Quantum Information Processing (2020) 19:240
https://doi.org/10.1007/s11128-020-02741-2

A quantum related-key attack based on the Bernstein–Vazirani algorithm

Huiqin Xie[1,2,3] · **Li Yang**[1,2,3]

Received: 15 October 2019 / Accepted: 2 July 2020 / Published online: 14 July 2020

Abstract
Due to the powerful computing capability of quantum computers, cryptographic researchers have applied quantum algorithms to cryptanalysis and obtained many interesting results in recent years. In this paper, we study related-key attack in the quantum setting and propose a specific related-key attack, which can recover the key of block ciphers efficiently as long as the attacked block ciphers satisfy certain condition. The attack algorithm employs the Bernstein–Vazirani algorithm as a subroutine and requires the attacker to query the encryption oracle with quantum superpositions. We give a condition under which the attack will succeed and prove that any block cipher either satisfies the condition or has a distinguishing attack. As a specific example of its application, we use the attack algorithm to extract the private key of the Even–Mansour cipher. The results of this study show the power of related-key attack when combined with quantum algorithms and provide guidance for the design of quantum-secure block ciphers.

Keywords Quantum cryptanalysis · Quantum cryptography · Quantum algorithm · Block cipher · Related-key attack

1 Introduction

Shor's algorithm [1] indicates that once scalable quantum computers are available, many widely used asymmetric cryptosystems, such as the Rivest–Shamir–Adleman (RSA) cryptosystem, will be broken. This has sparked an upsurge in research on post-quantum cryptography, which studies classical systems that are secure against

✉ Li Yang
yangli@iie.ac.cn

1 State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

2 State Key Laboratory of Cryptology, P. O. Box 5159, Beijing 100878, China

3 School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

quantum adversaries. In response to the threat of quantum computing, the National Institute of Standards and Technology (NIST) has initiated the process of standardizing post-quantum public-key algorithms [2].

Although less attention is paid to symmetric cryptosystems than to public-key cryptography, the former are also suffering threats from quantum attacks. For example, due to Grover's algorithm [3], general exhaustive search attacks can obtain a quadratic speedup. Therefore, in the post-quantum world, the key lengths of symmetric schemes need to be doubled to obtain an equivalent ideal security.

While Grover's algorithm provides only a quadratic speedup, some quantum algorithms can efficiently break the symmetric systems that have been proved to be secure against classical adversaries. For instance, Kuwakado and Morii made use of Simon's algorithm [4] to construct a three-round distinguisher of Feistel's scheme [5], which has been proved to be a secure pseudorandom permutation in a classical computing setting [6]. Kuwakado and Morii also used Simon's algorithm to extract the private key of the Even–Mansour cipher [7]. Afterwards, Santoli et al. [8] and Kaplan et al. [9] extended their results independently and applied Simon's algorithm to other symmetric primitives, such as the Galois GMAC and CLOC. Dong and Wang executed a quantum key-recovery attack on Feistel's scheme with an arbitrary number of rounds using the quantum distinguisher proposed in [5]. Subsequently they further attacked the generalized Feistel scheme by similar method [10,11].

Besides the attacks on specific symmetric primitives, quantum algorithms are also applied to traditional cryptanalytic tools. Zhou et al. [12] first used Grover's algorithm in differential attack, resulting in a quadratic speedup. Kaplan et al. [13] then further applied Grover's algorithm to linear cryptanalysis and several variants of differential cryptanalysis. Xie and Yang studied quantum differential attack and applied the Bernstein–Vazirani (BV) algorithm [14] to find high-probability differentials [15]. Up to now, the proposed quantum attacks for symmetric cryptosystems are mainly based on Grover's algorithm or Simon's algorithm.

All of the above attacks have been proposed in the quantum chosen-plaintext attack model [16–18]. Traditional chosen-plaintext attack allows the adversary to obtain the ciphertexts of arbitrary plaintexts [19]. This aspect is formalized by the ability of the adversary to query an encryption oracle, which can be regarded as a black box. The adversary's goal is to extract the information of the secret encryption key. Quantum chosen-plaintext attack confers more power on the adversary, allowing him/her to query the encryption oracle with quantum superpositions. It portrays the access that an adversary might have in the quantum computing environment. Common attack models also include chosen-ciphertext attack [20], related-key attack [21,22], known-key distinguishing attack [23] and so on. The corresponding quantum models of some of these attack models have been studied [17,18,24,25].

Related-key attack is a powerful cryptanalytic tool, which gives the adversary more ability than chosen-plaintext attack. In classical related-key attack, the adversary can ask for encryptions or decryptions of messages under different keys, whose values are unknown, but have some known mathematical relationship to the target key. For example, Winternitz and Hellman study the related-key model with the key relation of bit-flip [26]. This attack model has been applied to the 9-round Rijndael scheme [27]. Related-key attack models with more general key relations were considered

in [28,29]. Some important cryptographic protocols have been found to be insecure under the related-key attack, including KASUMI designed for the 3G confidentiality and integrity algorithms [30,31] and Wired Equivalent Privacy (WEP) used in WIFI wireless networks [32,33].

The quantum related-key attack model was first studied by Roetteler and Steinwandt [25] and has attracted the attention of cryptanalysts. Roetteler and Steinwandt showed that, assuming that the key of the block cipher can be uniquely determined by a small amount of accessible plaintext–ciphertext pairs, a quantum adversary can efficiently extract its key by using a quantum related-key attack. Hosoyamada and Aoki then investigated the related-key attack in more detail and proposed a polynomial-time quantum algorithm that recovers the key of the 2-round iterated Even–Mansour scheme with only two queries to the related-key oracle [34].

To actually implement quantum algorithms, quantum computers are needed. Although universal quantum computers have not been built, the research on constructions and applications of quantum algorithm are still significant. At present, several physical implementations for practical quantum computing have been proposed, including trapped ion [35], superconducting [36], linear optical systems [37], nuclear magnetic resonance [38] and so on. In trapped ion systems, qubits are represented by the internal electronic states of the ion, such as ground and excited states, and the coherent operations on qubits are characterized as gates. For superconducting quantum computers, the single-qubit gate is realized by Josephson junction.

The first quantum computer was realized in 1999 via superconductors by D-Wave Systems, a Canadian company. After a series of improvements, in 2018, they announced general commercial availability of a 2000-qubit quantum computer. However, the D-wave computers are non-universal, since they can only perform specific tasks. In 2017, IBM proclaimed the goal to build commercially available, universal quantum computing systems. Their IBM-Q quantum systems were delivered via the IBM Cloud platform. After that, other companies such as Microsoft, Google, Intel also joined the race of building universal quantum computers. In 2017, IBM reveals a quantum computer with 50 qubits, which can maintain the quantum state for 90 μs. In 2018, Google reported the development of a 72-qubit quantum chip, called "Bristlecone". In 2019, IBM reveals a quantum computer with 53 qubits. In terms of the implementation of the quantum algorithms, several experiments of Shor's algorithm for small numbers have been realized [39–42]. Until now the largest number factored by Shor's algorithm is 21.

In this paper, we study quantum related-key attack on block ciphers. Based on the BV algorithm, we propose a quantum attack for recovering the key of block ciphers and rigorously analyze its complexity. Our algorithm exploits the fact that the BV algorithm can be used to find linear structures of Boolean functions. We first construct a new function based on the attacked block cipher so that the constructed function has a nontrivial linear structure that reveals the information of the secret key; then we apply the BV algorithm to obtain the linear structures and recover the key.

We prove that as long as the attacked block cipher satisfies certain condition, the quantum attack algorithm can output its private key except for a negligible probability. Furthermore, we analyze the success condition of the attack and demonstrate that any well-defined block cipher should meet the condition, since the violation implies the

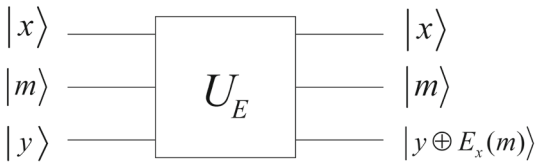

Fig. 1 Quantum gate U_E

existence of a distinguishing attack to the block cipher. The complexity of the attack is polynomial. As a specific application example of our quantum attack, we use the algorithm to extract the private key of the Even–Mansour construction.

