Article

Quantum Truncated Differential and Boomerang Attack

Huiqin Xie and Li Yang

Special Issue
Symmetry in Quantum Optics and Quantum Information Research

Edited by
Prof. Dr. Hong Guo, Dr. Ziyang Chen, Dr. Xiangyu Wang, Prof. Dr. Qiong Li and Dr. Bingjie Xu

MDPI

https://doi.org/10.3390/sym16091124

symmetry

Article

Quantum Truncated Differential and Boomerang Attack

Huiqin Xie [1,2,*] and Li Yang [3,4]

1 Department of Cryptography Science and Technology, Beijing Electronic Science and Technology Institute, Beijing 100070, China
2 Key Laboratory of Cryptography of Zhejiang Province, Hangzhou Normal University, Hangzhou 311121, China
3 Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100085, China
4 Key Laboratory of Cyberspace Security Defense, Beijing 100085, China
* Correspondence: xiehuiqindky@163.com

Abstract: In order to design quantum-safe block ciphers, it is crucial to investigate the application of quantum algorithms to cryptographic analysis tools. In this study, we use the Bernstein–Vazirani algorithm to enhance truncated differential cryptanalysis and boomerang cryptanalysis. We first propose a quantum algorithm for finding truncated differentials, then rigorously prove that the output truncated differentials must have high differential probability for the vast majority of keys in the key space. Subsequently, based on this algorithm, we design a quantum algorithm for finding boomerang distinguishers. The quantum circuits of the two proposed quantum algorithms contain only polynomial quantum gates and qubits. Compared with classical tools for searching truncated differentials or boomerang distinguishers, the proposed algorithms can maintain the polynomial complexity while fully considering the impact of S-boxes and key scheduling.

Keywords: quantum information; quantum cryptanalysis; symmetric cryptography; differential attack; boomerang attack

PACS: 03.67.-a; 03.67.Dd

Citation: Xie, H.; Yang, L. Quantum Truncated Differential and Boomerang Attack. *Symmetry* **2024**, *16*, 1124. https://doi.org/10.3390/sym16091124

Academic Editor: Hung T. Diep

Received: 21 July 2024
Revised: 24 August 2024
Accepted: 28 August 2024
Published: 30 August 2024

1. Introduction

Recently, research on quantum computers has continuously made new progress worldwide. Many scientists, companies and research institutions are committed to utilizing various quantum systems to develop quantum computers. It is foreseen that the successful development of quantum computers will have a profound impact in many fields. Cryptography is one such field.

The two most promising physical implementation schemes for quantum computers are trapped-ion [1] and superconducting circuit [2]. Ion-trap quantum computers have the advantage of great qubit connectivity and small decoherence, while superconducting quantum computers have the advantage of high designability and scalability. In recent years, investigations on ion-trap quantum computers have made great progress [3,4], especially in the improvement of high-fidelity gate [5]. The study of superconducting quantum computers has also made remarkable progress [6–8]. Google's Sycamore quantum computer and IBM's Eagle quantum computer are both based on superconductivity [9,10]. The power of quantum computers in information processing stems from the novel properties of quantum information that differ from those of classical information. Quantum computers possess the natural feature of parallel computing. When an n-qubit quantum computer processes data, operators actually operate on 2^n data states simultaneously. This parallelism may make some problems uncomputable in electronic computers become computable in quantum computers, such as factoring large integers, which is a difficult problem that many public key algorithms are built upon, but may be solved on quantum computers by running Shor's algorithm [11].

The threats posed by quantum computing to symmetric algorithms have also received considerable attention. The most typical example is Grover's algorithm [12], which requires only $O(\sqrt{M})$ complexity to search an unordered database with M elements, while $O(M)$ complexity is required in classical computing. Another important algorithm used to attack symmetric schemes is Simon's algorithm [13]. It was first used for attacking Feistel ciphers [14–16] and EM schemes [14,16]. It was then combined with Grover's algorithm for extracting the keys of ciphers with FX, Feistel and generalized Feistel structures [17–19]. For SPN ciphers, Jaques et al. investigated cryptanalysis of the AES algorithm using Grover's algorithm [20]. Zhang utilized quantum algorithms to attack generalized Feistel ciphers [21]. Xiang introduced a method for constructing periodic functions and used it to attack LBlock cipher [22]. In addition to the aforementioned quantum algorithms, the Bernstein–Vazirani (BV) algorithm [23] was recently utilized in cryptanalysis [24–27].

