Nested 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. This paper puts forward a multi-round quantum key search model for symmetric ciphers. We turn an unstructured key search for symmetric ciphers into a nested structured quantum search. By the preimage of punctured ciphertexts (or keystream), we get a convergent sequence of subspaces of the full key space. In each round, our search is performed only in a subspace containing the real key, while the rest part is removed from search space.
We find out several parameters, the length s of the punctured ciphertext (or keystream), the iteration number r, and the error δ in the search algorithm, which can influence the resulting complexity. The query complexity of our search model is $\tilde{\mathcal{O}}(2^{\alpha_n \cdot n})$, where α_n is much smaller than $\frac{1}{2}$. Specifically, the query complexity is $\tilde{\mathcal{O}}(r2^{\frac{n}{2(r-1)}})$ (ignore constant δ) better than Grover's $\tilde{\mathcal{O}}(2^{\frac{n}{2}})$, while parameter r increases as n increases. Our search model can be applied to symmetric ciphers. And it has been shown that doubling the key length is not an effective way anymore to resist the quantum search attacks. Even if the key length is increased by $r-1$ times, symmetric ciphers still struggle to obtain desired security for a re-selected value of r.

Keywords: Symmetric cipher · post-quantum security · quantum search algorithm.

1 Introduction

Shor's algorithm [20] poses a serious threat to public key cryptographic algorithms based on large number decomposition and discrete logarithm problems. It has forced cryptographers to investigate and design post-quantum secure public-key cryptography algorithms. In the designs of post-quantum secure symmetric ciphers such as stream ciphers and block ciphers, it's generally believed that increasing the key length properly can protect ciphers from quantum attacks, like doubling the key length. Because the unstructured quantum search algorithm, represented by Grover's algorithm [2], is able to achieve a quadratic acceleration of key search for symmetric ciphers. In this way, if search M targets in a set with size of N, then the search complexity is $\tilde{\mathcal{O}}(\sqrt{N/M})$. And this quadratic acceleration has been proved to be optimal [16].

As for structured search, it outperforms unstructured search in the aspect of search complexity. The reason is that, in structured search, an adversary can

identify a part of the search space in advance where the target is located, while abandoning the search in the rest part of the space, which indeed effectively improves the search complexity. And typical structured quantum search models are nested search model[19], highly structured search model[23,24] and so on. Therefore, there is an important issue related to symmetric ciphers' post-quantum security that, whether an unstructured search for symmetric ciphers can be transformed into a structured quantum search model.

A well-designed stream cipher or block cipher can be expressed as a pseudo-random function $F : \{0,1\}^{l_k} \times \{0,1\}^{l_{in}} \rightarrow \{0,1\}^{l_{out}}$ [1], where the inputs are l_k-bit key and l_{in}-bit plaintext (or initialization vector (IV)), and the output is l_{out}-bit ciphertext (or keystream). Thus, breaking stream ciphers or block cipher is equivalent to inverting a pseudo-random function. Or to say, given a ciphertext (or keystream) of sufficient length and the corresponding plaintext (or initial state), the correct key can be found by cryptanalytic techniques. Long-term cryptanalysis has demonstrated that it is difficult to obtain the key for a well-designed pseudo-random function using classical technical methods. In [14], the authors apply Grover oracle in the AES's key search. For detail, given a few 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 [11], the authors optimize the AES attacking quantum circuit based on [14], by searching two pairs of plaintext and ciphertext simultaneously in parallel in a quantum circuit, and prove that AES's quantum security is weaker than 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 [21], the authors design an attacking quantum circuit on stream cipher ChaCha, by outputting a keystream with the same length of key seed, which recovers 256-bit key seed in the quantum circuit with $1.233 \cdot 2^{251}$ quantum gates. In [13], 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.

However, the most important problem we consider is that, whether it is possible to implement structured quantum search in symmetric cipher's key search model, i.e., find out the key $k \in \{0,1\}^{l_k}$ of a pseudo-random function given the corresponding plaintext-ciphertext pairs (or IV-keystream pairs). The structured search we mention here is that, the adversary is able to determine the subset in advance where the key is located, searches in that subset and ignores the remaining part. In essence, that's the reason why structured search can outperform unstructured search in computation complexity. Take the nested quantum search algorithm as an example, unstructured search complexity is $\tilde{\mathcal{O}}(\sqrt{2^{l_k}})$ for a keyspace with size of 2^{l_k}, while nested quantum search complexity is $\tilde{\mathcal{O}}(\sqrt{2^{\alpha l_k}})$ where $\alpha < 1$. Of course, the nested quantum search algorithm is designed for constraint satisfaction problems, and is not applicable to solve the inverse problem of pseudo-random function. Therefore, the key point is that, whether solving the inverse problem of pseudo-random function can be transformed into a structured

search, which effectively reduced search complexity for symmetric cipher. In the following, we will illustrate the structured key search for symmetric cipher, and substitute breaking symmetric cipher for solving the inverse of a pseudo-random function.

Let's imagine a scenario now. For a given plaintext (or IV) v_1, assume that we can get every ciphertext (or keystream) c with every key k in the full key space $K_0 = \{0,1\}^{l_k}$, i.e., $c = E_{v_1}(k)$. Define an s-bit punctured ciphertext (or keystream) z, which is composed of some particular bits of c. Or to say, $z = (c^{(b_1)}, \cdots, c^{(b_s)})$, where $c^{(b_j)}$ is the b_j-th bit of c. For convenience, set punctured function $z = p_s(c)$. Set $Z^{(1)} = \{z = p_s \circ E_{v_1}(k) | k \in K_0\}$ as the punctured ciphertext (or keystream) multi-set. Given a classical chosen-plaintext (or IV) access, an adversary can get the ciphertext (or keystream) c_i with given v_1. In this way, $z_1 = p_s(c)$ can be obtained. For a punctured ciphertext (or keystream) $z_1 \in Z^{(1)}$, there is a preimage set $K_1 = \{(p_s \circ E_{v_1})^{-1}(z_1)\}$, which contains lots of elements for a small value of s. Obviously, $K_1 \subset K_0$. Hence, if one can get the preimage set K_1 by punctured ciphertext (or keystream) z_1, the adversary can narrow the search space to K_1 where real key locates in the second round of search (ignore $K_0 - K_1$). In this way, structured key search is realized. It seems that a larger value of s results in a smaller size of $|K_1|$, which can find the real key k^* in a faster way. However, this viewpoint contradicts our analysis.

In fact, for classical algorithm, the ciphertext (or keystream) set cannot be obtained efficiently when l_k is relatively large, such as $l_k = 128$. What's more, apart from exhaustive classical algorithm, there is no way to acquire K_1 with knowing z_1. However, it's pretty easy to achieve in the quantum computing environment. The set Z_1 can be obtained by designing a quantum punctured encryption oracle, which calculates every punctured ciphertext (or keystream) z with a given plaintext (or IV) v_1 by a uniform superposition of keys k in the full key space K_0. Besides, quantum algorithm can get the preimage K_1 if the measurement of corresponding qubits is z_1. Take Simon's algorithm [25] as an example. Firstly, apply an oracle, which achieves $\frac{1}{\sqrt{N}} \sum_x |x\rangle|0\rangle \to \frac{1}{\sqrt{N}} \sum_x |x\rangle|f(x)\rangle$. And then, measure the second register while the quantum state collapses. If the measurement is $|a\rangle$, then the state on the first register collapses to $\frac{1}{\sqrt{|A|}} \sum_{x \in A} |x\rangle$, where $A = \{x | f(x) = a\}$.

Inspired by Simon's algorithm, we propose a multi-round (r-round) nested search model. An adversary can construct a quantum punctured encryption oracle $\mathcal{O}_{g_{v_i,s}}$ with input of key k and output a flag whether the s-bit punctured ciphertext (or keystream) z is equal to the given z_i, where $z = g_{v_i,s}(k) = p_s \circ E_{v_i}(k)$, and (v_i, z_i) is a given plaintext-ciphertext pair (or IV-keystream pair). And then, construct a search oracle with search algorithm by quantum singular value transformation (QSVT) to amplify the amplitude of the state components with 'good' flag ($z = z_i$). At last, with the measurement of flag, quantum state collapses from a non-uniform superposition of current key space K_{i-1} (after amplitude amplification) to a uniform superposition of the preimage set of z_i, i.e., a subspace $K_i = \{k | g_{v_i,s}(k) = z_i, k \in K_{i-1}\}$. By iteratively applying above operations r times ($i = 1, 2, \cdots, r$), there's of great chance to

obtain the unique and correct key. In this way, during each round, only elements in space K_{i-1} are needed to search, rather than the full key space $K_0 = \{0,1\}^n$.

As for complexity, the query complexity for $\mathcal{O}_{g_{v_i,s}}$ of amplitude amplification in i-th round relies on the target's amplitude $\frac{\sqrt{|K_i|}}{\sqrt{|K_{i-1}|}}$, i.e., the square root of decreasing scale from space K_{i-1} to subspace K_i. We get $\frac{\sqrt{|K_i|}}{\sqrt{|K_{i-1}|}} \approx \frac{1}{2^{\frac{s}{2}}}$, in the first $r-1$ rounds, because of the pseudo-random property owned by function $g_{v_i,s}$ and the pairwise independent property among function $g_{v_1,s}, \cdots, g_{v_r,s}$. And s is the length of punctured ciphertext (or keystream). So the r-round query complexity is $\tilde{\mathcal{O}}(r \cdot 2^{\frac{s}{2}})$. The specific analysis is illustrated in Section 3.

Our search model is specifically designed for the inverse problem of pseudo-random function, and can be viewed as a variant kind of the nested quantum search[19]. The original nested quantum search model in [19] is designed for the constraint satisfaction problem, which cannot be directly used in the problem of finding the key of symmetric ciphers. Take a single-level nested search as example. The unknowns $x_0, \cdots, x_{n-1}$ can be expressed as a quantum state $|x_0 \cdots x_{n-1}\rangle$. At first, design a search oracle for the first s unknowns $x_0, \cdots, x_{s-1}$ to find out every possible solution satisfying the constraints, which is called could-be partial solution. And as for the value of rest $n-s$ parameters, consider the descendants of could-be partial solutions. Because, at the first search stage, solutions contradicting the constraints are filtered out, and only the descendants of could-be partial solutions have chance to be the true solution of the constraint satisfaction problem. The above process can be applied as a unitary and nested into a high-level search model, that's why it is called nested search. Due to the lack of the corresponding constraints in the inverse problem of pseudo-random functions, we cannot determine the first s bits of the key before finding the rest part by the original nested quantum search model. What's more, the authors in [19] have shown that the query complexity decreases as the iteration number r increases. However, they only can determine the query complexity of low-level nested quantum search. For example, about the single-level search, the complexity is $\tilde{\mathcal{O}}(\sqrt{b^{0.618n}})$, which is much better than $\tilde{\mathcal{O}}(\sqrt{b^n})$, the optimal complexity of unstructured search. As for high-level nested quantum search model, it's hard to determine the precise complexity. The reason is that the value of α is not easy to get as a root of a higher degree equation, where the query complexity is $\tilde{\mathcal{O}}(\sqrt{b^{\alpha n}})$ (in this case $b = 2$). Besides, the equation gets more difficult to solve as the number of nested level grows. Comparatively, our search model can give a concrete expression of complexity. Besides, the optimal query complexity can be obtained by obeying the parameter selection rules in Section 3, which is vital to evaluate the attack performance.

The nested search model in paper [19] uses the postponed measurement, which means there is no measurement in the intermediate but at last. However, the search in our model isn't in this way. At the end of each round, we only measure the auxiliary qubit $flag$ instead of other qubits for two purposes. One is that make quantum state collapse from a non-uniform superposition of current key space K_{i-1} to a uniform superposition of the subspace $K_i = \{k | g_{v_i,s}(k) =$

$z_i, k \in K_{i-1}\}$. It helps calculate clearly the specific state in each round and estimate with a more accurate probability. The other is that prevent a spurious key re-entry the search which results in an unpredictable error, for example $k \in \{k|g_{v_1,s}(k) \neq z_1, g_{v_2,s}(k) = z_2, \cdots, g_{v_r,s}(k) = z_r\}$.

The approach to amplitude amplification in this paper is search algorithm by quantum singular value transformation (QSVT) [3,4,5,6,10]. Compared with Grover's algorithm, it can search the target when only known the lower bound of target's amplitude. Besides, it owns the property of convergence, which indeed avoids the problem of 'overcooked'. Although it has larger complexity than Grover's, search algorithm by QSVT still follows the quadratic speed-up. The reason why we adopt search algorithm by QSVT rather than Grover's algorithm is that, in the last search round, there is a volatile scenario where the value of $|K_{r-1}|$ is uncertain but the lower bound of it can be sure. And thanks to the convergence property, amplitude amplification can be achieved when take the lower bound of amplitude as the parameter of search algorithm by QSVT.

The search algorithm by QSVT combines quantum signal processing. And it works as rotations with different phases and different rotary axis. 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. In this way, Grover's algorithm can be seen a special case, where Grover's algorithm works as rotations with a fixed phase in a two-dimensional plane spanned by the target vector and its orthogonal vector. The details will be elaborated in Section 2.

Our search model is designed for the inverse problem of pseudo-random function, which can be used as key search model for symmetric ciphers. We set several parameters, such as, the length of punctured ciphertext (keystream) s, the number of iterations r and the error in the search process δ. There are several constrains among s, r and the key length n of the symmetric ciphers, in order to make our model function well. In this way, we find out the optimal values for s and r when n is settled. For instance, when the key length $n = 128$, we can set $s = 19$ and $r = 7$, which results in the best complexity.

