bit (whether is uniform or not), where $z_j^* = z_{1,j} \lor \cdots \lor z_{c,j}$, and $j = 1, \cdots, h$. If $z_j^* = 0$, then the $j_{th}$ bit

Focus on the quantum state after step 4,

$$|\phi\rangle = \frac{1}{\sqrt{2^{n+m}}} \sum_{k=0}^{2^n-1} \sum_{v=0}^{2^m-1} |k\rangle |v\rangle |z\rangle$$

$$= \frac{1}{\sqrt{2^{n+m}}} \sum_{k=0}^{2^n-1} \sum_{v=0}^{2^m-1} |k\rangle \otimes |v\rangle \otimes |Out_1(k,v)\rangle |Out_2(k,v)\rangle \cdots |Out_h(k,v)\rangle.$$  

After $h$ Hadamard gates on register $Z$, the quantum state is

$$|\phi'\rangle = \frac{1}{\sqrt{2^{n+m+h}}} \sum_{k=0}^{2^n-1} \sum_{v=0}^{2^m-1} |k\rangle |v\rangle \otimes |\phi_i\rangle \otimes |0\rangle + (-1)^{Out_i(k,v)} |1\rangle.$$  

And the bit-wise state on register $Z$ now is

$$|\phi'_{Z_i}\rangle = \frac{1}{\sqrt{2^{n+m+1}}} \sum_{k=0}^{2^n-1} \sum_{v=0}^{2^m-1} \left( |0\rangle + (-1)^{Out_i(k,v)} |1\rangle \right)$$

$$= \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2^{n+m+1}}} \left( \sum_{k=0}^{2^n-1} \sum_{v=0}^{2^m-1} (-1)^{Out_i(k,v)} \right) |1\rangle.$$  

Hence, the bit-wise measurement of $|\phi'_{Z_i}\rangle = |1\rangle$ means that the $i_{th}$ bit is non-uniformly distributed, i.e., $Out_i$ isn’t a balanced function.

Thus, the specific process is as following:

1. Run Algorithm 4 $c$ times, where $c$ is constant. Make a note of each measurement on register $Z$, $z_i = (z_{i,1}, \cdots, z_{i,h})$, $i = 1, \cdots, c$.
2. Let $z^* = (z_{1}^*, \cdots, z_{h}^*)$ denote distributions of each bit (whether is uniform or not), where $z_j^* = z_{1,j} \lor \cdots \lor z_{c,j}$, and $j = 1, \cdots, h$. If $z_j^* = 0$, then the $j_{th}$ bit
is uniformly distributed, and $Out_j$ is a balanced boolean function. Else, the $j_{th}$ bit is non-uniformly distributed, $Out_j$ isn’t a balanced function.

3. Set $i_1, \cdots, i_s$ as indices of first $s$ bits of vector $z^*$ where $z^*_j = 0$, $j \in \{i_1, \cdots, i_s\}$.

In summary, we get the values of $i_1, \cdots, i_s$, which determine the construction of puncture function $p_s((z_1, \cdots, z_h)) = (z_{i_1}, \cdots, z_{i_s})$.

Hence, take Algorithm 4 as the preprocessing process, the special puncture functions $p_s(\cdot)$ are constructed by different structures of stream ciphers.

**Preprocessing Algorithm: Special Case** In Algorithm 4, we take the whole initialization vector space into consider. However, it might be highly demanding for stream ciphers because of the existence of weak initialization vectors. Besides, our search model only considers $r$ pairs of initialization vector and punctured keystream. So we can find the indices $i_1, \cdots, i_s$ by confirming the bit-wise distribution of keystream under every selected initialization vectors $v_1, \cdots, v_r$.

Thus, the specific process is as following:

1. Replace the operation of $m$ Hadamard gates with loading $v_i$ in step 2 of Algorithm 4. The quantum circuit of Algorithm 4 in this special case is shown in Figure 7.