In addition to specific attack strategies, cryptanalytic tools are also crucial for evaluating the security of cryptosystems. In this field, quantum algorithms were first used for differential cryptanalysis [25,28,29] and then for linear cryptanalysis [26,29,30]. Subsequently, quantum collision attacks on hash functions were studied [31,32]. Denisenko analyzed the complexity of quantum differential attack based on the quantum search algorithm [33]. Hosoyamada used quantum algorithms to speed-up classical multidimensional linear attack [34]. Xu et el. applied quantum search algorithm to differential meet-in-the-middle attack [35]. Quantum attacks under this model were also proposed [36–38]. Zhang proposed a quantum attack under quantum-related key model against the Sum of Even–Mansour construction [39]. Wu and Feng used BV algorithm to search for related-key differentials and recover key based on quantum counting algorithm [40]. These attacks showcased the superiority of quantum cryptanalytic tools over traditional cryptanalytic tools.

Many quantum attacks on block ciphers are too large in scale to be implemented or even simulated. However, researchers may be able to simulate a small part of the whole quantum attacks. For example, Zhou et al. simulated the quantum circuit of S-boxes instead of the whole cipher when studying the quantum circuit of AES [41]. Qiskit SDK is a powerful open-source tool for the simulations of quantum algorithms. Many small-scale quantum algorithms have been simulated using Qiskit [42–44]. LIGHTER-R is another useful tool proposed by Dasu, which can be used to design quantum circuits of Boolean functions [45].

Contributions. In this study, we explore the applications of the Bernstein–Vazirani algorithm to two variants of differential attacks: truncated differential and boomerang attacks. First, we design a quantum algorithm for searching truncated differentials that have a high probability for a large proportion of keys in the key space. Subsequently, based on this algorithm, we construct another quantum algorithm for searching for boomerang distinguishers. We demonstrate the correctness of both quantum algorithms using rigorous proofs. Both quantum algorithms request only polynomial quantum gates and qubits and have the following advantages:

- Quantum adversaries are able to perform the proposed attacks in Q_1 model. Namely, there is no need for quantum queries. Compared to many proposed quantum attack algorithms [14–16,18,19,46] that require quantum queries, our algorithms are easier to implement.
- Classical cryptanalytic tools for finding truncated differentials with high probability or boomerang distinguishers usually cannot concern all of the details of the involved S-boxes when they are not at a small-scale. The classical tools can only search for truncated differentials or boomerang distinguishers of extremely few rounds when the S-boxes have an 8-bit scale, which is very common in block ciphers. By comparison, our quantum algorithms fully utilize the superiority of quantum computing to improve this issue. They entirely characterized the S-boxes through the accurate implementation of the unitary operator of the block ciphers.

- Classical truncated differential attacks do not involve the key scheduling under the single-key attack model, but our algorithms incorporate the key scheduling into the quantum circuits and thus fully reflect its impact to the differential propagation.

Related works. Paul et al. combined classical boomerang attack with Grover's algorithm to quantize the traditional boomerang attack [47]. Zhou et al. improved this quantum attack strategy by allowing the retrieval of subkeys from both sides of block ciphers [48]. Boomerang attack includes two stages. The first stage is to find boomerang distinguishers and the second stage is to recover the key using the found distinguishers. The works in [47,48] only focused on the second stage. Both of them studied the use of quantum algorithms for accelerating the retrieval of key. Our work focuses on the first stage and studies the use of quantum algorithms for finding boomerang distinguishers.

2. Preliminaries

The main notations and their definitions are presented in Table 1.

Table 1. Notations.

Notation	Definition
$\mathcal{C}_{u,v}$	the set containing all Boolean functions mapping u bits to v bits
Φ	the empty set
$S_f(\cdot)$	the Walsh transform of f
Enc_k^t	a t-round block cipher
$Enc_k^t[j]$	the j-th component function of Enc_k^t
S/N	the ratio of signal to noise
$(\Delta x, \Delta y)$	a differential
$(\overline{\Delta} x, \overline{\Delta} y)$	a truncated differential

2.1. Differential

Throughout this study, $Enc_k : \mathbb{F}_2^n \to \mathbb{F}_2^n$ denotes a block cipher, where $k \in \mathbb{F}_2^m$ is the master key. $Enc_k^r = Enc_{k_r} \circ Enc_{k_{r-1}} \circ \cdots \circ Enc_{k_1}$ denotes the r-round iteration of Enc_k. Here $k_1, \cdots, k_r$ denote the round keys generated from k according to the key scheduling.

Suppose x and x' are two plaintexts, and $Enc_k^r(x) = y$, $Enc_k^r(x') = y'$. We call $\Delta y = y' \oplus y$ an output difference and $\Delta x = x' \oplus x$ an input difference. $(\Delta x, \Delta y)$ is defined as a differential of Enc_k^r. The probability of differential $(\Delta x, \Delta y)$ is defined as

$$\begin{aligned}\Pr_{x \leftarrow \mathbb{F}_2^n}[\Delta x \xrightarrow{Enc_k^r} \Delta y] &= \Pr_{x \leftarrow \mathbb{F}_2^n}[Enc_k^r(x \oplus \Delta x) \oplus Enc_k^r(x) = \Delta y] \\ &= \frac{1}{2^n}|\{x \in \mathbb{F}_2^n | Enc_k^r(x \oplus \Delta x) \oplus Enc_k^r(x) = \Delta y\}|.\end{aligned}$$

If this value is equal to p, we call $(\Delta x, \Delta y)$ a p-probability differential of Enc_k^r.