This paper considers two types of complexity, the query complexity and the search complexity. The query complexity is the number of querying quantum punctured encryption oracle in our multi-round search model. And the search complexity is the quantum gate count of our model, which is equal to the product of the query complexity and the gate count of quantum punctured encryption oracle. Therefore, in our model, the query complexity is same toward different symmetric ciphers with the same key length, while the search complexity is different.

To demonstrate the performance of the model, we analyze the complexity of several symmetric ciphers under our attack model. The conclusion shows that, the search complexity are $\tilde{\mathcal{O}}(2^{36.5})$ for AES-128, is $\tilde{\mathcal{O}}(2^{40.3})$ for AES-192, is $\tilde{\mathcal{O}}(2^{41.6})$ for AES-256, $\tilde{\mathcal{O}}(2^{28.3})$ for Grain-128, $\tilde{\mathcal{O}}(2^{33.8})$ for ZUC-128, and $\tilde{\mathcal{O}}(2^{37.8})$ for ZUC-256. We analyze the query complexity for different key length of symmetric ciphers, and show that it increases slower than Grover's algorithm. Notice

that query complexity remains unchanged for different symmetric ciphers with the same key length n. Specifically, for the symmetric cipher with the key length n, the query complexity is $\tilde{\mathcal{O}}(r \cdot 2^{\frac{s}{2}} \cdot \log(1/\delta)) = \tilde{\mathcal{O}}(r \cdot 2^{\frac{n}{2(r-1)}} \cdot \log(1/\delta))$, because r and s satisfy $r = \lfloor \frac{n}{s} \rfloor + 1$, where δ is an error used to estimate the approximation performance in the search algorithm by QSVT, and when $\delta = 0.01$ the approximation is good enough (explained in Section 2). Although $r \cdot 2^{\frac{n}{2(r-1)}} \geq r \cdot 2^{\frac{s}{2}}$, we neglect it thanks to the $\tilde{\mathcal{O}}$ notation. But in the main body, we will take them into consider for accuracy.

It seems increasing key length by $r-1$ times can protect symmetric ciphers from our search model. However, the truth is that r is a parameter depending on key length n rather than a constant. Or to say, set the original key length is n, and new key length is $n(r-1)$. By our search model, there is a relation between n and r, i.e., set $r = R(n)$. Then, for key length $n(r-1)$, let $r' = R(n(r-1))$, and the query complexity is $\tilde{\mathcal{O}}(r' 2^{\frac{n(r-1)}{2(r'-1)}} \log(1/\delta))$, which must be smaller than expected complexity $\tilde{\mathcal{O}}(r \cdot 2^{\frac{n(r-1)}{2(r-1)}} \log(1/\delta)) = \tilde{\mathcal{O}}(r \cdot 2^{\frac{n}{2}} \log(1/\delta))$. For example, $r = 7$ if $n = 128$, and the query complexity is $\tilde{\mathcal{O}}(2^{16.2})$, while $r = 21$ if $n = 1024$, and the query complexity is $\tilde{\mathcal{O}}(2^{32.7})$. More comparisons on complexity can be found in Table 2. Hence, increasing the key length is not an effective way for the post-quantum secure symmetric ciphers.

The rest of this paper is organized as follows. In Section 2, we introduce some symbols and quantum algorithms. Section 3 gives the specific construction of the nested quantum search model for symmetric ciphers. And, we apply the new nested model to analyze the security of stream ciphers Grain, ZUC and block cipher AES family, and list the corresponding query and search complexity in Section 4. Section 5 concludes this paper.

2 Preliminaries

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

Table 1: Symbol Description

Symbol	Representative Meaning
k	key seed
n	key seed length
v_i	plaintext or initialization vector
z	punctured ciphertext or keystream
s	length of punctured ciphertext or keystream
r	iteration number
K_i	key space
E_v	encryption function of symmetric ciphers with vector v
p_s	puncture function
$g_{v,s}$	composite function
$C_{g_{v,s}}$	complexity of oracle $\mathcal{O}_{g_{v,s}}$

2.1 Deutsch-Jozsa Algorithm

Deutsch-Jozsa Algorithm [18] can testify whether a boolean function $f(x)$ is balanced or constant.

Algorithm 1: Deutsch-Jozsa Algorithm

Input: Boolean function $f(x)$.
Output: $f(x)$ is balanced or not.

1: Prepare two quantum registers. The first is a n-qubit register initialized to $|0\rangle$, and the second is a one-qubit register initialized to $|1\rangle$.
$\triangleright\ |0\rangle^{\otimes n}|1\rangle$
2: Apply a Hadamard gate to each qubit. $\triangleright\ \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle\left(|0\rangle-|1\rangle\right)$
3: Apply the quantum oracle $|x\rangle|y\rangle$ to $|x\rangle|y\oplus f(x)\rangle$.
$\triangleright\ \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle\left(|0\rangle-|1\rangle\right)$
4: At this point the second qubit register may be ignored. Apply a Hadamard gate to each qubit in the first register.
$\triangleright\ \frac{1}{2^n}\sum_{x=0}^{2^n-1}\left[\sum_{y=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot y}\right]|y\rangle$
5: Measure the first register. Notice that the probability of measuring $|0\rangle^{\otimes n}=\left|\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}\right|^2$, which evaluates to 1 if $f(x)$ is constant and 0 if $f(x)$ is balanced.

By running constant times of Algorithm 1, the probability of measuring $|0\rangle^{\otimes n}$ can demonstrate the value $\sum_{x=0}^{2^n-1}(-1)^{f(x)}=||\{x|f(x)=1\}|-|\{x|f(x)=0\}||$. In this way, it can tell that whether boolean function $f(x)$ is balanced or not.

2.2 Grover's Algorithm

Grover's algorithm [2] solves the problem of searching some specific elements in the set $S=\{1,\ldots,N\}$. As for the function $f:\{1,\ldots,N\}\to\{0,1\}$, Grover's algorithm can find an element $\{\alpha\}$, for which $f(\alpha)=1$. And for other elements $x\in\{1,\ldots,N\}\backslash\{\alpha\}$, $f(x)=0$.

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

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

where $\left(|\alpha\rangle\langle\alpha|\otimes I\right)|\psi^{\perp}\rangle=0$.

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

$$G^t|\alpha\rangle|\phi\rangle=\sin\left[(2t+1)\,\theta\right]|\alpha\rangle|U(\phi)\rangle+\cos\left[(2t+1)\,\theta\right]|\psi^{\perp}\rangle.$$

We apply above operators to the quantum state $|s\rangle$, which means $|s'\rangle=G|s\rangle$. As is shown in Figure 1(a), the above process can be seen as a rotation of 2θ degree on the two-dimension plane, which is spanned by the superposition state corresponding to the special vector and its orthonormal state.

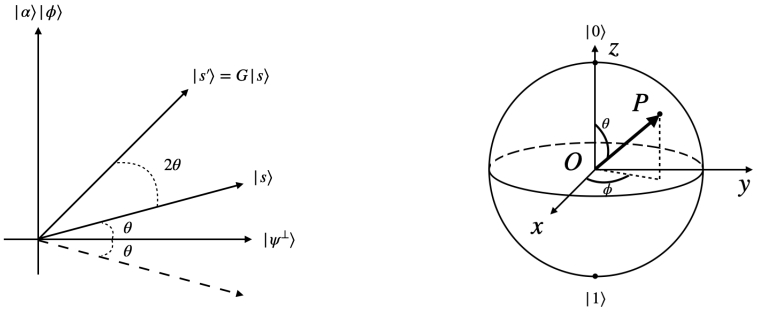
When the goal is searching M elements out of N, then

$$\sin^2(\theta) = M/N, 0 < \theta \leq \pi/2.$$

If $M \ll N$, $\theta \approx \sin(\theta) = \sqrt{M/N}$. We can successfully measure the special vectors $\{\alpha\}$ with probability of $1 - \frac{M}{N}$, after applying $\left\lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rceil$ operator G. Above all, Grover algorithm can achieve quadratic acceleration compared to classical unstructured database search algorithms.

2.3 Quantum Signal Processing

Set a quantum state $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$, where $\theta \in [0, \pi]$, $\phi \in [0, 2\pi]$. The parameters θ and ϕ can locate a point in Bloch sphere [9], as shown in Figure 1(b).



(a) Grover's algorithm (b) Bloch sphere

Fig. 1: Geometric representation

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, $W(a) = \begin{bmatrix} a & i\sqrt{1-a^2} \\ i\sqrt{1-a^2} & a \end{bmatrix}$ is an x-rotation of $\theta = -2\cos^{-1}(a)$ degree, and $S(\phi) = e^{i\phi Z}$ is a z-rotation of -2ϕ degree.

Definition 1. *[10] For a tuple of phases $\overrightarrow{\phi} = (\phi_0, \phi_1, \cdots, \phi_d) \in \mathbb{R}^{d+1}$, the QSP operation sequence $U_{\overrightarrow{\phi}}$ is defined as*

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

Based on the Definition 1, we have the following theorem:

Theorem 1. *[10] The QSP sequence $U_{\overrightarrow{\phi}}$ produces a matrix which may be expressed as a polynomial function of a:*

$$e^{i\phi_0 Z}\prod_{k=1}^{d}(W(a)e^{i\phi_k Z}) = \begin{bmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{bmatrix},$$

for $a \in [-1,1]$, and a $\overrightarrow{\phi}$ 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 $\overrightarrow{\phi}$ that produces a good approximation to any feasible polynomials P and Q.

2.4 Search Algorithm by QSVT

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

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 lower 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 $|A_\perp\rangle = \frac{1}{\mathcal{N}}\left(I - |A_0\rangle\langle A_0|\right)U|B_0\rangle$, where $\mathcal{N}$ 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

$$U = a\left(|A_0\rangle\langle B_0| - |A_\perp\rangle\langle B_\perp|\right) + \sqrt{1-a^2}\left(|A_\perp\rangle\langle B_0| + |A_0\rangle\langle B_\perp|\right),$$

where $|A_0\rangle$, $|A_\perp\rangle$ are left singular vectors, and $|B_0\rangle$, $|B_\perp\rangle$ 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 $1(poly(a) \to 1)$, 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. *[10] Given a unitary U, its inverse $U^\dagger$, and operator $A_\phi = e^{i\phi|A_0\rangle\langle A_0|}$, $B_\phi = e^{i\phi|B_0\rangle\langle B_0|}$,*

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

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 Figure 2(a), the geometry representation of Theorem 2 is as below:

Let $H_A = span(|A_0\rangle, |A_\perp\rangle)$ and $H_B = span(|B_0\rangle, |B_\perp\rangle)$ 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_ϕ works as a rotation around vector $|A_0\rangle$ with certain degree, and the operator B_ϕ 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 vector gradually converge on target vector $|A_0\rangle$ in the way of transforming between two singular vector spaces and rotating around singular vectors.

In Theorem 2, the optimal function for $poly(a)$ in search problem is sign function

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

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

1. $|P^{\Theta}_{\varepsilon,\Delta}(a)| \leq 1$, for $a \in [-1, 1]$.

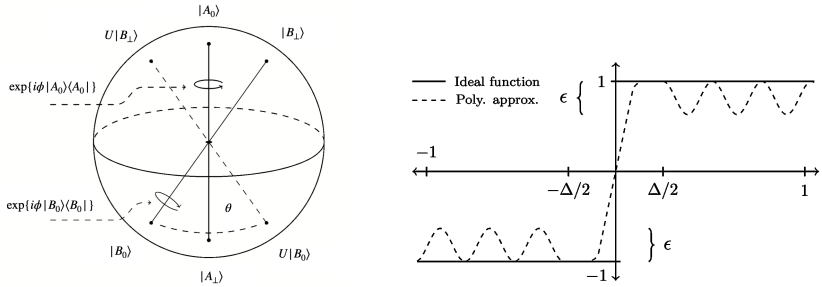
2. $|\Theta(a) - P^{\Theta}_{\varepsilon,\Delta}(a)| \leq \varepsilon$, for $a \in [-1, 1]\backslash(-\frac{\Delta}{2}, \frac{\Delta}{2})$.

All in all, $P^{\Theta}_{\varepsilon,\Delta}(a)$ can ε-approximate sign function $\Theta(a)$, as shown in Figure 2(b).

Let N denote the searching set's size, and $|a| = |\langle A_0|U|B_0\rangle| \geq \frac{1}{\sqrt{N}}$. In Figure 2(b), for an arbitrary value $|a| \geq \frac{1}{\sqrt{N}}$, $P^{\Theta}_{\varepsilon,\Delta}(a) \approx 1$, when $\Delta/2 \leq \frac{1}{\sqrt{N}}$. Thus, we get the Theorem 3.

Theorem 3. *[10] Given unitary operators U, $U^\dagger$, and rotation operators $A_\phi = e^{\mathrm{i}\phi|A_0\rangle\langle A_0|}$, $B_\phi = e^{\mathrm{i}\phi|B_0\rangle\langle B_0|}$,*

$$\langle A_0| \left[\prod_{k=1}^{d/2} U B_{\phi_{2k-1}} U^\dagger A_{\phi_{2k}}\right] U|B_0\rangle = P^{\Theta}_{\varepsilon,\Delta}(a),$$



(a) The geometry representation of Theorem 2

(b) Simulation of sign function in Theorem 3

Fig. 2: Geometry representation and optimal simulation function

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

And the odd polynomial $P^{\Theta}_{\varepsilon,\Delta}(a)$, as an approximation of $\Theta(a)$, satisfies the conditions of $P(a)$ in Theorem 1. At this point, the operator sequence $U_{\vec{\phi}} = \left(P^{\Theta}_{\varepsilon,\Delta}\right)^{(SV)}(W) \approx \Theta^{(SV)}(W)$.