2 Preliminaries and notations

Throughout this paper, we let $\mathbb{F}_2 = \{0, 1\}$ represent the finite field with two elements. E denotes an arbitrary block cipher with block size n and key length k. When a secret key $s \in \mathbb{F}_2^k$ is specified, E_s is a permutation from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. We assume that E can be efficiently implemented by a quantum circuit. That is, there exists a polynomial-time quantum circuit that takes a secret key and a plaintext as input, and outputs the corresponding ciphertext. The circuit implements the following unitary operator:

$$U_E : \sum_{m,x,y} |x\rangle|m\rangle|y\rangle \longrightarrow \sum_{m,x,y} |x\rangle|m\rangle|y \oplus E_x(m)\rangle.$$

This assumption clearly holds for all block ciphers used in practice. Since the quantum circuit of U_E does not involve the secret key s, anyone can execute the unitary operator U_E, including the attacker.

Because the unitary quantum gates $\{H, CNOT, Phase, \frac{\pi}{8}\}$ form a universal gate set [43], we can assume that the quantum circuit implementing U_E is composed of the gates in this set. Here, H is the Hadamard gate, $CNOT$ is the controlled-NOT gate, $Phase$ is the phase gate, and $\frac{\pi}{8}$ is the $\frac{\pi}{8}$ gate. Let $|E|_Q$ denote the number of universal gates in the quantum circuit implementing E, then $|E|_Q$ is a polynomial of parameter n. The attacker can integrate U_E into his circuits as in Fig. 1.

2.1 Quantum related-key attack

We first recall the traditional related-key attack model proposed in [26], where the key relation is restricted to bit-flips. In this model, after a secret key $s \in \mathbb{F}_2^k$ is determined, the attacker can query following two oracles:

$\mathcal{E}$: On inputting a plaintext $m \in \mathbb{F}_2^n$ and a bitmask $x \in \mathbb{F}_2^k$, $\mathcal{E}$ returns the encryption $E_{s\oplus x}(m)$.
$\mathcal{D}$: On inputting a ciphertext $c \in \mathbb{F}_2^n$ and a bitmask $x \in \mathbb{F}_2^k$, $\mathcal{D}$ returns the decryption $E^{-1}_{s\oplus x}(c)$.

After querying these oracles, the attacker needs to output a vector $s' \in \mathbb{F}_2^k$ as a guess for s. He succeeds if and only if $s' = s$.

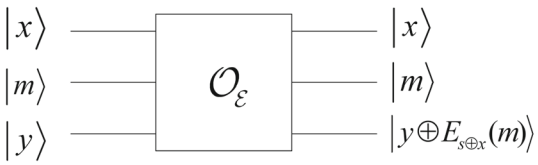

Fig. 2 Quantum gate $\mathcal{O}_{\mathcal{E}}$

The attack presented in this paper does not require the access to the decryption oracle $\mathcal{D}$, but the attacker is allowed to query the encryption oracle $\mathcal{E}$ with superpositions of keys. That is, the attacker can query a quantum oracle $\mathcal{O}_{\mathcal{E}}$ operating as follows:

$$\mathcal{O}_{\mathcal{E}} : \sum_{x,m,y} |x\rangle|m\rangle|y\rangle \longrightarrow \sum_{x,m,y} |x\rangle|m\rangle|y \oplus E_{s\oplus x}(m)\rangle.$$

This attack model is called quantum related-key attack model. The attacker can integrate the oracle $\mathcal{O}_{\mathcal{E}}$ into his circuits as in Fig. 2. Furthermore, we allow the attacker to query the oracle that only returns one bit of the cipher with superpositions of keys. That is, supposing $E_{s\oplus x} = (E_{s\oplus x,1}, E_{s\oplus x,2}, \ldots, E_{s\oplus x,n})$, where $E_{s\oplus x,1}, \ldots, E_{s\oplus x,n}$ are all Boolean functions with one-bit output. Then, for each $j = 1, 2, \ldots, n$, the attacker can query the quantum oracle

$$\mathcal{O}_{\mathcal{E}_j} : \sum_{x,m,y} |x\rangle|m\rangle|y\rangle \longrightarrow \sum_{x,m,y} |x\rangle|m\rangle|y \oplus E_{s\oplus x,j}(m)\rangle.$$

The scenario where quantum attackers can query cryptographic primitives with quantum superpositions has been considered in a large amount of research [16–18,44]. The access to the oracle $\mathcal{O}_{\mathcal{E}}$ implies that the attacker can query the encryption oracle equipped the target key s. Namely, the attacker can query the following oracle:

$$\mathcal{O}_{E_s} : \sum_{m,y} |m\rangle|y\rangle \longrightarrow \sum_{m,y} |m\rangle|y \oplus E_s(m)\rangle.$$

To do this, he only needs to query $\mathcal{O}_{\mathcal{E}}$ with the state $\sum_{m,y} |\mathbf{0}\rangle|m\rangle|y\rangle$ ($\mathbf{0}$ is the zero vector in $\mathbb{F}_2^k$) and discard the first register. Therefore, quantum related-key attack model can be viewed as an extension of the quantum chosen-plaintext attack model, where the attacker is only allowed to query the encryption oracle equipped the key s with superpositions.

2.2 Linear structure

Let $C_{k,n}$ denote the set of maps from $\mathbb{F}_2^k$ to $\mathbb{F}_2^n$. The notion of the linear structure is defined as follows:

Definition 1 [45] $F \in C_{k,n}$. A vector $a \in \mathbb{F}_2^k$ is said to be a linear structure of F if there exists $b \in \mathbb{F}_2^n$ such that

$$F(x) \oplus F(x \oplus a) = b, \quad \forall x \in \mathbb{F}_2^k.$$

If $a \in \mathbb{F}_2^k$, $b \in \mathbb{F}_2^n$ satisfies the above equation, then (a, b) is called a linear structure pair of F. Let U_F denote the set of all linear structures of F, and $U_F^b := \{a \in \mathbb{F}_2^k | F(x) \oplus F(x \oplus a) = b,\ \forall x \in \mathbb{F}_2^k\}$. Then, $U_F = \bigcup_b U_F^b$.

Let $F = (F_1, F_2, \ldots, F_n)$, where F_j $(j = 1, 2, \ldots, n)$ is the jth component function of F. $F_j \in C_{k,1}$, and for any $x \in \mathbb{F}_2^k$, F_j maps x to the jth bit of $F(x)$. Then a is a linear structure of F if and only if it is a linear structure of F_j for each $j = 1, 2, \ldots, n$. To find a linear structure of F, we only need to find linear structures of every F_j first and then select a common linear structure. Therefore, in order to find linear structures of functions in the set $C_{k,n}$ for a general parameter n, we only need to focus on the case of $n = 1$.

Linear structures of the functions in $C_{k,1}$ can be determined by their Walsh spectrum, which is defined as follows:

Definition 2 Suppose $f : \mathbb{F}_2^k \to \mathbb{F}_2$ is a function in $C_{k,1}$. The Walsh spectrum of f is defined as

$$\begin{aligned} S_f : \mathbb{F}_2^k &\longrightarrow \mathbb{F}_2 \\ \omega &\longrightarrow S_f(\omega) = \frac{1}{2^k} \sum_{x \in \mathbb{F}_2^k} (-1)^{f(x)+\omega \cdot x}, \end{aligned}$$

which is also a function in $C_{k,1}$.

Let U_f be the set of the linear structures of f, and $U_f^i := \{a \in \mathbb{F}_2^k | f(x) \oplus f(x \oplus a) = i,\ \forall x \in \mathbb{F}_2^k\}$ for $i = 0, 1$. We have $U_f = U_f^0 \cup U_f^1$. The following lemma shows how a linear structure can be determined by the Walsh spectrum.

Lemma 1 [46] *For any $f \in C_{k,1}$, let $N_f := \{\omega \in \mathbb{F}_2^k | S_f(\omega) \neq 0\}$. Then for $\forall i \in \{0, 1\}$, it holds that*

$$U_f^i = \{a \in \mathbb{F}_2^k | a \cdot \omega = i,\ \forall \omega \in N_f\}.$$

According to the above lemma, if one has a large enough subset W of N_f, it is possible to solve the linear equation group $\{x \cdot \omega = i | \omega \in W\}$ to obtain the linear structures of f. As discussed previously, by applying this method to find each F_j's linear structures, it is possible to obtain the linear structures of F. (Here, solving the linear equation group $\{x \cdot \omega = i | \omega \in W\}$ means seeking vectors x such that $x \cdot \omega = i$ for $\forall \omega \in W$.)