2. Run constant times of the changed Algorithm 4 with parameter $v_i$, and get the defined vector $z^* = (z^*_1, \cdots, z^*_h)$. To Avoid confusion, set $z^{*,i} = z^*$.

3. Run step 2 $r$ times with different initialization vector $v_i$ each time, for $i = 1, \cdots, r$. And get $r$ vectors $z^{*,1}, \cdots, z^{*,r}$.

4. Set $z^# = (z^#_1, \cdots, z^#_h)$, where $z^#_j = z^{*,1}_j \lor \cdots \lor z^{*,r}_j$, and $j = 1, \cdots, h$. In the same way, find the first $s$ indices as $i_1, \cdots, i_s$ with components of 0 in vector $z^#$, where $z^*_i = 0$, $j \in \{1, \cdots, s\}$.

![Fig. 7. Quantum Keystream Bit-wise Distribution Algorithm in Special Case](image)

In summary, we get the values of $i_1, \cdots, i_s$ and construct the function $p_s(\cdot)$ under a more loose condition, only suitable for specific initialization vector set $\{v_1, \cdots, v_r\}$. By the way, for the pseudo-randomness property, it’s better not to choose a weak initialization vector.
Key Search Model for Stream Ciphers After run constant times of preprocessing process, with the construction of \( p_s \), function \( g_{v,s} \) is guaranteed to be pseudo-random.

As for the complexity of oracle \( O_{g_{v,s}} \), as same as block ciphers, it is mostly equal to the complexity of encryption in stream ciphers. Let \( \text{Init}(n) \) denote the complexity of full-round initialization in stream ciphers, and \( \text{Output}(n, i_s) \) denote the complexity of outputting \( i_s \) bits keystream (excluding initialization process). Therefore, the complexity of the oracle \( C_{g_{v,s}} = \text{Init}(n) + \text{Output}(n, i_s) \).

The above analysis are based that there are a few bits distributed uniformly on 0-1 space. What if there is no or not enough bits to construct \( p_s \), no matter how large \( h \) is.

There are two ways to answer this question.

On the one hand, as for the construction of stream ciphers, the initialization process of stream cipher is aimed to fully mix the key seed (and optional initialization vector) into some seemingly random initial states as input of outputting process, which in deed we take advantage of. In particular, more random the initial states are, more bits of keystreams under the full key space are uniformly distributed, smaller \( i_1, \ldots, i_s \) can be found out to construct \( p_s \), and lower search complexity it turns out to be.

On the other hand, as for some stream ciphers whose keystreams appear not so random, distinguishing attack must be a huge security threat.

4 Implementation of Multi-round Key Search Model

To evaluate the specific searching complexity on symmetric ciphers, we implement our search model on block cipher AES family, two kinds of stream ciphers Grain-128 and ZUC-128/256.

As for stream ciphers Grain-128 and ZUC-128/256, we need to calculate quantum gates number in circuits of two stream ciphers with the initialization and keystream output process. As for block cipher AES, we have to figure out the complexity of quantum AES’s encryption oracle.

4.1 Stream Cipher Grain-128

Stream Cipher Grain-128 Oracle The Grain-128 algorithm [7] is proposed in 2006, with input of 128-bit key and 96-bit initialization vector, and output of keystream with arbitrary length. The Grain-128 algorithm consists of two processes, 256-round initialization process and keystream output process. The specific components include linear feedback shift registers, nonlinear feedback shift registers, and filter function generators, etc.

The construction of the oracle for Grain-128 is as follows.