In summary, search algorithm by QSVT is as below:

Algorithm 2: Unstructured Search Algorithm by QSVT[10]

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 $\left(P^{\Theta}_{\delta/2,\Delta}\right)^{(SV)}(W)$, where W is the block encoding of U.

2: Apply $\left(P^{\Theta}_{\delta/2,\Delta}\right)^{(SV)}(W)$ to the uniform superposition. If the auxiliary is measured as $|+\rangle$, then $|m\rangle$ remains in the register. Else, repeat the above process.

Algorithm 2 successes in the probability of at least 1-δ, and costs 1 extra auxiliary qubit with complexity of

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

which obeys well-established quantum lower bounds on the hardness of unstructured search in [16].

3 Nested Quantum Key Search Model for Symmetric Ciphers

A general key search model for symmetric ciphers takes use of the one-to-one mapping of key seed onto ciphertext (or keystream), marks the state component mapping to the given ciphertext (or keystream) with the 'good' label, applies Grover's algorithm with $\mathcal{O}(\sqrt{N/M})$ Grover's operators, and returns the key seed. Usually, $M = 1$, and $N = 2^n$, where n is the key length.

It has been proved that Grover's search algorithm is optimal for unstructured set [16], while nested search can break this threshold and speed up a lot for structured set [19]. So our search model begins with nested search for symmetric ciphers. However, there are two major questions about nested search for symmetric ciphers:

Q1. How to transform an unstructured key search of symmetric ciphers into a structured search?

Q2. How to estimate the number of candidate solutions in every round (level)?

For question 1, we design a nested quantum search model, shown in Figure 3. It is the quantum punctured encryption oracle that helps narrow search space from a key space K_{i-1} (current round) to a subspace K_i (next round), and determine the decreasing scale. By the amplitude amplification, the state components corresponding to the subspace K_i can be filtered from a uniform superposition of K_{i-1}. And thanks to the measurement of flag register, quantum state collapses to an expected state, the uniform superposition of subspace K_i.

Specifically, the adversary can query the quantum oracle $\mathcal{O}_{g_{v_i,s}}$ with key k as variable and v as parameter. As for the composite function $g_{v,s} : \{0,1\}^n \to \{0,1\}^s, k \mapsto z$,

$$g_{v,s}(k) = p_s \circ E_v(k) = p_s(c) = z,$$

mapping a key k to punctured ciphertext z (or keystream), where z is composed of some particular bits in c. (The exact design of $g_{v,s}$ and the definition of punctured ciphertext or keystream will be explained later.) By oracle $\mathcal{O}_{g_{v_i,s}}$, the adversary can get the superposition of punctured ciphertext (or keystream).

Besides, the adversary can query a classical encryption oracle $Enc_{k^*}(\cdot)$ which encrypts every inputting plaintext (or IV) v_i with a secret and unknown key k^*. Given a plaintext (or IV) v_i, the adversary can get the corresponding ciphertext (or keystream) by querying the classical oracle $Enc_{k^*}(\cdot)$. Then, puncture the ciphertext (or keystream) by the function p_s, and get the punctured cipehrtext (or keystream), $z_i = p_s \circ Enc_{k^*}(v_i)$. Here, the adversary can get r pairs $(v_1, z_1), \cdots, (v_r, z_r)$.

Here is our main idea of the nested search model. (Step 4 and 5 can be achieved in oracle $\mathcal{O}_{g_{v_i,s}}$.)

Step 1. Set the full key space $K_0 = \{0,1\}^n$ and $j = 1$.

Step 2. Define function $g_{v_j,s}$ with the given pair (v_j, z_j).

Step 3. Generate the multiset $Z^{(j)} = \{g_{v_j,s}(k) | k \in K_{j-1}\}$ under the superposition state $\frac{1}{|K_{j-1}|} \sum_{k \in K_{j-1}} |k\rangle$.

Step 4. Label z_j with 'good' flag, search in K_{j-1} by amplifying the amplitude of the state components corresponding to z_j, and measure the flag qubit, which results in a collapse of state that state components $\{k|g_{v_j,s}(k) \neq z_j\}$ disappear while $\{k|g_{v_j,s}(k) = z_j\}$ remain.

Step 5. Set the key space $K_j = \{g_{v_j,s}(k) = z_j | k \in K_{j-1}\}$. Obviously, $K_j \subset K_{j-1}$. After Step 4, the state stored on key register becomes a uniform superposition of keys in subspace K_j.

Step 6. Let $j \leftarrow j + 1$. If $j = r + 1$, then quit and return the key (on key register). Else, go to Step 2.

The Step 4 is inspired by Simon's algorithm that, if the flag qubit is measured as 'good', the state components stored on key register only and only are 'good' candidate solutions. Besides, the candidate solutions in each (j-th) round is the key space $K_j = \{g_{v_j,s}(k) = z_j | k \in K_{j-1}\}$.

The complete process of nested search model is shown in Figure 3.

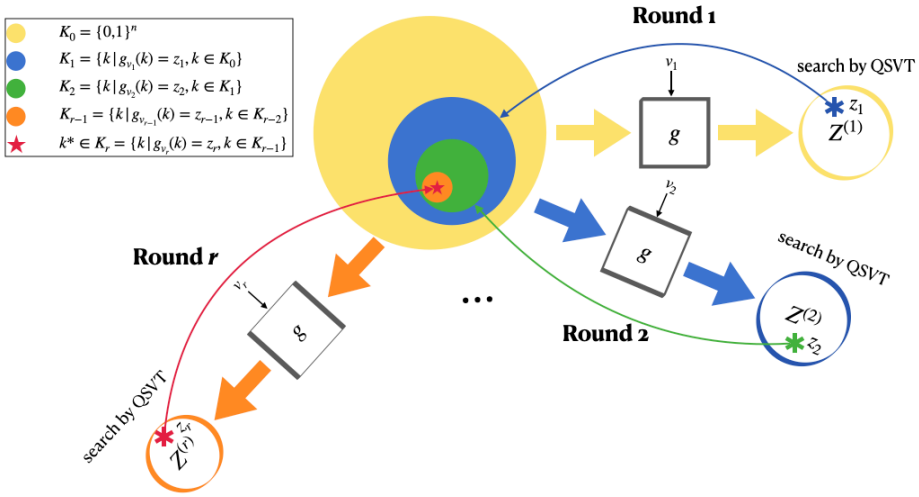


Fig. 3: The Nested Search Model

As for the estimating number of candidate solutions and the approach to amplify the amplitude, the analysis is based on the secure property of symmetric ciphers. More random the ciphertext (or keystream) seems, and more secure the symmetric cipher becomes. Hence, our model is established on the following two recognized assumptions [1].

Assumption 1 *For a strong pseudo-random function $f_m : \{0,1\}^n \to \{0,1\}^s$, with parameter m, a fixed vector (correct key) k^* and an arbitrary vector (key) k, there exists*

$$\Pr_{k \neq k^*} (f_m(k) = f_m(k^*)) = 2^{-s}.$$

Assumption 2 *For a well-designed encryption function f_m with parameter m, if given any r independent parameters $m_1, \cdots, m_r$, then the corresponding functions $f_{m_1}, \cdots, f_{m_r}$ are pairwise independent.*

As discussed above, a pseudo-random function $g_{v,s}(k)$ is designed. And let

$$p = \Pr_{k \neq k^*} \left(g_{v,s}(k) = g_{v,s}(k^*) \right).$$

Then, in the next two subsections, we analyze the correctness of our nested search model separately in two different situations, the ideal one and the practical one. Particularly, the ideal situation is that

$$p = \frac{1}{2^s},$$

while the practical is that

$$\left(\frac{1}{2} - \varepsilon \right)^s \leq p \leq \left(\frac{1}{2} + \varepsilon \right)^s,$$

where $0 \leq \varepsilon \leq \varepsilon_0 < \frac{1}{2}$, and ε_0 can be determined by s.

3.1 An Ideal Nested Key Search Model

Our search model takes advantage of pseudo-randomness (Assumption 1), a property of symmetric ciphers. So, in this section, our nested search model is based on an ideal situation

$$p = \Pr_{k \neq k^*} \left(g_{v,s}(k) = g_{v,s}(k^*) \right) = \frac{1}{2^s}, \tag{2}$$

where $g_{v,s}(k)$ is a pseudo-random function defined as follows.

Pseudo-random Function Construction The construction of the pseudo-random function $g_{v,s}(k)$ contains three steps.

Firstly, we set the encryption function of symmetric cipher,

$$E_v(\cdot) : \{0,1\}^n \to \{0,1\}^h, k \mapsto c,$$

where n represents the key length, k represents the key, and parameter $v \in \{0,1\}^m$. Specifically, for block ciphers, vector v represents a plaintext, vector c represents the ciphertext, h and m represents the block length ($h = m$). As for stream ciphers, vector v represents an initialization vector (IV) with length of m, and vector c represents the h-bit keystream.

Secondly, we design a punctured function $p_s(\cdot) : \{0,1\}^h \to \{0,1\}^s$,

$$p_s(c) = p_s((c^{(1)}, \cdots, c^{(h)})) = (c^{(b_1)}, \cdots, c^{(b_s)}) = z, \tag{3}$$

where $c^{(b_j)}$ is the b_j-th bit of c, $\{b_j|j=1,\cdots,s\}\subset\{1,\cdots,h\}$, and for $1\le u<w\le s$, $b_u<b_w$. In this way, define the punctured ciphertext (or keystream) z, which is punctured by function $p_s(\cdot)$, i.e. $z=p_s(c)$.

At last, we define the composite function $g_{v,s}:\{0,1\}^n\to\{0,1\}^s$,

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

By a delicate selection of $b_1,\cdots,b_s$ (proper design of function p_s), $g_{v,s}$ can be guaranteed as a pseudo-random function. Obviously, the function $g_{v,s}$ can be implemented in the quantum computation environment with the key k open.

The Multi-round Nested Key Search Model As Figure 3 shows, we begin with traversal of the full key space $K_0=\{0,1\}^n$. And set $i=1$. In each round, the state components $k\in K_i=\{k|g_{v_i,s}(k)=z_i,k\in K_{i-1}\}$ is called candidate solutions, and is marked with 'good' flag. Then, we amplify the amplitude of 'good' state components, and measure the auxiliary qubit $flag$ to make the quantum state eventually collapse to the key space K_i. Make $i=i+1$, and iteratively follow the above step, until only the correct key k^* is filtered out.

Hence, we establish the multi-round (r-round) nested key search model (Algorithm 3) with r pairs (v_i,z_i), as a chosen plaintext (or IV) attack. And the quantum circuit is shown in Figure 4. Besides, operator U_i (Algorithm 4) is designed and iteratively applied to achieve the search in each round with input of a pair (v_i,z_i).

Algorithm 3: Multi-round Nested Key Search Algorithm

Input: The r pairs of m-bit vector v_i and s-bit vector z_i, $i=1,\cdots,r$.
Output: The n-bit key.

1: Prepare an initial state stored on register K, and apply n Hadamard gates.
▷ $\frac{1}{\sqrt{2^n}}\sum_{k\in\{0,1\}^n}|k\rangle$
2: Let $i=1$, and full key space $K_0=\{0,1\}^n$.
▷ $\frac{1}{\sqrt{|K_0|}}\sum_{k\in K_0}|k\rangle$
3: Prepare an initial state stored on register $garbage_v$ and $garbage_z$.
▷ $\frac{1}{\sqrt{|K_0|}}\sum_{k\in K_0}|k\rangle\otimes|0\rangle^{\otimes(m+h)}$
4: Apply operator U_i (Algorithm 4) with inputs of vector v_i and z_i (ignore register $flag_i$).
▷ $\frac{1}{\sqrt{|K_i|}}\sum_{k\in K_i}|k\rangle\otimes|0\rangle^{\otimes(m+h)}$, $K_i=\{k|g_{v_i,s}(k)=z_i,k\in K_{i-1}\}$
5: Let $i\leftarrow i+1$. If $i\le r$, go to step 4. Else, measure and return the state in register K.

Here are some details about Algorithm 4.

1. For the value of h, the qubits of register $garbage_z$, $h=m$ for block ciphers, and $h=b_s$ for stream ciphers (the value of b_s is discussed in Section 3.4).

2. As for the design of oracle $\mathcal{O}_{g_{v_i,s}}$, it's shown in Figure 5. Firstly, load v_i on register $garbage_v$, whose complexity is $\tilde{\mathcal{O}}(\log 2^n)=\tilde{\mathcal{O}}(n)$ by [22]. Secondly,

Algorithm 4: Construction of operator U_i

Input: The m-bit vector v_i and s-bit vector z_i.

1: Add one auxiliary qubit on register $flag_i$, and initialize the state to $|-\rangle$.
▷ $\frac{1}{\sqrt{|K_{i-1}|}}\sum_{k\in K_{i-1}}|k\rangle|v_i\rangle|0\rangle^{\otimes s}|-\rangle$

2: Apply the oracle $\mathcal{O}_{g_{v_i,s}}$.
▷ $\frac{1}{\sqrt{|K_{i-1}|}}\sum_{k\in K_i}|k\rangle|0\rangle^{\otimes(m+h)}|+\rangle+\frac{1}{\sqrt{|K_{i-1}|}}\sum_{k\in K_{i-1}-K_i}|k\rangle|0\rangle^{\otimes(m+h)}|-\rangle$

3: Construct an operator $Q_i=\prod_{j=1}^{d/2}\mathcal{O}_{g_{v_i,s}}(I\otimes B_{\phi_{2j-1}})\mathcal{O}^{\dagger}_{g_{v_i,s}}(I\otimes A_{\phi_{2j}})$, where $A_\phi=e^{\mathrm{i}\phi|+\rangle\langle+|}$, $B_\phi=e^{\mathrm{i}\phi|-\rangle\langle-|}$, and phases ϕ_j are given to realize function $P^{\Theta}_{\varepsilon,\Delta}(a)=P^{\Theta}_{\delta/2,2^{-\frac{s}{2}}}(a)$ by Remez-type exchange algorithm.