2.3 Bernstein–Vazirani algorithm

The BV algorithm was proposed in 1997 [14]. Given the quantum oracle access of a function $f(x) = a \cdot x$, where $a \in \mathbb{F}_2^k$ is a secret string, the BV algorithm's original goal is to find a. However, it has been observed that when the BV algorithm is applied to a general Boolean function $f : \mathbb{F}_2^k \to \mathbb{F}_2$ in $C_{k,1}$, it will always return a vector in N_f [47]. Given the quantum oracle access of f, the BV algorithm is executed as follows:

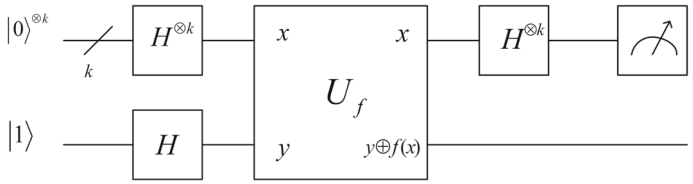

Fig. 3 Quantum circuit of the Bernstein–Vazirani (BV) algorithm

1. Perform the Hadamard operator $H^{(k+1)}$ on the initial state $|\psi_0\rangle = |0\rangle^{\otimes k}|1\rangle$ to obtain

$$|\psi_1\rangle = \sum_{x\in\mathbb{F}_2^k} \frac{|x\rangle}{\sqrt{2^k}} \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

2. Query the oracle of f, obtaining

$$|\psi_2\rangle = \sum_{x\in\mathbb{F}_2^k} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^k}} \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

3. Perform the Hadamard operator $H^{(k)}$ to the first k qubits and discard the $(k+1)$-th qubit, producing

$$\begin{aligned}|\psi_3\rangle &= \sum_{y\in\mathbb{F}_2^k} \left(\frac{1}{2^k} \sum_{x\in\mathbb{F}_2^k} (-1)^{f(x)+y\cdot x} \right) |y\rangle \\ &= \sum_{y\in\mathbb{F}_2^k} S_f(y)|y\rangle.\end{aligned}$$

By measuring $|\psi_3\rangle$ in the computational basis, the probability of outputting $y \in \mathbb{F}_2^k$ is $S_f(y)^2$. Thus, the measurement on $|\psi_3\rangle$ always yields a vector y in N_f.

Since applying the BV algorithm to a function $f \in C_{k,1}$ always returns a vector in N_f, by Lemma 1, it is possible to use the BV algorithm to find linear structures of an arbitrary function in $C_{k,1}$.

Executing the BV algorithm needs a total of $2k + 1$ Hadamard gates and one quantum query. The number of qubits required is $k + 1$. The quantum circuit of the BV algorithm is presented in Fig. 3.

3 Quantum algorithm for finding linear structures

A quantum algorithm for finding nonzero linear structures of functions in $\mathcal{C}_{k,n}$ was proposed by Xie and Yang in [15]. Suppose $F = (F_1, F_2, \ldots, F_n) \in \mathcal{C}_{k,n}$. For

each $j = 1, 2, \ldots, n$, their algorithm first calls the BV algorithm to obtain a subset of N_{F_j}. Then, it uses this subset to compute linear structures of F_j according to Lemma 1. Thereafter, the algorithm selects an nonzero common linear structure of $F_1, F_2, \ldots, F_n$ and outputs it. The output vector has a high probability of being a linear structure of F. We make a minor modification to the algorithm in [15] so that it outputs a set containing all linear structures of F instead of only a random linear structure. The modified algorithm is as follows:

Algorithm **FindStruct**

Initialization: $p(n)$ is a polynomial of n chosen by the attacker. $F = (F_1, F_2, \ldots, F_n) \in C_{k,n}$. The quantum oracle access of each F_j $(1 \leq j \leq n)$ is given.

1. For each $j = 1, 2, \ldots, n$, apply the BV algorithm $p(n)$ times to F_j to obtain a subset W_j of N_{F_j}. The size of W_j is $p(n)$.
2. For each $j = 1, 2, \ldots, n$, solve the linear equation group $\{x \cdot \omega = i_j | \omega \in W_j\}$ to obtain the solution $A_j^{i_j}$ for $i_j = 0, 1$. Let $A_j = A_j^0 \cup A_j^1$.
3. Compute the intersection $\bar{A} = A_1 \cap A_2 \cap \cdots \cap A_n$. For each $a \in \bar{A}$, let $\tilde{a} = (i_1, i_2, \ldots, i_n)$, where $i_1, i_2, \ldots, i_n$ are the corresponding superscripts such that $a \in A_1^{i_1} \cap A_2^{i_2} \cap \cdots \cap A_n^{i_n}$. Let $A = \{(a, \tilde{a}) | a \in \bar{A}\}$ and output A.

In the above algorithm, when the attacker computes $A_j = A_j^0 \cup A_j^1$ in Step 2, he actually needs to attach a tag to each vector in A_j. Specifically, if $a \in A_j^0$, then a tag $i_j = 0$ is attached to a when it is put into the set A_j; if $a \in A_j^1$, then a tag $i_j = 1$ is attached to a when it is put into the set A_j. Subsequently, when the attacker computes the intersection $\bar{A}$, for each $a \in A_1 \cap A_2 \cap \cdots \cap A_n$, he attaches the corresponding n tags $i_1, i_2, \ldots, i_n$ to a when putting it in $\bar{A}$. Therefore, when calculating the set A, the attacker can easily obtain the corresponding $\tilde{a}$ of each $a \in \bar{A}$ by tracking these tags. These tags are used to avoid the need to compute the intersection of n sets for an exponential number of times. With these tags, the attacker only needs to compute the intersection $\bar{A} = A_1 \cap A_2 \cap \cdots \cap A_n$ once to obtain the set A. However, without these tags, the attacker needs to compute the intersections $A_1^{i_1} \cap A_2^{i_2} \cap \cdots \cap A_n^{i_n}$ to obtain the linear structures in $U_F^{(i_1, \ldots, i_n)}$ for each $i_1, i_2, \ldots, i_n \in \{0, 1\}$. Thus, he would need to compute the intersection 2^n times.

The following two theorems demonstrate the feasibility of algorithm **FindStruct**. Theorem 2 was proved in [15], and thus we omit its proof.

Theorem 1 *Suppose $F = (F_1, F_2, \ldots, F_n) \in C_{k,n}$ and a is an arbitrary linear structure of F. Let α be the vector such that $a \in U_F^\alpha$. If running the algorithm **FindStruct** on F returns a set A, then (a, α) must belong to the set A.*

Proof Suppose $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$. Since $a \in U_F^\alpha$, we have $a \in U_{F_j}^{\alpha_j}$ for each $j = 1, 2, \ldots, n$. According to Lemma 1, for any vector $\omega \in N_{F_j}$, it holds that $a \cdot \omega = \alpha_j$. As per the properties of the BV algorithm, we know that the set $W_j \subseteq N_{F_j}$. Thus, a is a solution of the linear equation group $\{x \cdot \omega = \alpha_j | \omega \in W_j\}$ for each $j = 1, 2, \ldots, n$. Therefore, we have $a \in \bar{A}$, and $\alpha_1, \alpha_2, \ldots, \alpha_n$ are superscripts such that $a \in A_1^{\alpha_1} \cap A_2^{\alpha_2} \cap \cdots \cap A_n^{\alpha_n}$, which means $(a, \alpha) \in A$. □

Before stating Theorem 2, we need to define two parameters first. For any function $f \in C_{k,1}$, let

$$\Delta_f = \frac{1}{2^k} \max_{\substack{a \in \mathbb{F}_2^k \\ a \notin U_f}} \max_{i \in \mathbb{F}_2} |\{x \in \mathbb{F}_2^k | f(x \oplus a) + f(x) = i\}|. \tag{1}$$

Obviously $\Delta_f < 1$. If Δ_f is close to 1, then there exists a vector a that is not a linear structure of f but satisfies $f(x \oplus a) \oplus f(x) = i$ for most $x \in \mathbb{F}_2^k$ and some fixed i.

For an arbitrary function $f \in C_{k,1}$ and a subset of its linear structures $S \subseteq U_f$, we define

$$\delta_f(S) = \frac{1}{2^k} \max_{\substack{a \in \mathbb{F}_2^k \\ a \notin S}} \max_{i \in \mathbb{F}_2} |\{x \in \mathbb{F}_2^k | f(x \oplus a) + f(x) = i\}|. \tag{2}$$

For any $S \subseteq U_f$, it is obvious that $\Delta_f \leq \delta_f(S) \leq 1$. If S contains all linear structures of f, then $\delta_f(S) = \Delta_f < 1$.