1. Initialization process
   The state update formulas on each component during initialization are as follows.
a. Linear feedback shift register (LFSR):

\[ s_{i+128} = s_i + h(x) + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}. \]

b. Non-linear feedback shift register (NFSR):

\[ b_{i+128} = s_i + b_i + h(x) + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84}. \]

c. Filter function:

\[ h(x) = b_{i+12}s_{i+8} + s_{i+13}s_{i+20} + b_{i+95}s_{i+42} + s_{i+60}s_{i+79} + b_{i+12}b_{i+95}s_{i+95}. \]

We implement the Grain-128 algorithm into a quantum circuit by state update formulas on each register, which is simpler and easier than constructed by feedback polynomials. It is known that the number of quantum gates required in one round is 36. And the number of quantum gates within 256 rounds of initialization is

\[ \text{Init}(128) = 9216. \]

2. Keystream output process

Similarly, the state update formulas are as follows:

a. Linear feedback shift register (LFSR):

\[ s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}. \]

b. Non-linear feedback shift register (NFSR):

\[ b_{i+128} = s_i + b_i + h(x) + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84}. \]

c. Filter function:

\[ h(x) = b_{i+12}s_{i+8} + s_{i+13}s_{i+20} + b_{i+95}s_{i+42} + s_{i+60}s_{i+79} + b_{i+12}b_{i+95}s_{i+95}. \]

d. Keystream output:

\[ k_i = h(x) + s_i + s_{i+93} + s_{i+2} + b_i + b_{i+15} + b_{i+36} + b_{i+45} + b_{i+64} + b_{i+73} + b_{i+89}. \]

By the keystream output formula, if the bit \( s \) of punctured keystream required in Algorithm 3 is smaller than 32, it is no need to update states on registers. At this time, the number of gates required to output 1 bit is 18. Else, if \( s > 32 \), there are \( s - 32 \) bits needed to update, and the number of gates required to output 1 bit is 42. Hence, the complexity of outputting \( s \)-bit keystream is

\[ \text{Output}(128, s) = \begin{cases} 
18s & , \\ 42s - 768 & , 
\end{cases} \text{ for } s \leq 32 
\begin{cases} 
18s & , \\ 42s - 768 & , 
\end{cases} \text{ for } s > 32. \]

(11)
Multi-round Key Search Model Effect on Grain-128 For stream cipher Grain-128, the key length $n = 128$. By Table 2, in this case, the optimal solutions are $s = 19$ and $r = 7$. So the values of $i_1, \cdots, i_{19}$ are settled by constant times of Algorithm 4, which means our pseudo-random function $g_{v,19}(k) = p_{19} \circ E_v(k)$ is determined. Hence,

$$C_{g_{v,19}} = \text{Init}(n) + \text{Output}(n, i_{19}) = \begin{cases} 9216 + 18i_{19}, & i_{19} \leq 32 \\ 8448 + 42i_{19}, & i_{19} > 32 \end{cases}.$$

Denote $\delta = 0.01$ as the error tolerance of search algorithm, put it into formula (5), and the complexity is

$$\tilde{O} \left(7 \cdot \log(1/0.01) \cdot 2^9 \cdot C_{g_{v,19}} \right).$$

Hence, according to the quantitative relation between $i_s$ and $s$, the searching complexity against Grain-128 is shown in Table 3. Besides, for the stream cipher property that encryption has to be easy to implement, outputting more bits seems to increase little difficulty on search model.

**Table 3. Searching Complexity of the Stream Cipher Grain-128**

<table>
<thead>
<tr>
<th>$i_s$</th>
<th>$s/i_s$</th>
<th>Searching Complexity</th>
<th>Qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>1</td>
<td>$\tilde{O}(2^{27.77})$</td>
<td>282</td>
</tr>
<tr>
<td>21</td>
<td>0.9</td>
<td>$\tilde{O}(2^{27.77})$</td>
<td>284</td>
</tr>
<tr>
<td>24</td>
<td>0.8</td>
<td>$\tilde{O}(2^{27.78})$</td>
<td>287</td>
</tr>
<tr>
<td>28</td>
<td>0.7</td>
<td>$\tilde{O}(2^{27.79})$</td>
<td>291</td>
</tr>
<tr>
<td>32</td>
<td>0.6</td>
<td>$\tilde{O}(2^{27.80})$</td>
<td>295</td>
</tr>
<tr>
<td>38</td>
<td>0.5</td>
<td>$\tilde{O}(2^{27.84})$</td>
<td>301</td>
</tr>
</tbody>
</table>

4.2 Stream Cipher ZUC-Like

**Stream Cipher ZUC-128 Oracle** The ZUC-128 algorithm [9] is a synchronous stream cipher algorithm with input of a 128-bit key seed and a 128-bit initialization vector, and output of a 32-bit keystream at a time.