4: Apply operator Q_i to amplify the amplitude of $|+\rangle$ on register $flag_i$ (ignore register $garbage_v$ and $garbage_z$).
▷ $a_1\sum_{k\in K_i}|k\rangle|+\rangle+a_2\sum_{k\in K_{i-1}-K_i}|k\rangle|-\rangle,\ a_1\to\frac{1}{\sqrt{|K_i|}},\ a_2\to 0$

5: Measure the register $flag_i$. Meanwhile, the quantum state collapses.
▷ $\frac{1}{\sqrt{|K_i|}}\sum_{k\in K_i}|k\rangle\otimes|0\rangle^{\otimes(m+h)}\otimes|+\rangle$

apply encryption oracle of symmetric cipher $E_{v_i}(k)$, which stores $c=g_{v_i,s}(k)$ on register $garbage_z$. And then, bit-flip operator $I-2|z_i\rangle\langle z_i|$ only apply to some particular qubits, i.e., the b_1-th to the b_s-th bits of c (by formula (3)). It costs $\mathcal{O}(\log 2^s)=\mathcal{O}(s)$ gates according to [9]. At last, uncompute the first two steps, aiming for rolling back the state on garbage registers to recover the all-zero state for storing a new vector v and c in next round of search (Algorithm 3). By the way, we don't take the state of qubits in garbage registers into consider anymore later.

3. The operator Q_i in step 4 is designed by Theorem 3 including the value of d and ϕ_j, where the reason why parameter $\Delta=2^{-\frac{s}{2}}$ is analyzed later.

Specifically, set the full key space $K_0=\{0,1\}^n$, and key spaces $K_i=\{k|g_{v_1,s}(k)=z_1,\cdots,g_{v_i,s}(k)=z_i,k\in K_0\},i=1,\cdots,r$. In this way, the transformation among key spaces in each round of Algorithm 3 is

$$K_0\xrightarrow{g_{v_1,s}(\cdot)=z_1}K_1=\{k|g_{v_1,s}(k)=z_1\}\xrightarrow{g_{v_2,s}(\cdot)=z_2}K_2=\{k|g_{v_2,s}(k)=z_2,$$
$$g_{v_1,s}(k)=z_1\}\xrightarrow{g_{v_3,s}(\cdot)=z_3}\cdots\xrightarrow{g_{v_r,s}(\cdot)=z_r}K_r=\{k|g_{v_i,s}(k)=z_i,i=1,\ldots,r\}$$

just as Figure 3 shows.Hence, $K_i=\{k|g_{v_i,s}=z_i,k\in K_{i-1}\},1\le i\le r$. Or to say, $K_0,K_1,\cdots,K_r$ is a convergent sequence of subspace. More importantly, we will prove that K_r contains and only contains correct key in next section.

In particular, set $K_{i-1}\xrightarrow{search}K_i$ as the transformation between key spaces during each round. Besides, there are two parts to complete this transformation: amplitude amplification of the state components $k\in K_i$ (step 4 in Algorithm 4), and elimination of the state components $k\in K_{i-1}-K_i$ (step 5 in Algorithm 4).

1. The amplitude amplification of $k\in K_i$.

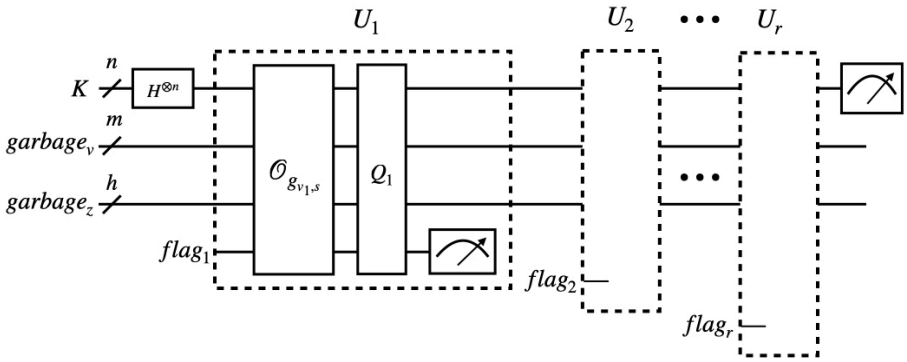


Fig. 4: The Multi-round Nested Key Search Model

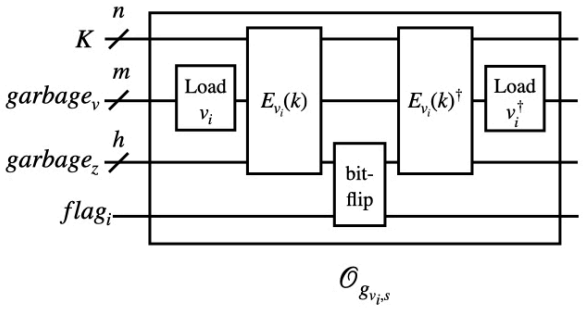


Fig. 5: The Design of Oracle $\mathcal{O}_{g_{v_i,s}}$

Focus on the quantum state before amplitude amplification (step 4 in Algorithm 4) (ignore the garbage registers),

$$|\varphi\rangle = \frac{1}{\sqrt{|K_{i-1}|}} \left(\sum_{k \in K_i} |k\rangle|+\rangle + \sum_{k \in K_{i-1}-K_i} |k\rangle|-\rangle \right).$$

And by the definition of K_i, for any $k \in K_{i-1} - K_i$, $\langle z_i | g_{v_i,s}(k) \rangle = 0$.

Set $|\alpha_i\rangle = \frac{1}{\sqrt{|K_i|}} \sum_{k \in K_i} |k\rangle$ as a uniform superposition state of keys k in key space K_i (candidate solutions), and then

$$|\varphi\rangle = \frac{\sqrt{|K_i|}}{\sqrt{|K_{i-1}|}} |\alpha_i\rangle|+\rangle + \frac{1}{\sqrt{|K_{i-1}|}} \sum_{k \in K_{i-1}-K_i} |k\rangle|-\rangle.$$

By the pseudo-randomness of $g_{v_i,s}$ and formula (2),

$$p = \Pr_{k \neq k^*}(g_{v_i,s}(k) = z_i) = \frac{1}{2^s}. \tag{4}$$

Set a random variable X represents the number of elements in K_i. By formula (4) and Assumption 2,

$$X \sim B\left(|K_{i-1}|, p\right) = B\left(|K_{i-1}|, \frac{1}{2^s}\right),$$

where $E(X) = |K_{i-1}| \cdot p = \frac{|K_{i-1}|}{2^s}$, $D(X) = |K_{i-1}| \cdot p(1-p) = \frac{|K_{i-1}|}{2^s}\left(1 - \frac{1}{2^s}\right)$. For the large $|K_{i-1}|$ and small $p = \frac{1}{2^s}$, X approximately follows the normal distribution $N(E(X), D(X))$. Hence, a random variable Y follows

$$Y = \frac{X}{|K_{i-1}|} \sim N(p, \frac{p\left(1-p\right)}{|K_{i-1}|}) = N(\frac{1}{2^s}, \frac{\frac{1}{2^s}\left(1 - \frac{1}{2^s}\right)}{|K_{i-1}|}). \tag{5}$$

And the 99.74% confidence interval of Y is

$$-3 \leq \frac{Y - E(Y)}{\sqrt{D(Y)}} \leq 3,$$

i.e.,

$$\frac{1}{2^s}\left(1 - 3\sqrt{\frac{2^s - 1}{|K_{i-1}|}}\right) \leq Y = \frac{|K_i|}{|K_{i-1}|} \leq \frac{1}{2^s}\left(1 + 3\sqrt{\frac{2^s - 1}{|K_{i-1}|}}\right).$$

Without loss of generality, let $|K_{i-1}| = 2^{s+h}$. The analysis are based on two different cases, $h \geq 0$ and $h < 0$.

i. If $h \geq 0$, then

$$\frac{1}{2^s}\left(1 - 3\sqrt{\frac{2^s - 1}{|K_{i-1}|}}\right) > \frac{1}{2^s}\left(1 - 3\sqrt{\frac{2^s}{|K_{i-1}|}}\right) = \frac{1}{2^s}\left(1 - \frac{3}{2^{\frac{h}{2}}}\right) \geq \frac{1}{2^{s+2}}$$

holds up with condition that

$$h \geq 4. \tag{6}$$

Thus, if the number of elements in key space $|K_{i-1}|$ satisfies

$$|K_{i-1}| \geq 2^{s+4}, \tag{7}$$

then the search algorithm by QSVT can function well while $2^{-\frac{s+2}{2}}$ is a lower bound of $\sqrt{Y}$.

ii. If $h < 0$, then

$$\frac{1}{2^{s+2}} < \frac{1}{2^s} \leq Y = \frac{|K_i|}{|K_{i-1}|}.$$

In the last searching round, the size of key space K_{r-1} must be smaller than 2^s, and then the percentage $Y = \frac{|K_i|}{|K_{i-1}|} \geq \frac{1}{2^s}$. However, thanks to the convergence of search algorithm by QSVT, it still works well while $2^{-\frac{s+2}{2}}$ is a lower bound of (amplitude) $\sqrt{Y}$.

All in all, if $h \geq 4$ or $h < 0$, then there is at least

$$\Pr(Y \in \left[\frac{1}{2^s}\left(1 - 3\sqrt{\frac{2^s - 1}{|K_{i-1}|}}\right), 1\right]) = 1 - \frac{1}{2}(1 - 0.9974) = 0.9987$$

probability that,

$$\frac{1}{2^{\frac{s+2}{2}}} < \sqrt{\frac{|K_i|}{|K_{i-1}|}}. \tag{8}$$

We take this common lower bound into search algorithm by QSVT, i.e., let

$$\Delta/2 \leq 2^{-\frac{s+2}{2}}$$

into formula (1), and get the query complexity of oracle $\mathcal{O}_{g_{v,s}}$

$$\tilde{\mathcal{O}}\left(1/\Delta \cdot \log\left(1/\delta\right)\right) = \tilde{\mathcal{O}}\left(2^{\frac{s}{2}} \cdot \log\left(1/\delta\right)\right)$$

and one round search complexity

$$\tilde{\mathcal{O}}\left(2^{\frac{s}{2}} \cdot \log\left(1/\delta\right) C_{g_{v,s}}\right), \tag{9}$$

where $C_{g_{v,s}}$ is the complexity of oracle $\mathcal{O}_{g_{v,s}}$.

The above approach to amplitude amplification is search algorithm by QSVT. What if we use Grover's algorithm? We give a detailed analysis in Appendix A. In brief, difficulty arouses by the uncertainty of $|K_{r-1}|$ (in the last round) with the usage of Grover's algorithm. Besides, for the lack of convergence property, Grover's algorithm brings a more complicated analysis than search algorithm by QSVT, especially in the last round. So we adopt the search algorithm by QSVT, rather than Grover's algorithm.

2. The elimination of $k \in K_{i-1} - K_i$.

After the amplitude amplification (step 4 in Algorithm 4), we get the quantum state

$$\begin{aligned}|\varphi'\rangle &= a_1 \sum_{k \in K_i} |k\rangle|+\rangle + a_2 \sum_{k \in K_{i-1} - K_i} |k\rangle|-\rangle \\ &= \left(a_1\sqrt{|K_i|}\right)\left(\frac{1}{\sqrt{|K_i|}} \sum_{k \in K_i} |k\rangle\right)|+\rangle + a_2 \sum_{k \in K_{i-1} - K_i} |k\rangle|-\rangle,\end{aligned}$$

where $a_1 \to \frac{1}{\sqrt{|K_i|}}$ and $a_2 \to 0$.

So it's of nearly 1 probability to measure $|+\rangle$ on register $flag_i$ thanks to amplitude amplification. As Simon's algorithm shows, the measurement of register $flag_i$ triggers the collapse of quantum state on register K, V and Z, which means only the state components corresponding to $|+\rangle$ remains. In this way, the state on registers after measurement is

$$|\varphi''\rangle = \frac{1}{\sqrt{|K_i|}} \sum_{k \in K_i} |k\rangle|+\rangle.$$

The major purpose of measurement on register $flag_i$ is to avoid that the wrong keys re-entry the following search rounds, for example, $k_0 \in \{k|g_{v_1,s}(k) \neq z_1, g_{v_2,s}(k) = z_2, \cdots, g_{v_r,s}(k) = z_r\}$. Without measurement, in the last $r-1$ rounds, the amplitude of k_0 should be amplified, which may generate an unpredictable error. Hence, it is needed to prepare a new auxiliary qubit in each round.

Parameter Selection In this section, we elaborate the selection rules of parameter s and r to accomplish two goals. The first one is that search algorithm by QSVT can work well with parameter $\Delta = 2^{-\frac{s}{2}}$. And the second one is that the returning key $k \in K_r$ is unique and correct.

1. Make sure search algorithm by QSVT work well during each round.

In previous, we take $2^{-\frac{s+2}{2}}$ as the common lower bound of target's amplitude with the condition of formula (7).