For a multiple-output Boolean function $F \in C_{k,n}$, we define $\Delta_F = \max_j \Delta_{F_j}$, where Δ_{F_j} is defined as Eq. (1). The larger the value of Δ_F, the more difficult it is to exclude the vectors that are not linear structures of F when applying the algorithm **FindStruct** to F. Similarly, we define $\delta_F(S) = \max_j \delta_{F_j}(S)$, where $\delta_{F_j}(S)$ is defined as Eq. (2). If $\delta_f(S) < 1$, then we can apply algorithm **FindStruct** to F to determine the set S. The larger the value of $\delta_F(S)$, the more difficult it is to exclude the vectors that are not in S when using algorithm **FindStruct**. If $\delta_F(S) = 1$, the output set A of algorithm **FindStruct** will always contain vectors that are not in S. Then it is necessary to determine the set S to further verify the vectors in set A according to additional conditions.

Theorem 2 [15] *Suppose $F \in C_{k,n}$ and $\Delta_F \leq p_0 < 1$ for some constant p_0. If running the algorithm **FindStruct** on F returns a set A, then for any $(a, i_1, i_2, \ldots, i_n) \in A$, it holds that*

$$\Pr\left[a \in U_F^{(i_1,\ldots,i_n)}\right] \geq 1 - p_0^{p(n)}.$$

That is, except for a negligible probability, the vectors in A must be the linear structures of F. Moreover, if there exists a subset $S \subseteq U_F$ such that $\delta_F(S) \leq p_0 < 1$ for some constant p_0, then for any $(a, i_1, i_2, \ldots, i_n) \in A$, it holds that

$$\Pr[a \in S] \geq 1 - p_0^{p(n)}.$$

For the special case where the set S only contains a nonzero linear structure s of F, that is, $S = \{\mathbf{0}, s\}$, we use the notation $\delta_F(s)$ to denote the parameter $\delta_F(\{\mathbf{0}, s\})$ for simplicity. In this situation, algorithm **FindStruct** can be used to find the vector s.

Theorem 1 indicates that all linear structures of F must be in the output set A. Noting that the zero vector is a trivial linear structure of F, the set A is always nonempty. Theorem 2 shows that, under the condition $\Delta_F \leq p_0 < 1$, the vectors in the set A

output by the algorithm **FindStruct** with $p(n) = O(n)$ must be linear structures of F except for a negligible probability.

4 Attack strategy

In this section, we present a strategy for attacking general block ciphers using the BV algorithm in the quantum related-key attack model. We first describe the attack and then analyze under what condition the attack will work and its complexity.

4.1 Description of the attack

The following two steps describe a strategy for attacking a symmetric cryptosystem E using the BV algorithm:

1. Construct a new function F based on the cipher E so that F satisfies two conditions: (I) the attacker has quantum oracle access to F; (II) F has a nontrivial linear structure that reveals the information of the secret key. Sometimes the linear structure itself is the secret key.
2. Apply the algorithm **FindStruct** to obtain the linear structure of F and use it to recover the secret key.

We now confine ourselves to the Electronic Codebook mode and give a specific attack strategy. For any vector $x \in \mathbb{F}_2^k$, we use $\lceil x \rceil^n$ to denote the vector consisting of the first n bits of x. If $k < n$, then $\lceil x \rceil^n$ is obtained by adding $n - k$ zeros at the end of x. That is,

$$\lceil x \rceil^n = \begin{cases} (x_1, x_2, \ldots, x_n) & k \geq n \\ (x_1, \ldots, x_n, \underbrace{0, \ldots, 0}_{k-n}) & k < n \end{cases} \tag{3}$$

Suppose $E_s : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a block cipher with a secret key $s \in \mathbb{F}_2^k$. Let m be an arbitrary plaintext in the plaintext space. Define the function

$$\begin{aligned} F_s^m : \mathbb{F}_2^k &\longrightarrow \mathbb{F}_2^n \\ x &\longrightarrow E_x(m) \oplus E_{s\oplus x}(m) \oplus \lceil x \rceil^n. \end{aligned} \tag{4}$$

Then, for any $x \in \mathbb{F}_2^k$, we have $F_s^m(x \oplus s) \oplus F_s^m(x) = \lceil s \rceil^n$. Therefore, the key s is a nonzero linear structure of F_s^m. More precisely, $(s, \lceil s \rceil^n)$ is a linear structure pair of F_s^m. Thus, we can find s by applying the algorithm **FindStruct** to F_s^m. Note that the linear structure pair we are looking for satisfies the condition that its first and second parts have the same n-bit prefix. We can modify the algorithm **FindStruct** so that it finds the linear structure pairs of the form $(a, \lceil a \rceil^n)$ instead of the general linear structure pairs (a, b). This will greatly reduce the complexity of the attack.

To simplify the statement, in the following algorithm, we assume that the key length k is no less than the block size n. This assumption does not cause a loss of generality. The attack algorithm when $k < n$ can be constructed similarly. The attack algorithm based on the improved **FindStruct** algorithm is as follows:

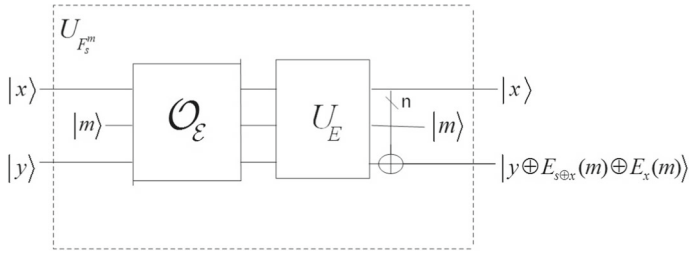

Fig. 4 Quantum circuit to implement F_s^m

Algorithm **RecoverKey**

1. Choose a polynomial $p(n)$ and a plaintext m uniformly at random. Define the function F_s^m as Eq. (4). Denote $F_s^m = (F_{s,1}^m, F_{s,2}^m, \ldots, F_{s,n}^m)$.
2. For $j = 1, 2, \ldots, n$, run the BV algorithm on $F_{s,j}^m$ $p(n)$ times to obtain a subset W_j of $N_{F_{s,j}^m}$. The size of W_j is $p(n)$.
3. For $j = 1, 2, \ldots, n$, solve the linear equation group $\{x \cdot \omega = i_j | \omega \in W_j\}$ to obtain the solution $\bar{A_j^{i_j}}$ for $i_j = 0, 1$. Let $\bar{A_j^0} = \{a \in A_j^0 | a_j = 0\}$, $\bar{A_j^1} = \{a \in A_j^1 | a_j = 1\}$, where a_j is the jth bit of a. Then, for any $a \in \bar{A_j^{i_j}}$, it holds that $a_j = i_j$. Let $A_j = \bar{A_j^0} \cup \bar{A_j^1}$.
4. Compute the intersection $\bar{A} = A_1 \cap A_2 \cap \cdots \cap A_n$. Let $A = \{(a, \lceil a \rceil^n) | a \in \bar{A}\}$. Verify the vectors in A to determine the correct key s.

Algorithm **RecoverKey** requires the quantum oracle access of F_s^m. The attacker can obtain this oracle by first querying the oracle $\mathcal{O}_{\mathcal{E}}$ to compute $|x, m, y\rangle \rightarrow |x, m, y \oplus E_{s\oplus x}(m)\rangle$ and then implementing the unitary operator $U_E : |x, m, y\rangle \rightarrow |x, m, y \oplus E_x(m)\rangle$ and n CNOT gates $CNOT^{(n)} : |x, m, y\rangle \rightarrow |x, m, y_1 \oplus x_1, \ldots, y_n \oplus x_n\rangle$ The quantum circuit to implement F_s^m is presented in Fig. 4.

4.2 Success condition of the attack

We now analyze the validity of algorithm **RecoverKey** by specifying the condition under which the attack will succeed. To do this, we first underscore the following two facts:

- Any linear structure pair of F_s^m with the form $(a, \lceil a \rceil^n)$ must be in the set A, which is computed in step 4 of algorithm **RecoverKey**. Therefore, $(s, \lceil s \rceil^n)$ must be in the set A.
- If $\delta_{F_s^m}(s) \leq p_0 < 1$ for some constant p_0, than for any vector (a, b) in the set A, the probability $(a, b) \neq (s, \lceil s \rceil^n)$ is less than $p_0^{p(n)}$.

The above two facts can be easily derived from Theorems 1 and 2, respectively. Combining these two facts shows that, under the condition $\delta_{F_s^m}(s) \leq p_0 < 1$, the attacker can obtain the private key s of the block cipher except for a negligible probability by running algorithm **RecoverKey** with $p(n) = O(n)$.

The condition $\delta_{F_s^m}(s) \leq p_0 < 1$ is a little abstract and its rationality needs to be further explained. To do this, we demonstrate that the violation of this condition implies the existence of a distinguishing attack to the block cipher E, which happens with a small probability for well-defined block ciphers.