Differential attack was proposed in 1991 and is one of the most commonly used cryptanalysis methods [49]. It utilizes a high-probability differential to break block ciphers. Let $Enc_k^t = Enc_{k_t} \circ Enc_{k_{t-1}} \circ \cdots \circ Enc_{k_1}$ be the t-round iteration of Enc_k, where $1 < t < r$. Enc_k^t is a reduced cipher of Enc_k^r. In differential attacks the adversaries first search for a differential of Enc_k^t having high probability, then use this differential to screen out the right subkey involved in the last $r - t$ rounds of Enc_k^r.

Variants of differential cryptanalysis have been proposed, including impossible differential attacks [50], truncated differential attacks [51] and boomerang attacks [52]. These attacks all utilize the no-random statistical properties of the ciphertext differences when specifying the plaintext differences.

Inspired by the concept of differential of block ciphers, we define the differential of Boolean functions. Let $\mathcal{C}_{u,v}$ be a set containing all Boolean functions that map u bits to v bits, where u, v are arbitrary positive integers. For any $f \in \mathcal{C}_{u,v}$ and $x, x' \in \mathbb{F}_2^u$, let $f(x) = y \in \mathbb{F}_2^v$ and $f(x') = y' \in \mathbb{F}_2^v$. We call $\Delta y = y' \oplus y$ an output difference and $\Delta x = x' \oplus x$ an input difference. $(\Delta x, \Delta y)$ is defined as a differential of f. The probability of differential $(\Delta x, \Delta y)$ is defined as

$$\Pr_{x \leftarrow \mathbb{F}_2^u}[\Delta x \xrightarrow{f} \Delta y] = \Pr_{x \leftarrow \mathbb{F}_2^u}[f(x \oplus \Delta x) \oplus f(x) = \Delta y].$$

If this value is equal to p, we call $(\Delta x, \Delta y)$ a p-probability differential of f. Especially, for any function f in $\mathcal{C}_{u,1}$, we define two sets

$$D_f^0 = \{\Delta x \in \mathbb{F}_2^u | f(x) \oplus f(x \oplus \Delta x) = 0, \forall x \in \mathbb{F}_2^u\},$$
$$D_f^1 = \{\Delta x \in \mathbb{F}_2^u | f(x) \oplus f(x \oplus \Delta x) = 1, \forall x \in \mathbb{F}_2^u\}.$$

Let $D_f = D_f^0 \cup D_f^1$. The vectors in D_f are called complete differentials of f. For any vector $\Delta x \in D_f^i$ $(i \in \{0,1\})$, $(\Delta x, i)$ is obviously a 1-probability differential. For any $\Delta x \notin D_f^i$, Δx cannot form a 1-probability differential of f as an input difference.

2.2. Quantum Computing

In quantum computing theory, information is stored in qubits. A qubit can be realized by any two-level quantum system, such as polarized photons. It is an analogue of a classical bit, but besides the states $|0\rangle$ and $|1\rangle$, it can also be in a state that is a linear combination of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β can be any complex numbers that satisfy $|\alpha|^2 + |\beta|^2 = 1$. It is usually called a superposition state. Multiple qubits are combined via tensor product. Suppose the first qubit is in the state $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$, the second qubit is in the state $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$, then the system of these two qubits is in the state

$$\begin{aligned} |\psi_1\rangle \otimes |\psi_2\rangle &= (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle) \\ &= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle. \end{aligned}$$

Classical bits can be manipulated by logic gates such as NAND and AND gates. Similarly, qubits are manipulated by quantum gates. Common quantum gates include Pauli-X gate (X gate), Phase gate (S gate), Hadamard gate (H gate), Controlled-NOT gate ($CNOT$ gate) and Toffoli gate. These gates act according to following rules:

$$\begin{array}{ll} X|0\rangle = |1\rangle, & X|1\rangle = |0\rangle, \\ S|0\rangle = |0\rangle, & S|1\rangle = e^{i\pi/4}|1\rangle, \\ H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), & H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \end{array}$$
$$CNOT|a\rangle|b\rangle = |a\rangle|b \oplus a\rangle, \quad \forall a, b \in \{0,1\},$$
$$Toffoli|a\rangle|b\rangle|c\rangle = |a\rangle|b\oplus\rangle|c \oplus ab\rangle, \quad \forall a, b, c \in \{0,1\}.$$

These quantum gates are represented in quantum circuits as in Figure 1. A collection of quantum gates interlinked by quantum wires is called a quantum circuit.