Let's analyze the last but one round, $K_{r-2} \xrightarrow{search} K_{r-1}$, where $|K_{r-2}| > 2^s > |K_{r-1}|$. Obeying the formula (7), we have to make sure

$$|K_{r-2}| \geq 2^{s+4}.$$

Noticed that in the $r-2$-th round, $K_{r-3} \xrightarrow{search} K_{r-2}$. Because of formula (8),

$$|K_{r-2}| = |K_0| \frac{|K_1|}{|K_0|} \cdots \frac{|K_{r-2}|}{|K_{r-3}|} > |K_0| \left(\frac{1}{2^{s+2}}\right)^{r-2} = 2^{n-(s+2)(r-2)}.$$

In this way, $n-(s+2)(r-2) \geq s+4$, i.e.,

$$n - sr + s - 2r \geq 0, \tag{10}$$

which means if parameter s and r satisfy the formula (10), the search algorithm by QSVT can work well with parameter $\Delta = 2^{-\frac{s}{2}}$ in each round.

2. Guarantee the returned key is unique and correct ($K_r = \{k^*\}$).

According to paper [11], we give the following analysis to make sure that the final measurement is the correct key.

Let 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, 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 s of vector z.

By the pseudo-randomness of function $g_{v,s}(\cdot)$,

$$\Pr_{k' \neq k^*}(g_{v,s}(k') = g_{v,s}(k^*)) = \frac{1}{2^s}.$$

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

$$r^2 \ll 2^s, \tag{11}$$

we have

$$p' = \prod_{j=0}^{r-1} \frac{1}{2^s - j} \approx 2^{-rs}. \tag{12}$$

Set the spurious key set as $SK = \{k' | k' \neq k^*, g_{v_i,s}(k^*) = g_{v_i,s}(k'), i = 1, \cdots, r\}$. By formula (12), $|SK| \approx (2^n - 1)2^{-rs}$.

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

$$\Pr(W = \alpha) = C_{2^n-1}^{\alpha}(p')^{\alpha}(1 - p')^{2^n - 1 - \alpha}.$$

By formula (12),

$$\Pr(W = 0) = C_{2^n-1}^{0}(p')^0(1 - p')^{2^n-1} = (1 - p')^{2^n-1} \approx e^{-2^{n-rs}}.$$

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

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

$$r = \lfloor \frac{n}{s} \rfloor + 1. \tag{13}$$

And then,

$$\frac{n}{r} < s \leq \frac{n}{r-1}. \tag{14}$$

Thus, if the parameters r and s satisfy the formulas (11) and (14), we can guarantee that the probability of returning a unique key is $e^{-2^{n-rs}}$.

Consequently, combined with formulas (10), (11) and (13), we get the selection rules

$$\begin{cases} n - sr + s - 2r \geq 0 \\ r^2 \ll 2^s \\ r = \lfloor \frac{n}{s} \rfloor + 1 \end{cases}. \tag{15}$$

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

$$n + m + h + r + q$$

qubits, and by formula (9) and formula (13), the total complexity is

$$\tilde{\mathcal{O}}\left(r \cdot 2^{\frac{s}{2}} \cdot \log(1/\delta) C_{g_{v,s}}\right) = \tilde{\mathcal{O}}\left(r \cdot 2^{\frac{n}{2(r-1)}} \cdot \log(1/\delta) C_{g_{v,s}}\right), \tag{16}$$

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

Table 2: Query Complexity with Different Key Length

n	s	r	$\tilde{\mathcal{O}}\left(r \cdot 2^{\frac{s}{2}} \log(1/\delta)\right)$	$\tilde{\mathcal{O}}\left(r \cdot 2^{\frac{n}{2(r-1)}} \log(1/\delta)\right)$	Grover's
128	19	7	$\tilde{\mathcal{O}}\left(2^{15.0}\right)$	$\tilde{\mathcal{O}}\left(2^{16.2}\right)$	$\tilde{\mathcal{O}}\left(2^{64}\right)$
192	25	8	$\tilde{\mathcal{O}}\left(2^{18.2}\right)$	$\tilde{\mathcal{O}}\left(2^{19.4}\right)$	$\tilde{\mathcal{O}}\left(2^{96}\right)$
256	26	10	$\tilde{\mathcal{O}}\left(2^{19.1}\right)$	$\tilde{\mathcal{O}}\left(2^{20.3}\right)$	$\tilde{\mathcal{O}}\left(2^{128}\right)$
512	37	14	$\tilde{\mathcal{O}}\left(2^{25.0}\right)$	$\tilde{\mathcal{O}}\left(2^{26.2}\right)$	$\tilde{\mathcal{O}}\left(2^{256}\right)$
1024	49	21	$\tilde{\mathcal{O}}\left(2^{31.6}\right)$	$\tilde{\mathcal{O}}\left(2^{32.7}\right)$	$\tilde{\mathcal{O}}\left(2^{512}\right)$
2048	71	29	$\tilde{\mathcal{O}}\left(2^{43.1}\right)$	$\tilde{\mathcal{O}}\left(2^{44.2}\right)$	$\tilde{\mathcal{O}}\left(2^{1024}\right)$

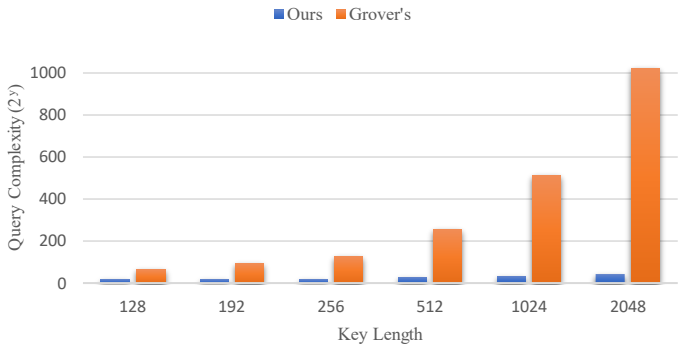


Fig. 6: Query Complexity of Our Search Model and Grover's Algorithm

Take six values of n, the length of key, and we get the optimal value of parameters and corresponding query complexity of $O_{g_{v,s}}$ (formula (16)) in Table 2, where $\delta = 0.01$.

To demonstrate the rationality of formula (16), for six different n and the corresponding r and s, we give the query complexity of $\tilde{\mathcal{O}}\left(r \cdot 2^{\frac{s}{2}} \log(1/\delta)\right)$ and $\tilde{\mathcal{O}}\left(r \cdot 2^{\frac{n}{2(r-1)}} \log(1/\delta)\right)$ respectively. In Table 2, the value of complexity in the 4-th column is less than the corresponding value in the 5-th column. Hence, it is rational to use $r \cdot 2^{\frac{n}{2(r-1)}} \log(1/\delta)$ instead of $r \cdot 2^{\frac{s}{2}} \log(1/\delta)$ in formula (16).

What's more, the error $\delta = 0.01$ is effective in the search algorithm by QSVT, making the polynomial function have a good approximation to the sign function.

By the way, the value of $\log(1/\delta)$ has little impact on the query complexity when $\delta = 0.01$, i.e., there is a difference of 2.73 in the exponential of $r \cdot 2^{\frac{n}{2(r-1)}}$ and $r \cdot 2^{\frac{n}{2(r-1)}} \log(1/\delta)$. Hence, we can use $r \cdot 2^{\frac{n}{2(r-1)}}$ to denote the query complexity, ignoring a constant of error δ. The query complexity of Grover's algorithm increases exponentially with the key length n, while the query complexity in our model increases slower, which can be found in Figure 6. Implied from Table 2 and Figure 6, increasing the key length of symmetric ciphers can hardly resist our quantum search algorithm. In other words, increasing the key length doesn't enhance the security against our search model as desired to be.

3.2 A Practical Nested Key Search Model

In this section, we analyze the situation where bias exists,

$$p = \Pr_{k \neq k^*} \left(g_{v,s}(k) = g_{v,s}(k^*)\right) \in \left[\left(\frac{1}{2} - \varepsilon\right)^s, \left(\frac{1}{2} + \varepsilon\right)^s\right],$$

where $0 \leq \varepsilon \leq \varepsilon_0 < \frac{1}{2}$. Still, we keep the selection rules (formula (15)), and figure out an expression of ε_0 about the parameter s, making our nested key search model function well.

When design an encryption function, bias can't be eliminated but diminished. In this way, we want to figure out the applicable confines of our model by finding the maximum of ε_0. Given Theorem 4, we can settle this problem, i.e., value of ε_0. The proof is in Appendix B.

Theorem 4. *Given r pairs (v_i, z_i), $i = 1, \cdots, r$, the probability p_i follows*

$$p_i = \Pr_{k \neq k^*} \left(g_{v_i,s}(k) = g_{v_i,s}(k^*)\right) \in \left[\left(\frac{1}{2} - \varepsilon\right)^s, \left(\frac{1}{2} + \varepsilon\right)^s\right],$$

where $0 \leq \varepsilon \leq \varepsilon_0$. Meeting the parameter selection rules (formula (15)), the r-round nested key search model can function well with parameter $\Delta = 2^{-\frac{s}{2}}$ and return the key seed with probability of at least an expected value u if

$$\varepsilon_0 = \min\left(\sqrt[rs]{-\frac{\ln u}{2^n - 1}} - \frac{1}{2}, \frac{1}{2} - \sqrt[s]{\frac{17 + 3\sqrt{25 - 4 \cdot 2^{-s}}}{2^{s+5} + 18}}\right). \tag{17}$$

Consequently, according to Table 2, take $\varepsilon = \frac{1}{2} - \sqrt[s]{\frac{17+3\sqrt{25-4\cdot 2^{-s}}}{2^{s+5}+18}}$ into formula (32), and get probability of returning the correct key, as shown in Table 3.

3.3 Quantum Key Search Model for Ideal Ciphers

Set the encryption function of ideal cipher

$$E_v : \{0,1\}^n \to \{0,1\}^h, \ k \mapsto c,$$

Table 3: Nested Search Effect If Bias Exists

Key length n	Punctured length s	Iteration number r	ε	Probability
128	19	7	2^{-25}	0.96
192	25	8	2^{-32}	0.99
256	26	10	2^{-33}	0.94

where k denotes the n-bit key, v denotes the m-bit plaintext (or IV), and c denotes corresponding h-bit ciphertext (or keystream). For an ideal cipher, function $E_v(\cdot)$ is a pseudo-random function.

Set the puncture function

$$p_s(c) = p_s\left((c^{(1)}, \cdots, c^{(h)})\right) = (c^{(1)}, \cdots, c^{(s)}),$$

where $s \leq h$. In this way, we define the first s bits of ciphertext (or keystream) as punctured ciphertext (or keystream).

Define a composite function $g_{v,s} : \{0,1\}^n \to \{0,1\}^s$,

$$g_{v,s}(k) = p_s \circ E_v(k) = p_s(c) = z.$$

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 key k to the punctured ciphertext (or keystream) 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 operations for realizing puncture function p_s only take negligible constant gates. And the bit-flip operator $I - 2|z_i\rangle\langle z_i|$ is of $\tilde{\mathcal{O}}(\log 2^s) = \tilde{\mathcal{O}}(s)$ complexity according to [9], which is negligible. Hence, the complexity $C_{g_{v,s}}$ of oracle $O_{g_{v,s}}$ is almost equal to the complexity C_{cipher} of encryption in ideal ciphers, i.e., $C_{g_{v,s}} = C_{cipher}$.

3.4 Quantum Key Search Model for Practical Ciphers

Different from ideal ciphers, there maybe no strong proof that the encryption function of practical cipher is a pseudo-random function. So the construction of puncture function p_s is our main problem, which extracts out some special bits of ciphertext (or keystream) to make $g_{v,s}$ a pseudo-random function. Take stream ciphers as an example.

Set h boolean functions

$$Out_j(k, v) : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}, (k, v) \mapsto c^{(j)},$$

mapping of the key k and initialization vector v onto the j-th bit of keystream c, for a large enough h, and $j = 1, \cdots, h$.

Consider a practical situation where bias exist, i.e., set

$$\Pr(Out_j(k, v_i) = 1) = \frac{1}{2} + \varepsilon_{i,j}, \Pr(Out_j(k, v_i) = 0) = \frac{1}{2} - \varepsilon_{i,j}, \tag{18}$$

where $0 \leq \varepsilon_{i,j} \leq \frac{1}{2}$. And in this way,

$$\left(\frac{1}{2} - \max_{b_j}(\varepsilon_{i,b_j})\right)^s \leq \Pr(g_{v_i,s}(k) = z_i) \leq \left(\frac{1}{2} + \max_{b_j}(\varepsilon_{i,b_j})\right)^s, \tag{19}$$

where $z_i = (c_i^{(b_1)}, \cdots, c_i^{(b_s)})$, $j = 1, 2, \cdots, s$.

As proved above, before utilizing our search model, it's needed to verify that whether this encryption function of stream cipher can meet our requirement

$$0 \leq \max_{i \in \{1,\cdots,r\}, j \in \{1,\cdots,s\}} (\varepsilon_{i,b_j}) \leq \varepsilon_0. \tag{20}$$

That's why we set a preprocessing procedure.

Preprocessing Procedure We design a quantum keystream bit-wise distribution algorithm to solve the problem, as Algorithm 5 shows. And, by Theorem 4, the main purpose of Algorithm 5 is to find out the boolean function $Out_j(\cdot, v_i)$ which meets

$$0 \leq \varepsilon_{i,j} \leq \varepsilon_0. \tag{21}$$

The preprocessing procedure, quantum keystream bit-wise distribution algorithm, is based on Deutsch-Jozsa Algorithm (Algorithm 1) [18]. By running constant times of Algorithm 1, the probability of measuring $|0\rangle^{\otimes n}$ can demonstrate the value $\sum_{x=0}^{2^n-1}(-1)^{f(x)} = ||\{x|f(x) = 1\}| - |\{x|f(x) = 0\}||$.