We first compute the parameter $\delta_{F_s^m}(s)$.

$$\delta_{F_s^m}(s) = \frac{1}{2^k} \max_j \max_{\substack{a \in \mathbb{F}_2^k \\ a \notin \{\mathbf{0}, s\}}} \max_{i \in \mathbb{F}_2} |\{x \in \mathbb{F}_2^k | F_{s,j}^m(x) \oplus F_{s,j}^m(x \oplus a) = i\}|.$$

Here, "$\mathbf{0}$" denotes the zero vector in $\mathbb{F}_2^k$. The condition $\delta_{F_s^m}(s) \leq p_0 < 1$ means that there exists a constant p_0 such that for any $j \in \{1, 2, \ldots, n\}$, any $a \notin \{\mathbf{0}, s\}$ and any $i \in \{0, 1\}$, it holds that

$$\frac{|\{x \in \mathbb{F}_2^k | F_{s,j}^m(x) \oplus F_{s,j}^m(x \oplus a) = i\}|}{2^k} \leq p_0. \tag{5}$$

Since $F_s^m(x) = E_x(m) \oplus E_{x \oplus s}(m) \oplus \lceil x \rceil^n$, Eq. (5) is equivalent to

$$\frac{|\{x \in \mathbb{F}_2^k | E_{x,j}(m) \oplus E_{x \oplus s,j}(m) \oplus E_{x \oplus a,j}(m) \oplus E_{x \oplus a \oplus s,j}(m) = i \oplus a_j\}|}{2^k} \leq p_0,$$

where $E_{x,j}$ is the jth component of E_x. Since $a \notin \{\mathbf{0}, s\}$, the vectors x, $x \oplus s$, $x \oplus a$, and $x \oplus s \oplus a$ are always four different keys. The above equation means that when averaging over all possible values of x, the exclusive value of the ciphertexts of m under these four keys will not be biased towards 0 or 1 with a probability close to 1. Generally speaking, a well-constructed block cipher should not have such obvious non-uniformity. Thus, the above condition seems reasonable. In fact, if the condition does not hold, we can construct a polynomial-time distinguisher of the block cipher E_s. Specifically, if there is no constant $p_0 < 1$ such that $\delta_{F_s^m}(s) \leq p_0$, then for any constant ϵ ($0 < \epsilon < 1$), there exists a vector $a \in \mathbb{F}_2^k$, $j_0 \in \{1, 2, \ldots, n\}$ and $i_0 \in \{0, 1\}$ such that

$$\frac{|\{x \in \mathbb{F}_2^k | E_{x,j_0}(m) \oplus E_{x \oplus s,j_0}(m) \oplus E_{x \oplus a,j_0}(m) \oplus E_{x \oplus s \oplus a,j_0}(m) = i_0\}|}{2^k} \geq 1 - \epsilon$$

holds for infinitely many n. Here, we choose a constant ϵ satisfying $\epsilon << 1$ and present a simple classical distinguishing attack against the block cipher E:

1. Polynomially choose many vectors $x_1, x_2, \ldots, x_r \in \mathbb{F}_2^k$ at random, where $r = O(n)$. Then, compute $E_{x_1}(m), \ldots, E_{x_r}(m), E_{x_1 \oplus a}(m), \ldots, E_{x_r \oplus a}(m)$.
2. Query $E_{x_1 \oplus s}(m), \ldots, E_{x_r \oplus s}(m), E_{x_1 \oplus a \oplus s}(m), \ldots, E_{x_r \oplus a \oplus s}(m)$. Then, compute $Z_t = E_{x_t}(m) \oplus E_{x_t \oplus a}(m) \oplus E_{x_t \oplus s}(m) \oplus E_{x_t \oplus a \oplus s}(m)$ for $t = 1, 2, \cdots r$. The probability that $Z_t = i_0$ is greater than $1 - \epsilon$.
3. According to the Hoeffding inequality, except for a negligible probability, the proportion of random variables $Z_t's$ whose value is equal to i_0 in the set $\{Z_t | t = 1, 2, \ldots, r\}$ is greater than $1 - 2\epsilon$.

However, if the function queried by the attacker is not the block cipher E_s, but a function that is a random permutation of plaintext space when fixing an arbitrary key, then the probability that the proportion of $Z_t's$ with the value i_0 in the set $\{Z_t|t = 1, 2, \ldots, r\}$ is larger than $1 - 2\epsilon$ is negligible. Therefore the above steps constitute a distinguisher of the block cipher E_s.

4.3 Complexity of the attack

In this subsection, we analyze the computational complexity of algorithm **RecoverKey**. To accurately compute the complexity, we separate the algorithm into three parts:

(1) Executing the BV algorithm $np(n)$ times,
(2) solving $2n$ linear equation groups, and
(3) finding the intersection of A_1, A_2, …, and A_n.

For the first part, the BV algorithm must be run once to execute $2k+1$ Hadamard gates, one unitary operator U_E, and one quantum query on $\mathcal{O}_\mathcal{E}$. Thus, a total of $(2k + 1 + |E|_Q)np(n)$ universal gates and $np(n)$ quantum queries are needed. The complexity of this part is polynomial. For the second part, the attacker needs to solve $2n$ linear equation groups, and each one has k variables and $p(n)$ equations. Solving a linear equation group with k variables and $p(n)$ equations via the Gaussian elimination method needs $O(p(n)k^2)$ calculations. Thus, the complexity of this part includes $O(2p(n)nk^2)$ classical calculation, which is also a polynomial of k and n. For the third part, the attacker needs to compute the intersection $A_1 \cap A_2 \cap \cdots \cap A_n$. Let $t = \max_j |A_j|$. Finding the intersection of these n sets using the sort method requires $O(nt \log t)$ calculations. The value of t relies on the properties of F_s^m and the value of $p(n)$. The operation of sets $\bar{A_j^0}$, $\bar{A_j^1}$ greatly reduces the sizes of the sets A_j^0 and A_j^1, thereby reducing the size of A_j. Since A_j is derived by the solution of a linear system with $p(n)$ equations, the size of A_j should decrease rapidly as $p(n)$ increases. The larger the value of $p(n)$ chosen by the attacker, the smaller the value of t. Thus, the attacker can choose a larger $p(n)$ to reduce t. (Even though this will increase the number of unitary gates and queries required in the other two parts, the complexity of these two parts is still polynomial as long as $p(n)$ is still a polynomial.) For the most general case, there is no guarantee that t is polynomial. However, under the condition that $\delta_{F_s^m}(s) \leq p_0 < 1$, t is a small number.

To sum up, algorithm **RecoverKey** needs $O((2k + 1 + |E|_Q)np(n))$ universal gates, $O(2p(n)nk^2 + nt \log t)$ classical calculations, and $O(np(n))$ quantum queries. If $\delta_{F_s^m}(s) \leq p_0 < 1$ for some constant p_0, then t will be a small number, and the polynomial $p(n)$ only needs to be $O(n)$.

For the most block ciphers, we cannot give a direct proof that they meet the success condition of the proposed quantum attack, but as analyzed in Sect. 4.2, well-defined block ciphers should meet the condition. To intuitively show the superiority of the algorithm **RecoverKey**, we list the computational complexity of the best classical attack and that of our attack on several most widely used block ciphers (Table 1). The

Table 1 Complexity comparison on specific block ciphers

Block cipher	Complexity of the best known classical attack	Citation	Complexity of algorithm **RecoverKey**
AES-128	2^{126}	[48]	128^2
AES-192	$2^{189.9}$	[48]	192^2
AES-256	$2^{254.3}$	[48]	256^2
DES	$2^{39}-2^{41}$	[49]	64^2
SIMON 128/256	2^{248}	[50]	128^2
PRESENT-80	$2^{79.34}$	[51]	64^2

comparison shows that, if the block ciphers meet the success condition, the efficiency of algorithm **RecoverKey** is much better than the known classical attacks.

5 Application to the Even–Mansour Cipher

To justify the practicality of the proposed attack, in this section we show its application to Even–Mansour Cipher. We will prove that, under the quantum related-key attack model, algorithm **RecoverKey** can efficiently extract the private key of the Even–Mansour cipher.