Define $H^{\otimes n} = H \otimes H \otimes \cdots \otimes H$. $H^{\otimes n}$ is the tensor product of n Hadamard gates and

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbb{F}_2^n} |x\rangle.$$

The 2^n states $|00\cdots00\rangle, |00\cdots01\rangle, \cdots, |11\cdots10\rangle, |11\cdots11\rangle$ are all called computational basis states. Applying quantum gate $H^{\otimes n}$ on a initial state $|0\rangle^{\otimes n}$ yields a superposition of all computational basis states.

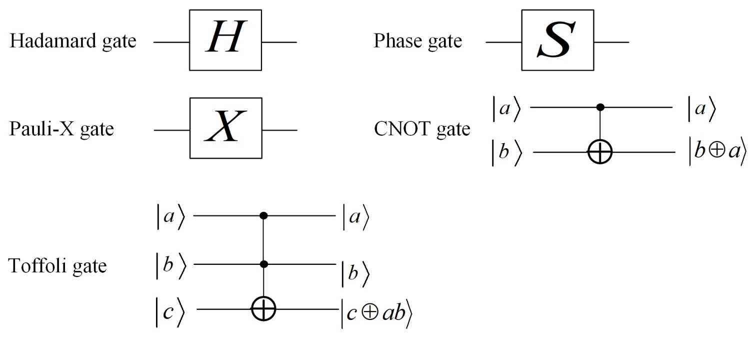

Figure 1. The notation of common quantum gates.

For any $f \in \mathcal{C}_{u,v}$, a quantum circuit realizes f is equivalent to realizing the following operator

$$U_f : \sum_{x,y} |x\rangle|y\rangle \to \sum_{x,y} |x\rangle|y \oplus f(x)\rangle.$$

U_f can be integrated into quantum circuits as presented in Figure 2.

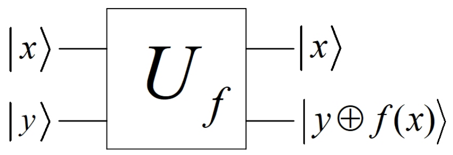

Figure 2. Quantum gate U_f.

When quantum state $\sum_{x\in\mathbb{F}_2^n} \alpha_x|x\rangle$ is measured on the computational basis states, the probability of outputting x is equal to $|\alpha_x|^2$. If $\alpha_x = 0$, the output is definitely not x. Any block cipher Enc^r can be efficiently realized using a quantum circuit. Namely, there is a quantum circuit with polynomial-complexity taking a state of plaintexts and master keys as input, and outputting the corresponding ciphertexts, realizing the unitary operator

$$U_{Enc^r} : \sum_{\substack{k\in\mathbb{F}_2^m \\ x,y\in\mathbb{F}_2^n}} |k\rangle|x\rangle|y\rangle \to \sum_{\substack{k\in\mathbb{F}_2^m \\ x,y\in\mathbb{F}_2^n}} |k\rangle|x\rangle|y \oplus Enc_k^r(x)\rangle.$$

All quantum circuits can be realized using only the gates in some universal gate set [53], such as $\{\frac{\pi}{8}, CNOT, Phase, H\}$. Thus, Enc^r can be realized using a quantum circuit containing only polynomial universal gates. Let the total amount of quantum universal gates in this circuit be $|Enc^r|_Q$. U_{Enc^r} can be integrated into the quantum circuits as shown in Figure 3.

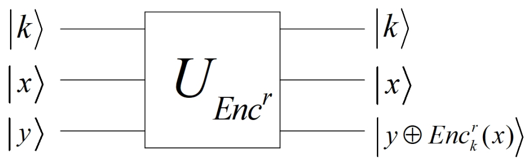

Figure 3. Quantum gate U_{Enc^r}.

Two models have been proposed to describe quantum adversaries: the Q_1 and Q_2 models [54–56]. Q_1 adversaries can perform local quantum operations but can merely make queries classically to the cryptography primitives. In addition to classical queries and local quantum operations, Q_2 adversaries can also query the quantum oracles of the cryptography primitives. Q_2 model is more demanding because it is difficult to achieve the quantum oracles of cryptography primitives in practice.