However, in the situation (formula (18)), the probability to measure $|0\rangle^{\otimes n}$ is

$$p_j = \left|\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{Out_j(k,v_i)}\right|^2 = \left|\frac{|Out_{i,j}^{-1}(1)| - |Out_{i,j}^{-1}(0)|}{2^n}\right|^2 = (2\varepsilon_{i,j})^2,$$

where $Out_{i,j}^{-1}(1) = \{k|Out_j(k, v_i) = 1\}$, $Out_{i,j}^{-1}(0) = \{k|Out_j(k, v_i) = 0\}$. For the convenience, set function $Out_{i,j}(k) = Out_j(k, v_i)$.

Without search algorithm, it's difficult to measure $|0\rangle^{\otimes n}$ for a small $\varepsilon_{i,j}$, let alone to verify whether $\varepsilon_{i,j} \leq \varepsilon_0$. In this way, we still use search algorithm by QSVT to amplify the amplification of $|0\rangle^{\otimes n}$. Select the value of parameter $\Delta/2 = 2\varepsilon_0$. Set a as the amplitude of $|0\rangle^{\otimes n}$ and a' as the amplitude after amplification, i.e., $a = 2\varepsilon_{i,j}$. Then, by the convergence of search algorithm by QSVT as Figure 2(b) shows,

$$|a'|^2 \approx \begin{cases} 1 & a \geq 2\varepsilon_0 \\ 0 & a < 2\varepsilon_0. \end{cases}$$

Consequently, after amplification, if we measure $|0\rangle^{\otimes n}$, the function $Out_{i,j}(\cdot)$ contradicts the formula (21). Else, the function $Out_{i,j}(\cdot)$ meets the formula

(21), and with given IV v_i, the j-th bit of keystream has the chance to be a part of punctured keystream. In particular, given r IVs, $v_1, \cdots, v_r$, if functions $Out_{1,j}(\cdot), \cdots, Out_{r,j}(\cdot)$ all meet the formula (21), then the j-th bit of keystream has chance to be a bit of punctured keystream.

Algorithm 5: Quantum Keystream Bit-wise Distribution Algorithm

Input: Boolean function $Out_j(k, v_i)$ with given v_i, and bias threshold ε_0.

Output: Whether $\varepsilon_{i,j} \leq \varepsilon_0$.

1: Prepare two quantum registers. The first is an n-qubit register K initialized to $|0\rangle^{\otimes n}$, and the second is a one-qubit register Z initialized to $|1\rangle$.

▷ $|0\rangle^{\otimes n}|1\rangle$

2: Apply a Hadamard gate to each qubit in register K and register Z.

▷ $\frac{1}{\sqrt{2^{n+1}}} \sum_{k=0}^{2^n-1} |k\rangle (|0\rangle - |1\rangle)$

3: Apply the quantum oracle $|k\rangle|z\rangle$ to $|k\rangle|z \oplus Out_j(k, v_i)\rangle$.

▷ $\frac{1}{\sqrt{2^{n+1}}} \sum_{k=0}^{2^n-1} (-1)^{Out_j(k,v_i)} |k\rangle (|0\rangle - |1\rangle)$

4: At this point, register Z can be ignored. Apply a Hadamard gate to each qubit in the first register K.

▷ $\frac{1}{2^n} \sum_{k=0}^{2^n-1} \left[\sum_{y=0}^{2^n-1} (-1)^{Out_j(k,v_i)} (-1)^{k \cdot y} \right] |y\rangle$

5: Add an auxiliary qubit in register $flag$, and initialize to $|-\rangle$. Apply a bit-flip operator $O_f = I^{\otimes n} - 2|0\rangle\langle 0|$ to the n bits on register K.

▷ $\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{Out_j(k,v_i)} |0\rangle^{\otimes n}|+\rangle + \frac{2^n-1}{2^n} |\psi\rangle|-\rangle, \langle\psi| (|0\rangle^{\otimes n}) = 0$

6: Apply the search algorithm by QSVT to amplify the amplitude of flag $|+\rangle$ with parameter $\Delta/2 = 2\varepsilon_0$.

7: Measure register K. If the measurement is $|0\rangle^{\otimes n}$, then $\varepsilon_{i,j} > \varepsilon_0$ and $Out_j(k, v_i)$ can't meet our requirement. Else, $\varepsilon_{i,j} < \varepsilon_0$ and $Out_j(k, v_i)$ can meet our requirement.

Algorithm 5 can only verify the j-th bit of keystream under a given IV v_i. And our nested search model requires r pairs of IV and s-bit punctured keystream $z = (c^{(b_1)}, c^{(b_2)}, \cdots, c^{(b_s)})$, where the bits $\{b_1, \cdots, b_s\}$ are selected by Algorithm 5. Hence, the specific preprocessing procedure is as following:

Pre. Set a large enough constant h.

Step 1. Let $i \leftarrow 1, count \leftarrow 0$, and set $A \leftarrow \emptyset$.

Step 2. Run Algorithm 5 to testify r functions $Out_i(k, v_1), \cdots, Out_i(k, v_r)$. If r functions all meet the formula (21), then $count \leftarrow count + 1$, $i \leftarrow i + 1$, and $A \leftarrow A \cup \{i\}$. Else, let $i \leftarrow i + 1$.

Step 3. Repeat Step 2 until $count = s$ or $i = h + 1$.

Step 4. If $count = s$, return the set A. Else if $i = h + 1$, this stream cipher is out of our nested search model's applicable confines.

In summary, we get the values of $b_1, \cdots, b_s \in A$, which determine the construction of puncture function

$$z = p_s(c) = p_s((c^{(1)}, \cdots, c^{(h)})) = (c^{(b_1)}, \cdots, c^{(b_s)}).$$

Hence, take Algorithm 5 as the preprocessing procedure. And the special puncture function $p_s(\cdot)$ is constructed via different structure of stream cipher. And it's guaranteed that

$$p_i = \Pr_{k \neq k^*} \left(g_{v_i,s}(k) = z_i\right) \in \left[\left(\frac{1}{2} - \varepsilon_0\right)^s, \left(\frac{1}{2} + \varepsilon_0\right)^s\right],$$

where the value of ε_0 is in formula (17), $i = 1, \cdots, r$.

Key Search Model for Stream Ciphers After running preprocessing procedure, construct function p_s and function $g_{v,s}$ which meets the condition in Theorem 4.

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

The search model is based that there are a few bits distributed almost uniformly on 0-1 space ($\varepsilon < \varepsilon_0$). What if there is no or not enough bits to construct $p_s(\cdot)$, no matter how large h is?

There are two points to answer this question:

On the one hand, as for the design of stream ciphers, the initialization process of stream cipher is aimed to fully mix the key seed (and optional initialization vector) into a seemingly random initial state, as an input of outputting process. In particular, more random the initial states are, more bits of keystreams under the full key space are uniformly distributed, and smaller value of $b_1, \cdots, b_s$ can be found out to construct p_s, resulting in a lower search complexity.

On the other hand, as for some stream ciphers whose keystreams appear not so random, i.e., $\varepsilon > \varepsilon_0$, 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 the number of quantum gates in two stream ciphers' circuits 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.

i. Linear feedback shift register(LFSR):

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

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

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

iii. 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

$$Init(128) = 9216. \tag{22}$$

2. Keystream output process

Similarly, the state update formulas are as follows:

i. Linear feedback shift register(LFSR):

$$s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}.$$

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

$$\begin{aligned} b_{i+128} = s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + \\ b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84}. \end{aligned}$$

iii. Filter function:

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

iv. Keystream output:

$$k_i = h(x) + s_{i+93} + b_{i+2} + b_{i+15} + b_{i+36} + b_{i+45} + b_{i+64} + b_{i+73} + b_{i+89}.$$

By the keystream output formula, if the parameter b_s calculated by Algorithm 5 is smaller than 32, it is no need to update states on registers. At this time, the number of gates to output 1 bit is 18. Else, if $b_s > 32$, there are $b_s - 32$ bits needed to update, and the number of gates to output 1 bit is 42. Hence, the complexity of outputting s-bit keystream is

$$Output(128, b_s) = \begin{cases} 18b_s \ , & b_s \leq 32 \\ 42b_s - 768 \ , & b_s > 32 \end{cases}. \tag{23}$$

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 value of parameters are $s = 19$ and $r = 7$.

So the values of $b_1, \cdots, b_{19}$ are settled by preprocessing procedure (Algorithm 5), which means our pseudo-random function $g_{v,19}(k) = p_{19} \circ E_v(k)$ is determined if formula (19) and (20) are met. Hence,

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

Denote $\delta = 0.01$ as the error tolerance of search algorithm, put it into formula (16), and the complexity is

$$\tilde{\mathcal{O}}\left(7 \cdot \log(1/0.01) \cdot 2^{9.5} \cdot C_{g_{v,19}}\right) = \tilde{\mathcal{O}}\left(2^{15.04} \cdot C_{g_{v,19}}\right).$$

Hence, according to the quantitative relation between b_s and s, the searching complexity against Grain-128 is shown in Table 4. 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 4: Searching Complexity of the Stream Cipher Grain-128

b_{19}	s/b_{19}	Searching Complexity	Qubits
19	1	$\tilde{\mathcal{O}}(2^{28.27})$	282
21	0.9	$\tilde{\mathcal{O}}(2^{28.27})$	284
24	0.8	$\tilde{\mathcal{O}}(2^{28.28})$	287
28	0.7	$\tilde{\mathcal{O}}(2^{28.29})$	291
32	0.6	$\tilde{\mathcal{O}}(2^{28.30})$	295
38	0.5	$\tilde{\mathcal{O}}(2^{28.34})$	301

4.2 Stream Cipher ZUC-Like

Stream Cipher ZUC-128 Oracle The ZUC-128 algorithm [8] is a synchronous stream cipher algorithm with input of an 128-bit key seed and an 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 [12], 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 $Init(128)$ is 435443 of 32 rounds initialization processes and first round of operations in working mode, and the complexity of outputting 32-bit keystream is $Output(128, 32) = 12339$.

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 punctured keystream, and 64 auxiliary qubits count.

Stream Cipher ZUC-256 Oracle The ZUC-256 algorithm [17] 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 $\tilde{\mathcal{O}}\left(2^{18.77}\right)$, 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 $b_1, \cdots, b_{19}$ are settled by preprocessing procedure (Algorithm 5), which means our pseudo-random function $g_{v,19}(k) = p_{19} \circ E_v(k)$ is determined if formula (19) and (20) are met.

Hence,

$$C_{g_{v,19}} = Init(n) + Output(n, b_{19}) = 435443 + 12339 \cdot \lceil \frac{b_{19}}{32} \rceil.$$

Denote $\delta = 0.01$ as the error tolerance of search algorithm, put it into formula (16), and the searching complexity is

$$\tilde{\mathcal{O}}\left(7 \cdot \log(1/0.01) \cdot 2^{9.5} \cdot C_{g_{v,19}}\right) = \tilde{\mathcal{O}}\left(2^{15.04} \cdot C_{g_{v,19}}\right).$$

2. Stream cipher ZUC-256

In the same way, for stream cipher ZUC-256, the key length $n = 256$, and the optimal values are $s = 26$ and $r = 10$. And our pseudo-random function $g_{v,26}(k) = p_{26} \circ E_v(k)$ is determined by preprocessing procedure (Algorithm 5), so do the values of $b_1, \cdots, b_{26}$, if formula (19) and (20) are met.

Hence,

$$C_{g_{v,26}} \approx Init(n) + Output(n, b_{26}) = 435443 + 12339 \cdot \lceil \frac{b_{26}}{32} \rceil.$$

Denote $\delta = 0.01$ as the error tolerance of search algorithm, put it into formula (16), and the searching complexity is

$$\tilde{\mathcal{O}}\left(10 \cdot \log(1/0.01) \cdot 2^{13} \cdot C_{g_{v,26}}\right) = \tilde{\mathcal{O}}\left(2^{19.05} \cdot C_{g_{v,26}}\right).$$

We compare the search model effect on ZUC-128 and ZUC-256 in Table 5 with the value of s, r, $s/(word\ length)$ 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, more random the keystream appears, more security stream cipher owns. In other words, the value $s/(word\ length)$ shouldn't be too small, and that why we only take consider of 1 or 2 words length keystream.

Table 5: Searching Complexity of the Stream Cipher ZUC-128/256

Stream Cipher	s	r	Case 1		Case 2	
			$s/32$	Complexity	$s/64$	Complexity
ZUC-128	19	7	59.4%	$\tilde{\mathcal{O}}(2^{33.81})$	29.7%	$\tilde{\mathcal{O}}(2^{33.85})$
ZUC-256	26	10	81.3%	$\tilde{\mathcal{O}}(2^{37.83})$	40.6%	$\tilde{\mathcal{O}}(2^{37.86})$

By Table 5, little complexity is increased by doubling key length and making b_s larger in our search model. Firstly, because of the stream cipher's design intention that the encryption process has to be easy to implement, outputting one more words increases little complexity on search model. Secondly, it's unfortunate that ZUC-256 has double key length than 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 Family

Block Cipher AES Oracle 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 [15], 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 6. Besides, the item $D * W$ demonstrates the upper bound of quantum circuit complexity of AES family, which we use in the calculation of search complexity.

Table 6: Quantum Circuit of the Block Cipher AES-128/192/256

Block Cipher	Qubits(D)	Toffoli Depth(W)	$D * W$
AES-128	270	11008	2972160
AES-192	334	13144	4390096
AES-256	398	15756	6270888

Multi-round Key Search Model Effect on AES Towards the AES-128, AES-192 and AES-128, the optimal value of parameters are in Table 2. And by formula (16), the corresponding complexity is

$$\tilde{\mathcal{O}}\left(r \cdot 2^{\frac{s}{2}} \cdot \log\left(1/\delta\right) \cdot C_{AES-n}\right) = \tilde{\mathcal{O}}\left(r \cdot 2^{\frac{s}{2}} \cdot \log\left(1/\delta\right) \cdot D * W\right),$$

where the complexity C_{AES-n} of $AES - n$ is shown in Table 6.