The Even–Mansour scheme builds a block cipher from a public random permutation [52]. For a public random permutation $P : \mathbb{F}_2^n \to \mathbb{F}_2^n$, the Even–Mansour cipher of the plaintext m with the key $s = (s_1, s_2) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ is defined as

$$E_s(m) = P(m \oplus s_1) \oplus s_2. \tag{6}$$

Let $k = 2n$. Then, the key space of E is $\mathbb{F}_2^k$. For an arbitrary plaintext $m \in \mathbb{F}_2^n$, we define F_s^m as Eq. (4):

$$\begin{aligned} F_s^m : \mathbb{F}_2^k &\longrightarrow \mathbb{F}_2^n \\ x &\longrightarrow E_x(m) \oplus E_{s\oplus x}(m) \oplus \lceil x \rceil^n. \end{aligned} \tag{7}$$

By Eqs. (6) and (7), for any $x = (x_1, x_2) \in \mathbb{F}_2^k$

$$F_s^m(x) = P(m \oplus x_1) \oplus P(m \oplus x_1 \oplus s_1) \oplus s_2 \oplus x_1.$$

Here x_1 and x_2 are both n-bit strings, and they denote, respectively, the first n bits and the last n bits of x. As analyzed in Sect. 4, $F_s^m(x \oplus s) \oplus F_s^m(x) = \lceil s \rceil^n$ holds for any $x \in \mathbb{F}_2^k$. Thus, $(s, \lceil s \rceil^n)$ is a linear structure pair of F_s^m.

Before proving that algorithm **RecoverKey** can efficiently extract the Even–Mansour cipher's key, we first analyze the linear structure set of F_s^m. Let $P = (P_1, P_2, \ldots, P_n)$, where P_j $(j = 1, 2, \ldots, n)$ is the jth component function of P. That is, $P_j \in C_{n,1}$, and for any $x \in \mathbb{F}_2^n$, P_j maps x to the jth bit of $P(x)$. Denote

$x = (x_1, x_2)$, where x_1 and x_2 denote, respectively, the first n bits and the last n bits of x. Similarly, let $s = (s_1, s_2)$, $a = (a_1, a_2)$. Then, for any $a = (a_1, a_2) \in \mathbb{F}_2^k$, we have

$$\begin{aligned} &F_s^m(x) \oplus F_s^m(x \oplus a) \\ &\quad = P(m \oplus x_1) \oplus P(m \oplus x_1 \oplus s_1) \oplus P(m \oplus x_1 \oplus a_1) \oplus P(m \oplus x_1 \oplus a_1 \oplus s_1) \oplus a_1. \end{aligned} \tag{8}$$

Thus, for any a satisfying $\lceil a \rceil^n \in \{0, s_1\}$, it holds that $F_s^m(x \oplus a) \oplus F_s^m(x) \equiv \lceil a \rceil^n$, namely, a is a linear structure of F_s^m. In fact, according to Eq. (8), whether a vector $a \in \mathbb{F}_2^k$ is a linear structure of F_s^m depends only on its first n bits. In addition, if we obtain s_1, then we can compute s_2 by $s_2 = P(m \oplus s_1) \oplus E_s(m)$ and thus obtain $s = (s_1, s_2)$. Therefore, when using algorithm **RecoverKey** to find the linear structures of F_s^m (or to find the key s), we only need to consider the first n bits of the vector. We define the parameter as follows:

$$\begin{aligned} \delta_{F_s^m}(s_1) &\triangleq \delta_{F_s^m}(\{b \in \mathbb{F}_2^k | \lceil b \rceil^n = \mathbf{0} \ or \ \ s_1\}) \\ &= \max_j \max_{\substack{a \in \mathbb{F}_2^k \\ \lceil a \rceil^n \notin \{\mathbf{0}, s_1\}}} \max_{i \in \mathbb{F}_2} \frac{|\{x \in \mathbb{F}_2^k | F_{s,j}^m(x \oplus a) + F_{s,j}^m(x) = i\}|}{2^k}. \end{aligned} \tag{9}$$

Here "$\mathbf{0}$" denotes the zero vector in $\mathbb{F}_2^n$. The parameter $\delta_{F_s^m}(s_1)$ is similar to $\delta_{F_s^m}(s)$, except that the maximal about a is taken over the complement of set $\{b \in \mathbb{F}_2^k | \lceil b \rceil^n = \mathbf{0} \text{ or } \ s_1\}$ instead of the complement of set $\{\mathbf{0}, s\}$. Consider the case where $\delta_{F_s^m}(s_1) \leq p_0 < 1$ for some constant p_0, and suppose we run algorithm **RecoverKey** on the Even–Mansour cipher E_s. According to Theorem 2, for any $a \in A_j^{i_j}$, the probability that $\lceil a \rceil^n \notin \{\mathbf{0}, s_1\}$ is negligible. Therefore, we can still use algorithm **RecoverKey** to obtain s_1. To do this, the attacker only needs to omit the last n bits of the solutions when solving the linear equation group $\{x \cdot \omega = i_j | \omega \in W_j\}$. That is, in step 3 of algorithm **RecoverKey**, the attacker defines the set

$$\begin{aligned} \hat{A}_j^{i_j} &= \{\lceil a \rceil^n | a \in A_j^{i_j}\} \\ \bar{A}_j^{i_j} &= \{a_1 = (a_1^1, a_1^2, \ldots, a_1^n) \in \hat{A}_j^{i_j} | a_1^j = i_j\} \\ A_j &= \bar{A}_j^0 \cup \bar{A}_j^i. \end{aligned}$$

In step 4, the attacker still computes the intersection $\bar{A} = A_1 \cap A_2 \cap \cdots \cap A_n$. Then, except for a negligible probability, the nonzero vector in $\bar{A}$ is s_1.

Based on the above analysis, as long as $\delta_{F_s^m}(s_1) \leq p_0$ for some constant $p_0 < 1$, we can apply algorithm **RecoverKey** to the Even–Mansour cipher E_s to recover the key s. Therefore, to prove the feasibility of algorithm **RecoverKey** to the Even–Mansour cipher, we only need to prove that $\delta_{F_s^m}(s_1) \leq p_0 < 1$ for some constant p_0.

Theorem 3 *$\delta_{F_s^m}(s_1) \leq \frac{11}{12}$ holds except for a negligible probability when P is a uniformly random permutation.*

Proof According to Eqs. (8) and (9), we have

$$\begin{aligned}\delta_{F_s^m}(s_1) = & \max_{\substack{a\in\mathbb{F}_2^k \\ \lceil a\rceil^n\notin\{\mathbf{0},s_1\}}} \max_{i,j} \frac{1}{2^k}|\{(x_1,x_2)\in\mathbb{F}_2^k | P^j(m\oplus x_1)\oplus P^j(m\oplus x_1\oplus s_1)\\ & \oplus P^j(m\oplus x_1\oplus a_1)\oplus P^j(m\oplus x_1\oplus a_1\oplus s_1) = i\oplus a_1^j\}|\\ = & \max_{\substack{a\in\mathbb{F}_2^k \\ \lceil a\rceil^n\notin\{\mathbf{0},s_1\}}} \max_{i} \max_{j} \frac{1}{2^k}|\{(x_1,x_2)\in\mathbb{F}_2^k | P^j(x_1)\oplus P^j(x_1\oplus s_1)\\ & \oplus P^j(x_1\oplus a_1)\oplus P^j(x_1\oplus a_1\oplus s_1) = i\oplus a_1^j\}|\\ = & \max_{\substack{a_1\in\mathbb{F}_2^n \\ a_1\notin\{\mathbf{0},s_1\}}} \max_{i,j} \frac{1}{2^n}|\{x_1\in\mathbb{F}_2^n | P^j(x_1)\oplus P^j(x_1\oplus s_1)\oplus P^j(x_1\oplus a_1)\\ & \oplus P^j(x_1\oplus a_1\oplus s_1) = i\}|,\end{aligned}$$

where $a = (a_1, a_2)$, and a_1^j is the jth bit of a_1. Thus, we only need to proof that for any $j \in \{1, 2, \ldots, n\}$, any $i \in \{0, 1\}$, and any $a_1 \in \mathbb{F}_2^n$, such that $a_1 \notin \{\mathbf{0}, s_1\}$,

$$\frac{1}{2^n}\{x_1\in\mathbb{F}_2^n | P^j(x_1)\oplus P^j(x_1\oplus s_1)\oplus P^j(x_1\oplus a_1)\oplus P^j(x_1\oplus a_1\oplus s_1) = i\}| \le \frac{11}{12}$$