2.3. Bernstein–Vazirani Algorithm

Bernstein–Vazirani (BV) algorithm [23] was designed to solve the problem: $s \in \{0,1\}^n$ being a secret vector, with a quantum circuit of function $f(x) = s \cdot x$, how to obtain the value of s. BV algorithm runs as follows:

1. Implement Hadamard transform $H^{(n+1)}$ on $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$, giving
$$|\psi_1\rangle = \sum_{x\in\mathbb{F}_2^n} \frac{|x\rangle}{\sqrt{2^n}} \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$
2. Using the quantum circuit of f to get
$$|\psi_2\rangle = \sum_{x\in\mathbb{F}_2^n} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$
3. Discarding the unentangled last qubit, perform Hadamard operator $H^{(n)}$ on the remaining qubits, getting
$$|\psi_3\rangle = \sum_{y\in\mathbb{F}_2^n} (\frac{1}{2^n} \sum_{x\in\mathbb{F}_2^n} (-1)^{f(x)+y\cdot x})|y\rangle. \tag{1}$$
Since $f(x) = s \cdot x$, we have
$$|\psi_3\rangle = \sum_{y\in\mathbb{F}_2^n} (\frac{1}{2^n} \sum_{x\in\mathbb{F}_2^n} (-1)^{(s\oplus y)\cdot x})|y\rangle = |s\rangle.$$
Therefore, measuring $|\psi_3\rangle$ gives the value of s.

For any function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ in $\mathcal{C}_{n,1}$, the Walsh transform is defined as the function

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

Equation (1) shows that, if BV algorithm is run on a general function $f \in \mathcal{C}_{n,1}$, the final quantum state without measured will be

$$\sum_{y\in\mathbb{F}_2^n} S_f(y)|y\rangle,$$

where $S_f(\cdot)$ is the Walsh transform of f. When this state is measured, the probability of $y \in \mathbb{F}_2^n$ being output is $S_f(y)^2$. Thus, BV algorithm running on f must output y such that $S_f(y) \neq 0$.

Figure 4 shows the circuit of BV algorithm. BV algorithm needs totally $2n + 1 + |f|_Q$ universal gates. The corresponding quantum circuit requires $n + 1$ qubits.

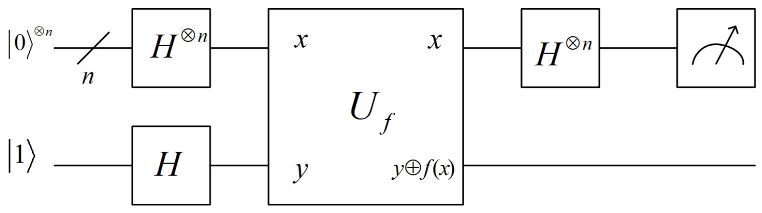

Figure 4. Quantum circuit of BV algorithm.

Utilizing the fact that BV algorithm always outputs the vectors in the support set of the Walsh transform, Li et al. constructed a quantum algorithm used for finding differentials with high probability [57].

For any $f \in \mathcal{C}_{n,1}$, let

$$\begin{aligned}\gamma_f &= \frac{1}{2^n} \max_{\substack{\Delta x \in \mathbb{F}_2^n \\ \Delta x \notin D_f}} \max_{i \in \{0,1\}} |\{x \in \mathbb{F}_2^n | f(x \oplus \Delta x) \oplus f(x) = i\}| \\ &= \max_{\substack{\Delta x \in \mathbb{F}_2^n \\ \Delta x \notin D_f}} \max_{i \in \{0,1\}} \Pr_{x \leftarrow \mathbb{F}_2^n} [\Delta x \xrightarrow{f} i].\end{aligned}$$

It is easy to verified that $\gamma_f < 1$. This parameter is the maximum differential probability of f except for the probability-1 differentials. The authors of [25,57] proved the following theorems that illustrate the soundness of the aforementioned algorithm.

Theorem 1 ([57])**.** *If Algorithm 1 outputs two sets Z^0 and Z^1 when applied to a function $f \in \mathcal{C}_{n,1}$, then for any $\Delta x \in Z^i$ $(i = 0, 1)$, any ϵ satisfying $0 < \epsilon < 1$, it holds that*

$$\Pr\left[1 - \frac{|\{x \in \mathbb{F}_2^n | f(x \oplus \Delta x) + f(x) = i\}|}{2^n} < \epsilon\right] > 1 - e^{-2q(n)\epsilon^2}. \tag{2}$$

Algorithm 1 Quantum algorithm for finding high-probability differentials.

Input: the quantum circuit realizing $f \in \mathcal{C}_{n,1}$, a polynomial $q(n)$ of n.
Output: a differential of function f.

```
1: Define a set $W := \Phi$;
2: for $l = 1, 2, \cdots, q(n)$ do
3:     Apply BV algorithm to $f$, obtaining an vector $u$ with $n$ bits such that $S_f(u) \neq 0$;
4:     Let $W = W \cup \{u\}$;
5: end for
6: Solve the equation $\{x \cdot u = i | u \in W\}$, getting the solution set $Z^i$ for both $i = 0, 1$;
7: if $Z^0 \cup Z^1 \subseteq \{\vec{0}\}$ then
8:     Output "No";
9: else
10:    Output $Z^0$ and $Z^1$;
11: end if
```

Theorem 2 ([25])**.** *Suppose $f \in \mathcal{C}_{n,1}$ and there is a constant a_0 such that $\gamma_f \leq a_0 < 1$. If Algorithm 1 outputs two sets Z^0 and Z^1 when applied to f with $q(n) = n$, then for any vector $\Delta x \notin D_f^i$ $(i = 0, 1)$, we have*

$$\Pr[\Delta x \in Z^i] \leq a_0^n.$$

Theorem 1 demonstrates that, for any vector $\Delta x \in Z^i$ $(i = 0, 1)$, the differential probability of $(\Delta x, i)$ to f is greater than $1 - \epsilon$ with a probability greater $1 - e^{-2q(n)\epsilon^2}$.