Combined with Table 2 and Table 6, we calculate the searching complexity separately for AES-128, AES-192 and AES-256 in Table 7.

It can be concluded from Table 7, AES-256 has the double key length in AES-128, but the searching complexity just rise slightly. In other words, increasing key length of block ciphers doesn't enhance the security against our key search model.

Table 7: Searching Complexity of AES-128/192/256

Block Cipher	s	r	Searching Complexity
AES-128	19	7	$\tilde{\mathcal{O}}(2^{36.5})$
AES-192	25	8	$\tilde{\mathcal{O}}(2^{40.3})$
AES-256	26	10	$\tilde{\mathcal{O}}(2^{41.6})$

5 Conclusion

Begin with a classical chosen-plaintext (chosen-IV) attack, our model turns an unstructured key search into a nested structured search. Specifically, in the i-th round with a given pair (v_i, z_i), the candidate solutions $k \in K_i = \{k | g_{v_i,s}(k) = z_i, k \in K_{i-1}\}$ are our targets whose amplitudes are amplified using search algorithm by QSVT, where $k \in K_{i-1}$ are candidate solutions in the $i-1$-th round. At the same time, the measurement of $flag$ qubit triggers collapse of quantum state, resulting in the elimination of those state components $k \in K_{i-1} - K_i$.

There are two different cases about the construction of a punctured encryption function $g_{v_i,s} : \{0,1\}^n \to \{0,1\}^s, k \mapsto z$. For a well-designed ideal cipher, it is thought to own the property of strong pseudo-randomness, which guarantees each bit of the ciphertext (or keystream) is uniformly distributed. Hence, the output of function $g_{v_i,s}(k)$ can take the first s bits of the corresponding ciphertext (or keystream). For a practical cipher without a strong pseudo-random proof, we propose a preprocessing algorithm to decide which bits of the ciphertext should be chosen as the punctured ciphertext (or keystream). Hence, for both ideal ciphers and practical ciphers, we can construct the corresponding punctured encryption function $g_{v_i,s}(\cdot)$. After that, by measurement, we can get the uniform superposition of candidate solutions K_i, which is the preimage of z_i by function $g_{v_i,s}(k)$ when $k \in K_{i-1}$. Furthermore, we can estimate the number of K_i and the whole search complexity.

As for computing cost, the gate count is $\tilde{\mathcal{O}}\left(r \cdot 2^{\frac{n}{2(r-1)}} \cdot \log(1/\delta) \cdot C_{g_{v,s}}\right)$, where n denotes key length, s denotes the length of punctured ciphertext (or keystream), r denotes the iteration number, constant δ denotes the error in search algorithm, and $C_{g_{v,s}}$ denotes the complexity of oracle $O_{g_{v,s}}$. Besides, the required qubits are $n + m + h + r + q$ for symmetric ciphers, where m denotes block length (or initialization vector length), h denotes the block length (or the keystream length), r denotes the number of auxiliary qubits (or iteration), and q denotes the number of extra qubits in the quantum encryption oracle for symmetric ciphers.

According to the parameter selection rules, the query complexity of our search model $\tilde{\mathcal{O}}\left(r \cdot 2^{\frac{n}{2(r-1)}}\right)$ outperforms Grover's algorithm $\tilde{\mathcal{O}}\left(2^{\frac{n}{2}}\right)$. A traditional viewpoint is that doubling the key length can resist the attack from quantum search algorithm. Obviously, our search model contradicts it. Due to the universality of quantum search algorithms, the security of all current sym-

metric ciphers will be questioned. Thus, it's high time for us to propose some new design ideas of symmetric ciphers in pursuit of post-quantum security.

References

1. Katz, J., Lindell Y.: Introduction to modern cryptography. 3rd edn. CRC press, Boca Raton (2020)
2. Grover, L. K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 212–219. Association for Computing Machinery, New York (1996)
3. Guang, H. L.: Quantum signal processing by single-qubit dynamics. Cambridge: Massachusetts Institute of Technology (2017)
4. Guang, H. L., Theodore J. Y., Isaac, L. C.: Methodology of resonant equiangular composite quantum gates. Physical Review X **6**(4), 041067 (2016)
5. Guang, H. L., Isaac, L. C.: Hamiltonian simulation by qubitization. Quantum **3**, 163 (2019)
6. András, G., Yuan, S., Guang, H., L., Nathan, W.: Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 139–204. Association for Computing Machinery, New York (2019)
7. Hell, M., Johansson, T., Maximov, A., et al.: A stream cipher proposal: Grain-128. In: 2006 IEEE International Symposium on Information Theory, 1614–1618. IEEE, Seattle (2006)
8. Xiutao, F.: The ZUC Stream Cipher Algorithm. Journal of Information Security Research **2**(11), 1028–1041 (2016)
9. Michael, A. N., Isaac, L. C.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2010)
10. Martyn, J. M., Rossi, Z. M., Tan, A. K., et al.: A grand unification of quantum algorithms. PRX Quantum **2**(4), 040203 (2021)
11. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing Grover Oracles for Quantum Key Search on AES and LowMC. In: Canteaut, A., Ishai, Y. (eds) Advances in Cryptology - EUROCRYPT 2020. EUROCRYPT 2020. Lecture Notes in Computer Science(), vol 12106. Springer, Cham (2020)
12. Zhuang, S.: Quantum circuit implementations of symmetric ciphers. Institute of Information Engineering, Chinese Academy of Sciences, Beijing (2021)
13. Bonnetain, X., Schrottenloher, A., Sibleyras, F.: Beyond Quadratic Speedups in Quantum Attacks on Symmetric Schemes. In: Dunkelman, O., Dziembowski, S. (eds) Advances in Cryptology - EUROCRYPT 2022. EUROCRYPT 2022. Lecture Notes in Computer Science(), vol 13277. Springer, Cham (2022)
14. Grassl M., Langenberg B., Roetteler M., Steinwandt R.: Applying Grover's algorithm to AES: quantum resource estimates. In: Takagi, T. (ed.) PQCrypto 2016. LNCS, vol. 9606, 29–43. Springer, Heidelberg (2016).
15. Li, Z., Cai, B., Sun, H., et al.: Novel quantum circuit implementation of Advanced Encryption Standard with low costs. Sci. China Phys. Mech. Astron. 65, 290311 (2022)
16. Zalka, C.: Grover's quantum searching algorithm is optimal. Physical Review A, 60, 2746–2751 (1997)
17. Design Team.: ZUC-256 stream cipher. Journal of Cryptologic Research **5**(2), 167–179 (2018).

18. Deutsch, D., Richard J.: Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences 439, 553 - 558 (1992).
19. Cerf, N., Grover, L., Williams, C.: Nested Quantum Search and NP-Hard Problems. AAECC 10, 311–338 (2000)
20. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, 124–134 (1994)
21. Bathe, B., Anand R., Dutta S.: Evaluation of Grover's algorithm toward quantum cryptanalysis on ChaCha. Quantum Inf. Process., 20, 394 (2021)
22. Vittorio, G., Seth, L., Lorenzo, M.: Quantum Random Access Memory. Phys. Rev. Lett. 100, 160501 (2008)
23. Hunziker, M., Meyer, D.A.: Quantum Algorithms for Highly Structured Search Problems. Quantum Information Processing 1, 145–154 (2002)
24. Hogg, T.: Highly Structured Searches With Quantum Computers. Physical Review Letters 80, 2473–2476 (1998)
25. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(5), 1474–1483 (1997)

A The success probability under Grover's algorithm in search model

In order to figure out the success probability of search with usage of Grover's algorithm, we give an analysis as follows. As we all know if the amplitude of target state is $a \approx \sin(a)$, then after $t(a) = \lfloor \frac{\pi}{4} \cdot \frac{1}{a} \rfloor$ Grover operators, the amplitude is amplified to $f_t(a) = \sin(2t \cdot a) \approx \sin(\frac{\pi}{2}) = 1$. At the i-th round of our search model, set a random variable

$$Y_i = \frac{|K_i|}{|K_{i-1}|} \sim N(\mu, \sigma_i^2), i = 1, 2, \cdots, r,$$

where

$$\mu = \frac{1}{2^s}, \sigma_i^2 = \frac{\mu\,(1-\mu)}{|K_{i-1}|}.$$

And, the amplitude of 'good' state is $a = \sqrt{Y_i}$ at the i-th round. However, at this point, the value of a is uncertain, so that the exact number $t(a)$ is unknown.

1. In the first $r-2$ rounds, $K_{i-1} \xrightarrow{search} K_i$, $i = 1, \cdots, r-2$.

Consider random variables, $Y_i = \frac{|K_i|}{|K_{i-1}|} \sim N(\mu, \sigma_i^2), i = 1, \cdots, r-2$.

At the i-th round, for amplitude amplification, the optimal number of Grover operators is $t_1 = t(\sqrt{\mu}) = \lfloor \frac{\pi}{4} \cdot \frac{1}{\sqrt{\mu}} \rfloor$, and then amplified amplitude $f_{t_1}(a) = \sin(2t_1 a) \approx \sin(\frac{\pi}{2} \cdot \frac{a}{\sqrt{\mu}})$.

Set a random variable

$$T_i = \sin(\frac{\pi}{2} \cdot \frac{\sqrt{Y_i}}{\sqrt{\mu}}).$$

The Figure 7 shows the relations between Y_i (starting amplitude) and T_i (amplified amplitude), when applying t_1 Grover operators.

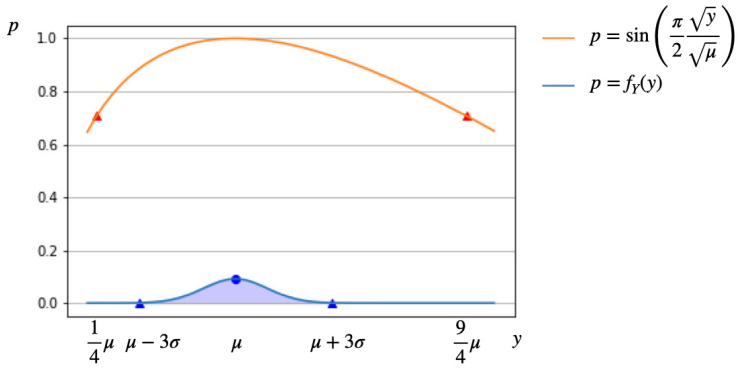


Fig. 7: Value and Distribution of Random Variable Y_i and T_i

By the formula (11), i.e.,

$$r \ll 2^{\frac{s}{2}}.$$

Thus, we treat the value $2^{-\frac{s}{2}}$ as negligible, i.e.,

$$1 - \frac{c}{2^{\frac{s}{2}}} = 1, \tag{24}$$

where c is a constant.

Without loss of generality, set $|K_{i-1}| = 2^{s+h_{i-1}}$. Then, by formula (24),

$$\sigma_i = \sqrt{\frac{\mu\,(1-\mu)}{|K_{i-1}|}} \approx \frac{1}{\sqrt{2^s \cdot |K_{i-1}|}} = \frac{1}{2^{s+\frac{h_{i-1}}{2}}}.$$

In this way, the 99.74% confidence interval of Y_i

$$[\mu - 3\sigma_i, \mu + 3\sigma_i] = \left[\frac{1}{2^s}\left(1 - \frac{3}{2^{\frac{h_{i-1}}{2}}}\right), \frac{1}{2^s}\left(1 + \frac{3}{2^{\frac{h_{i-1}}{2}}}\right)\right].$$

It's easy to conclude that $|K_{r-3}| = 2^{s+h_{r-3}} > 2^{2s}$, i.e.,

$$h_1 > \cdots > h_{r-4} > h_{r-3} > s,$$

for the choice of r and distributions of $Y_1, \cdots, Y_{r-3}$.

For $1 \le i \le r-2$, and by formula (24),

$$\mu - 3\sigma_i \ge \mu - 3\sigma_{r-2} \approx \frac{1}{2^s}\left(1 - \frac{3}{2^{\frac{h_{r-3}}{2}}}\right) \approx \frac{1}{2^s}.$$

Thus,

$$\Pr(Y_i = \mu) \approx 1, i = 1, \cdots, r-2. \tag{25}$$

Or to say, it's of probability $p \approx 1$ to amplify the amplitude by $t(\frac{1}{\sqrt{2^s}}) = \lfloor \frac{\pi}{4}\sqrt{2^s} \rfloor$ Grover operators in the first $r-2$ rounds.

2. At the $r-1$-th round, $K_{r-2} \xrightarrow{search} K_{r-1}$.

Consider $Y_{r-1} = \frac{|K_{r-1}|}{|K_{r-2}|} \sim N(\mu, \sigma_{r-1}^2)$, where $\mu = \frac{1}{2^s}$, $\sigma_{r-1}^2 = \frac{\mu(1-\mu)}{|K_{r-2}|}$.