The ZUC-128 algorithm consists of two processes.

1. Initialization process

Divide the key $k$ and the initialization vector $IV$ by 8 bits, where $k = k_0 \| k_1 \| \cdots \| k_{15}$, and $IV = IV_0 \| IV_1 \| \cdots \| IV_{15}$. Load them into linear feedback shift registers, where $s_i = k_i \| d_i \| IV_i$, $0 \leq i \leq 15$, and $d_i$ is a 15-bit constant. Set memory unit variables $R_1 = R_2 = 0$, run Initialization process 32 rounds. (The output $W$ of the nonlinear function $F$ needs to round off the last 1 bit to participate in the state update process of the LFSR)

2. Keystream output process
After loading the key, the iterative process of bit-reorganization, nonlinear function, and LFSR state update is first executed in sequence, but no keystream is output. After that, the word keystream output process begins. Every iteration, a 32-bit (one word) keystream \( z = W \oplus X_3 \) is output.

As for the construction of the oracle for ZUC-128, a quantum circuit for the stream cipher ZUC-128 is designed in [14], where a round of initialization process requires 3000 Toffoli gates, 9488 CNOT gates and 736 Pauli X gates, the first round of operations in working mode executed after initialization requires 2754 Toffoli gates, 8849 CNOT gates, 672 Pauli X gates, and a round of operation in working mode (outputs a 32-bit keystream) requires 2754 Toffoli gates, 8913 CNOT gates, and 672 Pauli X gates.

To sum up, the complexity \( \text{Init}(128) \) is 435443 of 32 rounds initialization processes and first round of operations in working mode, and the complexity of outputting 32-bit keystream is \( \text{Output}(128, 32) = 12339 \).

As for the qubits, 496 qubits hold each state on the linear feedback shift register, 64 qubits hold the two memory unit variables \( R_1 \) and \( R_2 \), \( s \) qubits hold the values of the output keystream, and 64 auxiliary qubits count.

**Stream Cipher ZUC-256 Oracle** The ZUC-256 algorithm [24] is a synchronous stream cipher algorithm with input of a 256-bit key seed and an 128-bit initialization vector, and output of a 32-bit keystream at a time.

Compared with ZUC-128, stream cipher ZUC-256 involves three parts, linear feedback shift register, bit-reorganization, and finite state machine, with the same operation rules as ZUC-128.

However, the only difference lies in the key/IV loading scheme of ZUC-256 stream cipher. Every 32-bit state on linear feedback shift register is organized by four parts from a byte component of key or initialization vector or some 7-bit constants. Because the key/IV loading process only takes a constant number of quantum gates, which can be negligible in calculating the total complexity of the oracle. Thus, the complexity of ZUC-256 oracle is \( \mathcal{O}(2^{18.77}) \), same as the ZUC-128 oracle.

**Multi-round Key Search Model Effect on ZUC-128 and ZUC-256** This section we apply our search model to ZUC-128 and ZUC-256.

1. Stream cipher ZUC-128

For stream cipher ZUC-128, the key length \( n = 128 \), the optimal values are \( s = 19 \) and \( r = 7 \). And the values of \( i_1, \cdots, i_{19} \) are settled by constant running times of Algorithm 4, which means our pseudo-random function \( g_{v, 19}(k) = p_{19} \circ E_v(k) \) is determined.