holds except for a negligible probability, where P^j is a component of a uniformly random permutation on $\mathbb{F}_2^n$. To prove this, we define an unordered set of four *different* vectors in $\mathbb{F}_2^n$ as a *quadruple*. For example, four different vectors $x, y, z, w \in \mathbb{F}_2^n$ form a quadruple $\{x, y, z, w\}$, and $\{x, y, z, w\} = \{x, y, w, z\} = \{x, z, y, w\} = \cdots = \{w, z, y, x\}$. We use the notation $(\cdot, \cdot, \cdot, \cdot)$ to denote the ordered sequence of four vectors. Then, a quadruple $\{x, y, z, w\}$ corresponds to $A_4^4 = 24$ ordered sequences $(x, y, z, w), (x, y, w, z), (x, z, y, w), \ldots, (w, z, y, x)$. For a quadruple $\{x, y, z, w\}$, its difference is defined as the unordered set of the sums of arbitrary two vectors in the quadruple, that is, $\{x\oplus y, x\oplus z, x\oplus w, y\oplus z, y\oplus w, z\oplus w\}$. Note that the difference is an unordered set of $C_4^2 = 6$ sums, but these six sums need not to be unequal with each other. For any ordered sequence (x, y, z, w), its difference is defined as the difference of its corresponding quadruple $\{x, y, z, w\}$. Thus, the 24 ordered sequences $(x, y, z, w), (x, y, w, z), (x, z, y, w), \ldots, (w, z, y, x)$ have the same difference. For an arbitrary given difference $\{u_1, u_2, \ldots, u_6\}$, suppose there are G quadruples whose differences are equal to it. Then, by simple calculations, we have that $\frac{2^n}{24} \le G \le 2^n$. Using the above notations, we have

$$\begin{aligned}&|\{x_1\in\mathbb{F}_2^n | P^j(x_1)\oplus P^j(x_1\oplus s_1)\oplus P^j(x_1\oplus a_1)\oplus P^j(x_1\oplus a_1\oplus s_1) = i\}|\\ &\quad = |\{(x,y,z,w)\in\mathbb{F}_2^{4n} | P^j(x)\oplus P^j(y)\oplus P^j(z)\oplus P^j(w) = i, x\oplus y = s_1, x\oplus\\ &\qquad z = a_1, x\oplus w = a_1\oplus s_1, y\oplus z = a_1\oplus s_1, y\oplus w = a_1, z\oplus w = s_1\}|\\ &\quad = \frac{1}{24}|\{(x,y,z,w)\in\mathbb{F}_2^{4n} | P^j(x)\oplus P^j(y)\oplus P^j(z)\oplus P^j(w) = i, \text{difference of}\end{aligned}$$

$$
\begin{aligned}
&(x, y, z, w)\text{is } \{s_1, a_1, a_1 \oplus s_1, a_1 \oplus s_1, a_1, s_1\}\}| \\
\leq\ & \frac{1}{24} \times 24|\{\{x, y, z, w\}|P^j(x) \oplus P^j(y) \oplus P^j(z) \oplus P^j(w) = i, \text{difference of} \\
&(x, y, z, w)\text{is } \{s_1, a_1, a_1 \oplus s_1, a_1 \oplus s_1, a_1, s_1\}\}| \\
=\ & |\{\{x, y, z, w\}|P^j(x) \oplus P^j(y) \oplus P^j(z) \oplus P^j(w) = i, \text{difference of } (x, y, z, w) \\
&\text{is } \{s_1, a_1, a_1 \oplus s_1, a_1 \oplus s_1, a_1, s_1\}\}| \qquad (10)
\end{aligned}
$$

We denote the G quadruples with difference $\{s_1, a_1, a_1 \oplus s_1, a_1 \oplus s_1, a_1, s_1\}$ as $\{x_1, y_1, z_1, w_1\}, \{x_2, y_2, z_2, w_2\}, \ldots, \{x_G, y_G, z_G, w_G\}$. Since the sum of the six sums in difference $\{s_1, a_1, a_1 \oplus s_1, a_1 \oplus s_1, a_1, s_1\}$ is equal to 0, we have $x_l \oplus y_l \oplus z_l \oplus w_l = \mathbf{0}$ for $l = 1, 2, \ldots, G$. Moreover, by simple analysis, for any $l \neq q$, $\{x_l, y_l, z_l, w_l\} \cap \{x_q, y_q, z_q, w_q\} = \Phi$. Define random variables $M_l = P^j(x_l) \oplus P^j(y_l) \oplus P^j(z_l) \oplus P^j(w_l)$ for $l = 1, 2, \ldots, G$. Then, $M_1, M_2, \ldots, M_G$ are independent and identically distributed variables. Since P is a uniformly random permutation, we have

$$
\begin{aligned}
\Pr[M_l = 1] &= \frac{1}{2} + \frac{4}{N^2} - \frac{3}{N} < \frac{1}{2} \\
\Pr[M_l = 0] &= \frac{1}{2} - \frac{4}{N^2} + \frac{3}{N} < \frac{5}{6}. \qquad (11)
\end{aligned}
$$

Let $N_l = M_l \oplus i \oplus 1$ for $l = 1, 2, \ldots, n$. Then, $N_l = 1$ if and only if $P^j(x_l) \oplus P^j(y_l) \oplus P^j(z_l) \oplus P^j(w_l) = i$. We let μ be the expectation value of N_l. Then, according to Eq. (11), we have $\mu \leq \frac{5}{6}$. By Hoeffding's inequality, we have

$$
\Pr\left[\frac{1}{G}\sum_{l=1}^{G} N_l - \mu \geq \frac{1}{12}\right] \leq 2\exp\left(-\frac{1}{72}G\right).
$$

Since $\mu \leq \frac{5}{6}$, it holds that

$$
\Pr\left[\frac{1}{G}\sum_{l=1}^{G} N_l - \frac{5}{6} \geq \frac{1}{12}\right] \leq 2\exp\left(-\frac{1}{72}G\right).
$$

Therefore, $\Pr[\frac{1}{G}\sum_{l=1}^{G} N_l \geq \frac{11}{12}] \leq 2\exp(-\frac{1}{72}G)$. Since $\frac{2^n}{24} \leq G \leq 2^n$, we have

$$
\Pr\left[\sum_{l=1}^{G} N_l \geq \frac{11}{12}2^n\right] \leq 2\exp\left(-\frac{1}{72}G\right) \leq 2\exp\left(-\frac{1}{1728}2^n\right).
$$

That is,

$$
\begin{aligned}
&\Pr\Bigg[\frac{1}{2^n}|\{\{x, y, z, w\}|P^j(x) \oplus P^j(y) \oplus P^j(z) \oplus P^j(w) = i, \text{difference of } (x, y, z, w) \\
&\text{is } \{s_1, a_1, a_1 \oplus s_1, a_1 \oplus s_1, a_1, s_1\}\}| \geq \frac{11}{12}\Bigg] \leq 2\exp\left(-\frac{1}{1728}G\right).
\end{aligned}
$$

According to Eq. (10), we have

$$\Pr\left[\frac{1}{2^n}|\{x_1 \in \mathbb{F}_2^n | P^j(x_1) \oplus P^j(x_1 \oplus s_1) \oplus P^j(x_1 \oplus a_1) \oplus P^j(x_1 \oplus a_1 \oplus s_1) = i\}| \geq \frac{11}{12}\right] \leq 2\exp\left(-\frac{1}{1728}G\right),$$

which completes the proof. □

Due to Theorem 3 and previous analysis, we have proved that algorithm **RecoverKey** can efficiently recover the key of Even–Mansour cipher except a negligible probability.

6 Conclusion

We apply the BV algorithm to related-key attack model, and propose a quantum algorithm for recovering the key of block ciphers in quantum related-key attack model. We give the condition under which the attack will succeed and demonstrate that any well-defined block cipher should meet the condition. The attack requires the query to the encryption oracle with superpositions of keys. Query complexity of the algorithm is polynomial. We believe that this work illustrates the power of related-key attack in the quantum setting and provides some guidance for designing quantum-secure block ciphers.

Acknowledgements This research was funded by the National Natural Science Foundation of China (grant no. 61672517), the National Cryptography Development Fund (grant no. MMJJ20170108) and Beijing Municipal Science & Technology Commission (Project Number: Z191100007119006).