3. Quantum Truncated Differential Attack

Knudsen introduced the truncated differential attack in 1994 [51]. This cryptanalytic method has been widely applied to attack symmetric ciphers [58,59]. In the initial version of differential attacks, the adversaries utilize full differences of plaintexts and ciphertexts, whereas in truncated differential attacks the adversaries consider differences partially determined. The adversaries only predict some bits of the differentials rather than the entire differentials.

We still consider the block cipher Enc_k^r with the key space $\mathbb{F}_2^m$. A truncated differential $(\overline{\Delta}x, \overline{\Delta}y)$ of Enc_k^r is a pair of vectors such that $\overline{\Delta}x, \overline{\Delta}y \in \{*, 0, 1\}^n$, where $*$ denotes an undetermined bit. Let $\overline{\Delta}x = (\overline{\Delta}x_1, \cdots, \overline{\Delta}x_n)$, $\overline{\Delta}y = (\overline{\Delta}y_1, \cdots, \overline{\Delta}y_n)$. then $\overline{\Delta}x_i, \overline{\Delta}y_i \in \{*, 0, 1\}$. The bits of $\overline{\Delta}x$ ($\overline{\Delta}y$) that take the value of zero or one are defined as predicted bits, whereas those with a value of $*$ are defined as unpredicted bits.

A truncated difference is equivalent to a set of complete differences. Define

$$\Omega_{\overline{\Delta}x} = \left\{\Delta x = (\Delta x_1, \cdots, \Delta x_n) \in \mathbb{F}_2^n \middle| \Delta x_i = \overline{\Delta}x_i \text{ if } \overline{\Delta}x_i \neq *, i \in \{1, 2, \cdots, n\}\right\},$$

$$\Omega_{\overline{\Delta}y} = \left\{\Delta y = (\Delta y_1, \cdots, \Delta y_n) \in \mathbb{F}_2^n \middle| \Delta y_i = \overline{\Delta}y_i \text{ if } \overline{\Delta}y_i \neq *, i \in \{1, 2, \cdots, n\}\right\},$$

then truncated differences $\overline{\Delta}x$ and $\overline{\Delta}y$ are equivalent to $\Omega_{\overline{\Delta}x}$ and $\Omega_{\overline{\Delta}y}$, respectively. If a complete input difference Δx is in $\Omega_{\overline{\Delta}x}$, that is, $\Delta x_j = \overline{\Delta}x_j$ for all $j \in \{1, \cdots, n\}$ such that $\overline{\Delta}x_j \neq *$, we say that Δx matches the truncated difference $\overline{\Delta}x$, and this case is denoted as $\Delta x \sim \overline{\Delta}x$. Similarly, $\Delta y \sim \overline{\Delta}y$ implies that Δy matches the truncated difference $\overline{\Delta}y$.

Conditional probability

$$\begin{aligned}\Pr_{x \leftarrow \mathbb{F}_2^n}[\overline{\Delta}x \overset{Enc_k^r}{\rightarrow} \overline{\Delta}y] &= \Pr_{x \leftarrow \mathbb{F}_2^n}[Enc_k^r(x \oplus \Delta x) \oplus Enc_k^r(x) \sim \overline{\Delta}y | \Delta x \sim \overline{\Delta}x] \\ &= \Pr_{x \leftarrow \mathbb{F}_2^n}[Enc_k^r(x \oplus \Delta x) \oplus Enc_k^r(x) \in \Omega_{\overline{\Delta}y} | \Delta x \in \Omega_{\overline{\Delta}x}]\end{aligned}$$

is defined as the probability of $(\overline{\Delta}x, \overline{\Delta}y)$. If p is equal to the probability of $(\overline{\Delta}x, \overline{\Delta}y)$, we call $(\overline{\Delta}x, \overline{\Delta}y)$ a p-probability truncated differential of Enc_k^r.

Let Enc^t $(1 < t < r)$ be a reduced cipher of Enc^r. In a truncated differential attack, the adversaries first search for a truncated differential of Enc^t that has a high probability and then use this truncated differential, denoted as $(\overline{\Delta}x, \overline{\Delta}y)$, to recover the subkeys involved in the last $r - t$ rounds. In detail, the adversaries fix the plaintext difference $\overline{\Delta}x$ and then use $2M$ pairs of plaintexts, whose differences match $\overline{\Delta}x$, to make encryption queries and obtain $2M$ pairs of corresponding ciphertexts. Subsequently, for each possible candidate subkey of the last $r - t$ rounds, the adversaries use it to decrypt $r - t$ rounds to obtain M output differences of Enc_k^t, in the meantime calculate the amount of the differences that match $\overline{\Delta}y$. Finally, the right subkey is the subkey having the maximum count.