Hence,

\[
C_{g_{v, 19}} = \text{Init}(n) + \text{Output}(n, i_{19}) = 435443 + 12339 \cdot \left\lfloor \frac{i_{19}}{32} \right\rfloor.
\]
Denote $\delta = 0.01$ as the error tolerance of search algorithm, put it into formula (5), and the searching complexity is

$$\tilde{O} \left( 7 \cdot \log \left( \frac{1}{0.01} \right) \cdot 2^9 \cdot C_{g_{v,19}} \right).$$

2. Stream cipher ZUC-256

In the same way, for stream cipher ZUC-256, the key length $n = 256$, and the optimal values are $s = 20$ and $r = 13$. And our pseudo-random function $g_{v,20}(k) = p_{20} \circ E_v(k)$ is determined by constant times of Algorithm 4, so do the values of $i_1, \cdots, i_{20}$.

Hence,

$$C_{g_{v,20}} \approx \text{Init}(n) + \text{Output}(n, i_{20}) = 435443 + 12339 \cdot \left\lceil \frac{i_{20}}{32} \right\rceil.$$ 

Denote $\delta = 0.01$ as the error tolerance of search algorithm, put it into formula (5), and the searching complexity is

$$\tilde{O} \left( 13 \cdot \log \left( \frac{1}{0.01} \right) \cdot 2^{9.5} \cdot C_{g_{v,20}} \right).$$

We compare the search model effect on ZUC-128 and ZUC-256 together in Table 4 on the value of $s$, $r$, $s/\text{words length}$ and searching complexity.

Case 1 is that stream cipher ZUC outputs 1 word keystream, which means there are at least $s$ bits distributed uniformly in the first word keystream. And, case 2 is that stream cipher ZUC outputs 2 words keystream, which means there are at least $s$ bits distributed uniformly in the first two word keystream. As discussed above, we believe for the security of stream cipher, the more random keystream appears, the more security it owns. In other words, the value $s/\text{words length}$ shouldn’t be too small, and that why we only take consider of 1 and 2 words length keystream.

### Table 4. Searching Complexity of the Stream Cipher ZUC-128/256

<table>
<thead>
<tr>
<th>Stream Cipher</th>
<th>$s$</th>
<th>$r$</th>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$s/32$</td>
<td>Complexity</td>
</tr>
<tr>
<td>ZUC-128</td>
<td>19</td>
<td>7</td>
<td>59.4%</td>
<td>$O(2^{34.31})$</td>
</tr>
<tr>
<td>ZUC-256</td>
<td>20</td>
<td>13</td>
<td>62.5%</td>
<td>$O(2^{34.70})$</td>
</tr>
</tbody>
</table>

By Table 4, more bits of keystream and double key length arouse little difficulty in our search model. Firstly, for the stream cipher property that encryption has to be easy to implement, outputting one more words seems to increase no complexity on search model. Secondly, it’s unfortunate that ZUC-256 has a double length of key in ZUC-128, while the security against our search model is barely enhanced. In other words, increasing the key length of stream ciphers doesn’t enhance the security against our key search model.
4.3 Block Cipher AES

Block Cipher AES Oracle The three main kinds in block cipher AES family [19] are AES-128, AES-192, and AES-256, whose key length is 128, 192, and 256 bits, and round is 10, 12, and 14, separately. Besides, all three algorithm encrypt with block length of 128 bits.

Block cipher AES consists of a rounding function and key schedule, based on the substitution-permutation network structure. Firstly, there are three subroutines of a round function, SubBytes, ShiftRows, MixColumns, and AddRoundKey. Secondly, for key schedule, it consists of three subroutines, SubWord, RotWord, and Rcon. In SubBytes and SubWord subroutines, S-box substitution is applied to build up the whole encryption system’s nonlinearity. And in ShiftRows and RotWord subroutines, some particular permutations are implemented by appropriate rewiring. As for MixColumns subroutine, a specific matrix is used to operate the entire column. In AddRoundKey subroutine, the bitwise XOR is operated of the 128-bit roundkey to the internal AES state. At last, Rcon is a round constant.