References

1. Shor, P. W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of Foundations of Computer Science, Santa Fe, NM, pp. 124–134 (1994)
2. NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016)
3. Grover, L. K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96), Philadelphia, PA, pp. 212–219 (1996)
4. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(5), 1474–1483 (1997)
5. Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In: 2010 IEEE International Symposium on Information Theory Proceedings (ISIT 2010), pp. 2682–2685. Austin, TX, USA (2010)
6. Luby, M., Rackoff, C.: How to construct pseudo-random permutations from pseudo-random functions. In: Williams, H.C. (ed.) Advances in Cryptology-CRYPTO '85 Proceedings LNCS, vol. 218218, pp. 447–447. Springer, Heidelberg (1985)
7. Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: 2012 International Symposium on Information Theory and its Applications (ISITA 2012), pp. 312–316. Honolulu, HI (2012)
8. Santoli, T., Schaffner, C.: Using simon's algorithm to attack symmetric-key cryptographic primitives. Quantum Inf. Comput. **17**(1&2), 65–78 (2017)

9. Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, LNCS, vol. 9815, pp. 207–237. Springer, Heidelberg (2016)
10. Dong, X., Wang, X.: Quantum key-recovery attack on feistel structures. Sci. China Inf. Sci. **61**(10), 102501 (2018)
11. Dong, X., Wang, X.: Quantum cryptanalysis on some generalized Feistel schemes. Sci. China Inf. Sci. **62**(2), 22501 (2019)
12. Zhou, Q., Lu, S., Zhang, Z., Sun, J.: Quantum differential cryptanalysis. Quantum Inf. Process. **14**(6), 2101–2109 (2015)
13. Kaplan, M., Leurent, G., Leverrier, A., & Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. In: FSE 2017, IACR Transactions on Symmetric Cryptology, **2016**(1), pp. 71–94 (2017)
14. Bernstein, E., Vazirani, U.: Quantum complexity theory. SIAM J. Comput. **26**(5), 1411–1473 (1997)
15. Xie, H., Yang, L.: Using Bernstein-Vazirani algorithm to attack block ciphers. Designs, Codes and Cryptography **87**(5), 1161–1182 (2019)
16. Boneh, D., Özgür, D., Fischlin, M., et al.: Random oracles in a quantum world. Advances in Cryptology–ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (2011)
17. Boneh, D., Zhandry, M.: Secure Signatures and Chosen Ciphertext Security in a Quantum Computing World. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology-CRYPTO 2013, LNCS, vol. 8043, pp. 361–379. Springer, Heidelberg (2013)
18. Gagliardoni, T., Hülsing, A., Schaffner, C.: Semantic security and indistinguishability in the quantum world. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology-CRYPTO 2016, LNCS, vol. 9816, pp. 60–89. Springer, Heidelberg (2016)
19. Goldreich, O.: Foundations of Cryptography, Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
20. Bellare, M.; Desai, A.; Jokipii, E.; Rogaway, P.: A concrete security treatment of symmetric encryption. In: Proceedings 38th Annual Symposium on Foundations of Computer Science (FOCS'97), pp. 394–403 (1997)
21. Biham, E.: New types of cryptanalytic attacks using related keys. J. Cryptol. **7**(4), 229–246 (1994)
22. Knudsen, L.R.: Cryptanalysis of LOKI91. In: Seberry, J., Zheng, Y. (eds.) Advances in Cryptology-AUSCRYPT '92 LNCS, vol. 718, pp. 22–35. Springer, Heidelberg (1993)
23. Andreeva, E., Bogdanov, A., Mennink, B.: Towards Understanding the Known-Key Security of Block Ciphers. In: Moriai S. (eds.) Fast Software Encryption. FSE 2013. LNCS, vol 8424, pp. 348–366 Springer, Heidelberg (2014)
24. Alagic, G., Broadbent, A., Fefferman, B., et al.: Computational Security of Quantum Encryption. In: Nascimento, A., Barreto, P. (eds.) Information Theoretic Security, ICITS 2016. LNCS, vol. 10015, pp. 47–71. Springer, Cham (2016)
25. Roetteler, M., Steinwandt, R.: A note on quantum related-key attacks. Inf. Process. Lett. **115**(1), 40–44 (2015)
26. Winternitz, R., Hellman, M.: Chosen-key attacks on a block cipher. Cryptologia **11**(1), 16–20 (1987)
27. Ferguson, N., Kelsey, J., Lucks, S., et al.: Improved cryptanalysis of Rijndael. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) Fast Software Encryption. FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
28. Bellare, M., Kohno, T.: A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications. In: Biham, E. (ed.) Advances in Cryptology-EUROCRYPT 2003, LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)
29. Albrecht, M.R., Farshim, P., Paterson, K.G., et al.: On Cipher-Dependent Related-Key Attacks in the Ideal-Cipher Model. In: Joux, A. (ed.) Fast Software Encryption, FSE 2011. LNCS, vol. 6733, pp. 128–145. Springer, Heidelberg (2011)
30. Biham, E., Dunkelman, O., Keller, N.: A Related-Key Rectangle Attack on the Full KASUMI. In: Roy, B. (ed.) Advances in Cryptology-ASIACRYPT 2005, LNCS, vol. 3788, pp. 443–461. Springer, Heidelberg (2005)
31. Dunkelman, O., Keller, N., Shamir, A.: A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In: Rabin, T. (ed.) Advances in Cryptology-CRYPTO 2010, LNCS, vol. 6223, pp. 393–410. Springer, Heidelberg (2010)
32. Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the Key Scheduling Algorithm of RC4. In: Vaudenay, S., Youssef, A.M. (eds.) Selected Areas in Cryptography, SAC 2001. LNCS, vol. 2259, pp. 1–24. Springer, Heidelberg (2001)

33. Mantin, I.: A Practical Attack on the Fixed RC4 in the WEP Mode. In: Roy, B. (ed.) Advances in Cryptology-ASIACRYPT 2005, LNCS, vol. 3788, pp. 395–411. Springer, Heidelberg (2005)
34. Hosoyamada, A., Aoki, K.: On Quantum Related-Key Attacks on Iterated Even-Mansour Ciphers. In: Obana, S., Chida, K. (eds.) Advances in Information and Computer Security, IWSEC 2017. LNCS, vol. 10418, pp. 3–18. Springer, Cham (2017)
35. Cirac, J.I., Zoller, P.: Quantum computations with cold trapped ions. Phys. Rev. Lett. **74**(20), 4091–4094 (1995)
36. Nakamura, Y., Pashkin, Y.A., Tsai, J.S.: Coherent control of macroscopic quantum states in a single-Cooper-pair box. Nature **398**(6730), 786–788 (1999)
37. Knill, E., Laflamme, R., Milburn, G.J.: A scheme for efficient quantum computation with linear optics. Nature **409**(6816), 46–52 (2001)
38. Gershenfeld, N.A., Chuang, I.L.: Bulk spin-resonance quantum computation. Science **275**(5298), 350–356 (1997)
39. Monz, T., Nigg, D., Martinez, E.A., et al.: Realization of a scalable Shor algorithm. Scienc **351**(6277), 1068–1070 (2016)
40. Lucero, E., Barends, R., Chen, Y., et al.: Computing prime factors with a Josephson phase qubit quantum processor. Nat. Phys. **8**(10), 719–723 (2012)
41. Martin-Lopez, E., Laing, A., Lawson, T., et al.: Experimental realization of Shor's quantum factoring algorithm using qubit recycling. Nat. Photonics **6**(11), 773–776 (2012)
42. Politi, J.C., Matthews, J.L., O'brien, J.L.: Shor's quantum factoring algorithm on a photonic chip. Science **325**(5945), 1221 (2009)
43. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information, 10th edn. Cambridge University Press, Cambridge (2000)
44. Damgård, I., Funder, J., Nielsen, J.B., et al.: Superposition attacks on cryptographic protocols. In: Padró, C. (ed.) Information Theoretic Security, ICITS 2013. LNCS, vol. 8317, pp. 142–161. Springer, Cham (2013)
45. O'connor, L., Klapper, A.: Algebraic nonlinearity and its applications to cryptography. J. Cryptol. **7**(4), 213–227 (1994)
46. Dubuc, S.: Characterization of linear structures. Des. Codes Cryptogr. **22**(1), 33–45 (2001)
47. Li, H., Yang, L.: A quantum algorithm to approximate the linear structures of Boolean functions. Math. Struct. Comput. **28**(1), 1–13 (2018)
48. Tao, B., Wu, H.: Improving the biclique cryptanalysis of AES. In: Australasian Conference on Information Security and Privacy. Springer, Cham, pp. 39–56 (2015)
49. Junod, P.: On the complexity of Matsui's attack. In: International Workshop on Selected Areas in Cryptography. Springer, Berlin, Heidelberg. pp. 199–211 (2001)
50. Chen, H., Wang, X.: Improved linear hull attack on round-reduced Simon with dynamic key-guessing techniques. In: International Conference on Fast Software Encryption. Springer, Berlin, Heidelberg, pp. 428–449 (2016)
51. Faghihi Sereshgi, M.H., Dakhilalian, M., Shakiba, M.: Biclique cryptanalysis of MIBS-80 and PRESENT-80 block ciphers. Secur. Commun. Netw. **9**(1), 27–33 (2015)
52. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. J. Cryptol. **10**(3), 151–162 (1997)