As for the 99.74% confidence interval $[\mu-3\sigma_{r-1}, \mu+3\sigma_{r-1}]$, it has been proved that $\frac{1}{4}\mu \leq \mu-3\sigma_{r-1}$. Due to the symmetry of bell-shaped curve, $\frac{9}{4}\mu \geq \mu+3\sigma_{r-1}$, i.e.,

$$[\mu - 3\sigma_{r-1}, \mu + 3\sigma_{r-1}] \subset \left[\frac{1}{4}\mu, \frac{9}{4}\mu\right].$$

As Figure 7 shows, in the worst case that $y = \frac{1}{4}\mu$ or $y = \frac{9}{4}\mu$, $\sin(\frac{\pi}{2} \cdot \frac{\sqrt{y}}{\sqrt{\mu}}) \approx 0.71$, while in the best case that $y = \mu$, $\sin(\frac{\pi}{2} \cdot \frac{\sqrt{y}}{\sqrt{\mu}}) = 1$. Besides, the expectation of accuracy is

$$E(T_{r-1}) \geq \int_{\frac{1}{4}\mu}^{\frac{9}{4}\mu} \sin(\frac{\pi}{2} \cdot \frac{\sqrt{y}}{\sqrt{\mu}}) \cdot f_{Y_{r-1}}(y) dy = \int_{\frac{1}{4}\mu}^{\frac{9}{4}\mu} \sin(\frac{\pi}{2} \cdot \frac{\sqrt{y}}{\sqrt{\mu}}) \cdot \frac{1}{\sqrt{2\pi}\sigma_{r-1}} e^{-\frac{(y-\mu)^2}{2\sigma_{r-1}^2}} dy.$$

And it can be proved that

$$E(T_{r-1}) \geq \int_{\frac{1}{4}\mu}^{\frac{9}{4}\mu} \sin(\frac{\pi}{2} \cdot \frac{\sqrt{y}}{\sqrt{\mu}}) \cdot \frac{1}{\sqrt{2\pi}\sigma_*} e^{-\frac{(y-\mu)^2}{2\sigma_*^2}} dy \approx 0.97,$$

where $\sigma_{r-1}^2 \leq \sigma_*^2 = \frac{\mu(1-\mu)}{2^{s+4}}$ for $|K_{r-2}| \geq 2^{s+4}$, and $s = 19, 25, 26$.

3. At the r-th round, $K_{r-1} \xrightarrow{search} K_r$.

Consider $Y_r = \frac{|K_r|}{|K_{r-1}|} \sim N(\mu', \sigma_r^2)$.

By the selection rules formula (15), it's confirmed that $|K_r| = 1$, i.e.,

$$Y_r = \frac{|K_r|}{|K_{r-1}|} = \frac{1}{|K_{r-1}|}, \ E(Y_r) = \mu' = \frac{1}{2^{n-s(r-1)}}.$$

The amplitude of 'good' state is $a = \frac{1}{\sqrt{|K_{r-1}|}}$. Thus, our main goal is to figure out the value and distribution of $|K_{r-1}|$.

Set a variable

$$Z = Y_1 \cdots Y_{r-1},$$

where $Y_i \sim N(\mu, \sigma_i^2)$, and $i = 1, \cdots, r-1$. And it's easy to deduce that

$$E(Z) = \mu^{r-1}.$$

Then,

$$|K_{r-1}| = |K_0| \cdot \frac{|K_1|}{|K_0|} \cdots \frac{|K_{r-1}|}{|K_{r-2}|} = 2^n \cdot Z,$$

$$Y_r = \frac{1}{|K_{r-1}|} = \frac{1}{2^n Z}.$$

By Figure 7, when $Y_r \in [\frac{1}{4}\mu', \frac{9}{4}\mu']$, $p(Y_r) = \sin\left(\frac{\pi}{2}\frac{\sqrt{Y_r}}{\sqrt{\mu'}}\right) \geq \frac{\sqrt{2}}{2}$,

$$\Pr(Y_r \in [\frac{1}{4}\mu', \frac{9}{4}\mu']) = \Pr(Z \in [\frac{4}{9 \cdot 2^n \mu'}, \frac{4}{2^n \mu'}]) = \Pr(Z \in [\frac{4}{9} \cdot \mu^{r-1}, 4 \cdot \mu^{r-1}]).$$

So our problem turns out to be calculation of the probability $\Pr(Z \in [\frac{4}{9} \cdot \mu^{r-1}, 4 \cdot \mu^{r-1}])$.

Deduced by the definition of Z, the analytic solution is

$$\Pr\left(Z \in [\frac{4}{9} \cdot \mu^{r-1}, 4 \cdot \mu^{r-1}]\right) = \int_{\frac{4}{9}\cdot\mu^{r-1}}^{4\cdot\mu^{r-1}} \int_{-\infty}^{+\infty} \frac{1}{|t_{r-2}|} f_{Y_{r-1}}(\frac{t_{r-1}}{t_{r-2}})$$
$$\int_{-\infty}^{+\infty} \frac{1}{|t_{r-3}|} f_{Y_{r-2}}(\frac{t_{r-2}}{t_{r-3}}) \cdots \int_{-\infty}^{+\infty} \frac{1}{|t_1|} f_{Y_2}(\frac{t_2}{t_1}) f_{Y_1}(t_1)\ dt_1\ dt_2\ \cdots\ dt_{r-1}.$$

But, it's difficult to get the exact result.

In another way, by formula (25),

$$\begin{aligned}\Pr(Z \in [\frac{4}{9}\mu^{r-1}, 4\mu^{r-1}]) &= \Pr(Y_1 \cdots Y_{r-1} \in [\frac{4}{9}\mu^{r-1}, 4\mu^{r-1}]) \\ &\geq \Pr(Y_1 = \cdots = Y_{r-2} = \mu, Y_{r-1} \in [\frac{4}{9}\mu, 4\mu]) \\ &= \Pr(Y_{r-1} \in [\frac{4}{9}\mu, 4\mu]) \\ &\geq \int_{\frac{4}{9}\mu}^{4\mu} \frac{1}{\sqrt{2\pi}\sigma_*} e^{-\frac{(x-\mu)^2}{2\sigma_*^2}} dx \approx 0.9869.\end{aligned}$$

Hence,

$$\Pr\left(Y_r \in [\frac{1}{4}\mu', \frac{9}{4}\mu']\right) = \Pr(Z \in [\frac{4}{9}\mu^{r-1}, 4\mu^{r-1}]) \geq 0.9869.$$

In this way, the needed number of Grover operators is $\lfloor \frac{\pi}{4}\sqrt{2^{n-(r-1)s}} \rfloor$, and the success probability is $p \geq 0.9869$.

In summary, in the first $r-1$ round, the needed number of Grover operators is $\lfloor \frac{\pi}{4}\sqrt{2^s} \rfloor$, while in the last round, the needed number of Grover operators is $\lfloor \frac{\pi}{4}\sqrt{2^{n-(r-1)s}} \rfloor$.

However, in the last round, the prerequisite for Grover's algorithm is that $|K_{r-1}| \gg 1$. And the uncertainty of $|K_{r-1}|$ brings troubles in testifying whether $|K_{r-1}| \gg 1$, resulting in a more complex error estimation and propagation. Thus, we prefer search algorithm by QSVT rather than Grover algorithm.

B The proof of Theorem 4

Consider separately in two cases, $p \in \left[\left(\frac{1}{2} - \varepsilon\right)^s, \frac{1}{2^s}\right]$ and $p \in \left[\frac{1}{2^s}, \left(\frac{1}{2} + \varepsilon\right)^s\right]$.

1. When $p \in \left[\left(\frac{1}{2} - \varepsilon\right)^s, \frac{1}{2^s}\right]$, the decreasing scale from $|K_{i-1}|$ to $|K_i|$ might be too large. It's needed to confirm that $\frac{1}{2^{s+2}}$ still work well as the lower bound of $Y = \frac{|K_i|}{|K_{i-1}|}$ with probability at least 0.9987.

As formula (5) indicates, the 99.74% confidence interval is

$$p - \frac{3\sqrt{p(1-p)}}{\sqrt{|K_{i-1}|}} \leq Y \leq p + \frac{3\sqrt{p(1-p)}}{\sqrt{|K_{i-1}|}}.$$

Set the function $f(p) = p - \frac{3\sqrt{p(1-p)}}{\sqrt{|K_{i-1}|}}$, and solve the inequality

$$p - \frac{3\sqrt{p(1-p)}}{\sqrt{|K_{i-1}|}} \geq \frac{1}{2^{s+2}}. \tag{26}$$

And formula (26) is equivalent to a quadratic inequality

$$(1 + \frac{9}{|K_{i-1}|})p^2 - (\frac{1}{2^{s+1}} + \frac{9}{|K_{i-1}|})p + \frac{1}{2^{2s+4}} \geq 0, \tag{27}$$

when

$$p \geq \frac{1}{2^{s+2}}, \tag{28}$$

deduced from formula (26). For simplicity, let $|K_i| = 2^{s+h}$, and $h \geq 0$. According to formula (27), get

$$-\frac{b}{2a} = \frac{1}{2} \cdot \frac{2^h + 18}{2^h 2^{s+1} + 18} \geq \frac{1}{2^{s+2}},\ \Delta = \frac{9}{2^{2s+h}} + \frac{81}{2^{2s+2h}} - \frac{9}{2^{3s+h+2}} > 0,$$

and two roots

$$p_0, p_1 = \frac{-b \pm \sqrt{\Delta}}{2a} = \frac{2^{h-1} + 9 \pm \frac{3}{2}\sqrt{2^{h+2} + 36 - 2^{h-s}}}{2^{h+s+1} + 18}.$$

Thus, only when $p \in [0, p_0] \cup [p_1, 1]$, inequality (27) holds. However, it's easy to deduce that $p_0 < \frac{1}{2^{s+2}}$ which contradicts formula (28). Consequently, we have to make sure $p \in \left[\left(\frac{1}{2} - \varepsilon\right)^s, \frac{1}{2^s}\right] \subset [p_1, 1]$, i.e.,

$$p_1 \leq \left(\frac{1}{2} - \varepsilon\right)^s.$$

And the solution is

$$0 \leq \varepsilon \leq \frac{1}{2} - \sqrt[s]{\frac{2^{h-1} + 9 + \frac{3}{2}\sqrt{2^{h+2} + 36 - 2^{h-s}}}{2^{h+s+1} + 18}}. \tag{29}$$

Let function $t(h) = \frac{1}{2} - \sqrt[s]{\frac{2^{h-1}+9+\frac{3}{2}\sqrt{2^{h+2}+36-2^{h-s}}}{2^{h+s+1}+18}}$, and variable $x = 2^h$. Solve the inequality $t(h) \geq 0$, which making the formula (29) hold. Taking $x = 2^h$ into the inequality $t(h) \geq 0$, we get

$$x^2 + (\frac{25}{2^s} - 16)x + \frac{144}{2^{2s}} - \frac{144}{2^s} \geq 0.$$

The roots of this quadratic equation is that

$$x_0 < 0, x_1 = \frac{1}{2}\left(16 - \frac{25}{2^s} + \sqrt{16^2 - \frac{224}{2^s} + \frac{49}{2^{2s}}}\right) \leq 16,$$

i.e., if

$$2^h \geq 16, h \geq 4,$$

then $t(h) > 0$, which perfectly satisfies condition (6) in the ideal situation.

Because $t(h)$ is a monotonic increasing function,

$$0 \leq \varepsilon \leq \varepsilon_0 \leq t(4),$$

when $h \geq 4$, i.e.,

$$\varepsilon_0 \leq t(4) = \frac{1}{2} - \sqrt[s]{\frac{17 + 3\sqrt{25 - 4 \cdot 2^{-s}}}{2^{s+5} + 18}}. \tag{30}$$

Hence, if the inequality (30) holds, search algorithm by QSVT can still work well with parameter $\Delta = 2^{-\frac{s}{2}}$.

2. When $p \in \left[\frac{1}{2^s}, \left(\frac{1}{2} + \varepsilon\right)^s\right]$, the decreasing scale from K_{i-1} to K_i might be too little, so that the final measurement may be a spurious key rather than the correct key k^*. We have to confirm that the iteration number r guarantees the uniqueness and correctness of our model.

Let

$$p_i = \Pr_{k' \neq k^*} \left(g_{v_i,s}(k') = g_{v_i,s}(k^*) = z_i\right),$$

and $p_i \in \left[\frac{1}{2^s}, \left(\frac{1}{2} + \varepsilon\right)^s\right]$.

Then, given r pairs of $(v_1, z_1), \cdots, (v_r, z_r)$,

$$p = \Pr_{k' \neq k^*} \left(g_{v_i,s}(k^*) = g_{v_i,s}(k'), i = 1, \cdots, r\right) = \prod_{i=1}^{r} p_i.$$

Hence,

$$p \in \left[\frac{1}{2^{rs}}, \left(\frac{1}{2} + \varepsilon\right)^{rs}\right]. \tag{31}$$

As analyzed above, the random variable W, representing the number of spurious keys, $W \sim B(2^n - 1, p)$. And by the formula (31),

$$\Pr(W = 0) = (1 - p)^{2^n - 1} \approx e^{-(2^n - 1) \cdot p} \geq e^{-(2^n - 1) \cdot \left(\frac{1}{2} + \varepsilon\right)^{rs}}. \tag{32}$$

Choose constant u as the least expected probability, and solve the inequality

$$e^{-(2^n - 1) \cdot \left(\frac{1}{2} + \varepsilon\right)^{rs}} \geq u.$$

The solution is

$$\varepsilon \leq \varepsilon_0 \leq \sqrt[rs]{-\frac{\ln u}{2^n - 1}} - \frac{1}{2}. \tag{33}$$

Hence, if inequality (33) holds, after r iterations, it's of at least u probability to return the unique and correct key k^*.

In summary, by the formula (30) and (33),

$$\varepsilon_0 = \min\left(\sqrt[rs]{-\frac{\ln u}{2^n-1}} - \frac{1}{2}, \frac{1}{2} - \sqrt[s]{\frac{17+3\sqrt{25-4\cdot 2^{-s}}}{2^{s+5}+18}}\right),$$

where u is a constant of the least expected probability. At this time, even if $p \in \left[\left(\frac{1}{2}-\varepsilon_0\right)^s, \left(\frac{1}{2}+\varepsilon_0\right)^s\right]$, our nested search model can work as well as in the ideal situation $p = \frac{1}{2^s}$.