In [22], the authors design four kinds of quantum circuits for each AES-128/192/256 separately, which can be used as an oracle implemented in Grover’s key search model and our search model as well. And we select one designed quantum circuit for each block ciphers in Table 5.

<table>
<thead>
<tr>
<th>Block Cipher</th>
<th>Qubits</th>
<th>Toffoli</th>
<th>Depth</th>
<th>Total Number of Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128</td>
<td>400</td>
<td>1108</td>
<td></td>
<td>99824</td>
</tr>
<tr>
<td>AES-192</td>
<td>464</td>
<td>1340</td>
<td></td>
<td>115256</td>
</tr>
<tr>
<td>AES-256</td>
<td>528</td>
<td>1540</td>
<td></td>
<td>139919</td>
</tr>
</tbody>
</table>

Multi-round Key Search Model Effect on AES Towards the AES-128, AES-192 and AES-128, the optimal solutions are in Table 2. And by formula (5), the corresponding complexity is

\[ \tilde{O} \left( r \cdot 2^{\frac{r+1}{2}} \cdot \log \left( \frac{1}{\delta} \right) \cdot C_{AES-n} \right), \]

where the complexity \( C_{AES-n} \) of AES \(-\) \( n \) is shown in Table 5.

Combined with Table 2 and Table 5, we get the searching complexity separately on AES-128, AES-192 and AES-256 in Table 6.

It can be concluded from Table 6, AES-256 has the double length of key in AES-128, but the searching complexity just rise slightly. In other words, increasing a key length of block ciphers doesn’t enhance the security against our key search model.
Table 6. Searching Complexity of the Block Cipher AES-128/192/256

<table>
<thead>
<tr>
<th>Block Cipher</th>
<th>Searching Complexity</th>
<th>Qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES-128</td>
<td>(\tilde{O}(2^{31.1}))</td>
<td>425</td>
</tr>
<tr>
<td>AES-192</td>
<td>(\tilde{O}(2^{32.4}))</td>
<td>493</td>
</tr>
<tr>
<td>AES-256</td>
<td>(\tilde{O}(2^{33.0}))</td>
<td>561</td>
</tr>
</tbody>
</table>

5 Conclusion

Our model takes use of the security property, pseudo-randomness, of symmetric ciphers. In other word, main idea of our model is to construct a pseudo-random function \(g_{v,s}: \{0,1\}^n \rightarrow \{0,1\}^s, k \mapsto z\). For block ciphers, the encryption function is thought to be pseudo-random, which guarantees each bit of ciphertext is uniformly distributed. Besides, for stream ciphers, our standpoint is that, more random the initial states appears, more secure stream cipher gets, more bits of keystreams under the full key space distributed uniformly, smaller value \(i_s\) gets, and lower complexity of search model will be. In addition, our search model designs an extra preprocessing process for stream ciphers to construct \(g_{v,s}\).

As for searching cost, the complexity is \(\tilde{O} \left( r \cdot 2^{\frac{m+s}{2}} \cdot \log \left( \frac{1}{\delta} \right) \cdot C_{g_{v,s}} \right)\), where \(n\) denotes key length, \(s\) denotes the length of punctured keystream/ciphertext, \(r\) denotes the iteration in our model, and \(C_{g_{v,s}}\) denotes the complexity of constructed oracle \(O_{g_{v,s}}\). Besides, the required bits of qubits are \(n + m + s + r + q\) for symmetric ciphers, where \(m\) denotes initialization vector/block length, \(r\) denotes the number of auxiliary qubits or the iteration, and \(q\) denotes the number of extra qubits in the quantum symmetric cipher encryption oracle.

According to the parameter selection rules, the searching complexity of our search model outperform Grover algorithm against symmetric ciphers. Furthermore, our search model shows that increasing the key seed length had little influence on the resulting complexity. Thus, this common countermeasure to resist the quantum search attacks does not work anymore in the quantum computation environment. It is necessary to propose new design idea for symmetric ciphers in the future.

References