The amount of plaintext pairs required in such a counting scheme and the success probability of obtaining the right key are determined by the ratio of signal to noise [49], and its definition is

$$S/N = \frac{L \times p}{\alpha \times \lambda},$$

where L denotes the total amount of possible subkeys involved in the last $r - t$ rounds, p denotes the probability of $(\overline{\Delta}x, \overline{\Delta}y)$, α denotes the average count that every plaintext pair contributes and λ denotes the proportion of pairs not discarded in the preprocessing procedure. We do not consider any pre-discarding processes, therefore we set $\lambda = 1$. A truncated differential attack succeeds only when $S/N > 1$. Thus, the adversaries should use a truncated differential that makes the ratio of signal to noise greater than one. The greater S/N is, the easier it is to single out the right subkey.

In the following, we propose a quantum algorithm used for finding truncated differentials. In a classical truncated differential attack, because the adversaries do not know the value of k of the reduced cipher Enc_k^t, they must find a truncated differential whose probability is high regardless of the value of the key k. Therefore, our quantum algorithm is designed to search for truncated differentials that have high probability for a large proportion of keys in $\mathbb{F}_2^m$. Specifically, by choosing a polynomial $\tau(n)$, the adversaries can force our quantum algorithm to output truncated differentials that have a high probability for more than $(1 - \frac{1}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$. We present the algorithm and analyze its effectiveness and complexity.

3.1. Finding Truncated Differentials via BV Algorithm

Given a reduced block cipher Enc_k^t, let $Enc_k^t(x) = (Enc_k^t[1](x), \cdots, Enc_k^t[n](x))$. That is, $Enc_k^t[j]$ denotes the j-th component function of Enc_k^t. An intuitive method for finding high-probability truncated differentials is to implement Algorithm 1 on every $Enc_k^t[j]$. If Algorithm 1 finds differentials of several component functions that all have high probability and have a common input difference, then we can derive a truncated differential of Enc_k^t that has high probability. However, running Algorithm 1 on $Enc_k^t[j]$ requires quantum queries of Enc_k^t. It is impossible to achieve this even under Q_2 model because Enc_k^t is a reduced cipher instead of the complete cipher Enc_k^r. In the original differential attack, the adversaries are also not able to query the reduced version. They thus analyzed the detailed constructions of the cipher and searched for truncated differentials whose probabilities were high regardless of the value the key took. Inspired by this idea, we consider searching for the truncated differentials with a high probability for most keys.

Since all constructions of the cipher Enc_k^t, except for the private key k, are public, the function

$$\begin{aligned} Enc^t : \{0,1\}^n \times \{0,1\}^m &\longrightarrow \{0,1\}^n \\ (\ x\ \ ,\ \ k\ ) &\longrightarrow Enc_k^t(x) \end{aligned}$$

take the key as the input and is known and determined to the adversaries. Thus, the adversaries have access to the quantum circuit of the unitary operator

$$U_{Enc^t} : |x\rangle|k\rangle|y\rangle \to |x\rangle|k\rangle|y \oplus Enc^t(x,k)\rangle = |x\rangle|k\rangle|y \oplus Enc_k^t(x)\rangle.$$

Let $|Enc^t|_Q$ be the amount of quantum universal gates in this circuit. The adversaries also have the quantum circuit of every component function

$$\begin{aligned} Enc^t[j] : \{0,1\}^n \times \{0,1\}^m &\longrightarrow \{0,1\}^n \\ (\ x\ \ ,\ \ k\ ) &\longrightarrow Enc_k^t[j](x). \end{aligned}$$

The corresponding amount of gates is $|Enc^t[j]|_Q$ ($j = 1, \cdots, n$). The adversaries have the quantum circuits of $Enc^t[j]$'s. Therefore, they can run Algorithm 1 on $Enc^t[j]$'s without quantum queries. The adversaries can run Algorithm 1 to obtain the differentials of high probability of every $Enc^t[j]$, then by taking a common input difference of part component functions as the input difference, they can obtain a truncated differential having high probability. According to this idea, we propose Algorithm 2 for finding truncated differentials of block ciphers.

The flowchart of Algorithm 2 is presented in Figure 5. Steps 1–18 of Algorithm 2 are used to determine the high-probability differentials of $Enc^t[j]$ for every $j = 1, \cdots, n$. The purpose of steps 19–26 is to choose a difference which is a common input difference of as many $Enc^t[j]$ as possible. Algorithm 2 outputs a truncated differential (a, b) of Enc^t. The symbol "$*$" in b means that the corresponding bits are unpredicted. In a quantum truncated differential attack, the adversaries first choose a polynomial $\tau(n)$ and a constant σ $(0 < \sigma < 1)$, then implement Algorithm 2 to get an output (a, b). According to Theorem 3 which is proven in Section 3.2, the differential probability of (a, b) is greater than σ for more than $(1 - \frac{1}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$ with an overwhelming probability.

Algorithm 2 Quantum algorithm for finding high-probability truncated differentials

```
Input:  The quantum circuit of $Enc^t$, a polynomial $\tau(n)$ and a constant $\sigma$ $(0 < \sigma < 1)$
        chosen by the adversaries.
Output: a high-probability truncated differential of $Enc^t$.
1:  Let $q(n) = \frac{1}{2(1-\sigma)^2}\tau(n)^2 n^3$;
2:  Define a set $W := \Phi$;
3:  for $j = 1, 2, \cdots, n$ do
4:      for $l = 1, \cdots, q(n)$ do
5:          Apply BV algorithm to $Enc^t[j]$ to get an output $u = (u_1, \cdots, u_n, u_{n+1}, \cdots, u_{n+m})$;
6:          Let $W = W \cup \{(u_1, \cdots, u_n)\}$;
7:      end for
8:      Solve the linear equation $\{x \cdot u = i_j | u \in W\}$ by Gaussian elimination method,
        obtaining the solution sets $Z_j^{i_j}$ for $i_j = 0, 1$, respectively;
9:      Compute the set $Z_j = Z_j^0 \cup Z_j^1$;
10:     Let $\overline{Z}_j = \Phi$;
11:     for $a \in Z_j^0$ do
12:         Let $\overline{Z}_j = \overline{Z}_j \cup \{(a, 0)\}$;
13:     end for
14:     for $a \in Z_j^1$ do
15:         Let $\overline{Z}_j = \overline{Z}_j \cup \{(a, 1)\}$;
16:     end for
17:     Let $W = \Phi$;
18: end for
19: for $d = n, n-1 \cdots, 1$ do
20:     if $S/N = 2^d\sigma > 1$ then
21:         if there are $d$ different subscripts $j_1, \cdots, j_d$ s.t. $Z_{j_1} \cap \cdots \cap Z_{j_d} \supsetneq \{\vec{0}\}$ then
22:             Choose at random a vector $a \in Z_{j_1} \cap \cdots \cap Z_{j_d}$, and for $j = 1, \cdots, n$, let
                $$b_j = \begin{cases} i_j, & j \in \{j_1, \cdots, j_d\} \\ *, & j \notin \{j_1, \cdots, j_d\}, \end{cases}$$
                where $i_j$ denotes the bit appended to $a$ in the set $\overline{Z}_j$, i.e., $(a, i_j) \in \overline{Z}_j$;
23:             Let $b = (b_1, \cdots, b_n)$ and return $(a, b)$;
24:         end if
25:     end if
26: end for
27: Return "No";
```

To implement steps 21-22, the adversaries traverse the variables $j_1, j_2, \cdots, j_d$ in sequence. For $j_1 = 1, 2, \cdots, n-d+1, j_2 = j_1, j_1+1, \cdots, n-d+2, \cdots, j_d = j_{d-1}, j_{d-1}+1, \cdots, n$, Algorithm 2 needs to compute the intersection of the sets $Z_{j_1}, Z_{j_2}, \cdots, Z_{j_d}$. If the intersection contains nonzero vectors, Algorithm 2 randomly chooses a nonzero vector and outputs it.

In order to demonstrate the feasibility of the output truncated differential (a, b), it is necessary to compute the ratio of signal to noise S/N. To this end, we first calculate the parameter α, which is equal to the average count that every plaintext pair contributes. There are d bits of the difference b predicted, therefore a total of 2^{n-d} output differences matching the truncated difference b. In the counting process, the ciphertexts of a fixed pair of plaintexts are decrypted using L candidate subkeys. The resulting L output differences can be viewed as random vectors. Therefore, every plaintext pair contributes

$$\alpha = \frac{2^{n-d}}{2^n} \times L = \frac{L}{2^d}$$

counts on average. Then

$$N/S \geq \frac{L \times \sigma}{\frac{L}{2^d} \times 1} = 2^d \sigma > 1.$$

This value is greater than one because of the condition $2^d \sigma > 1$ in the step 14 of Algorithm 2. After obtaining the output (a, b), the adversaries can utilize it to find the right subkey involved in the last $r - t$ rounds, similar to the traditional truncated differential attack. This attack should work for at least $(1 - \frac{1}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$. Even if "No" is output, the adversaries can adjust the polynomial $\tau(n)$ and σ to increase the success probability.

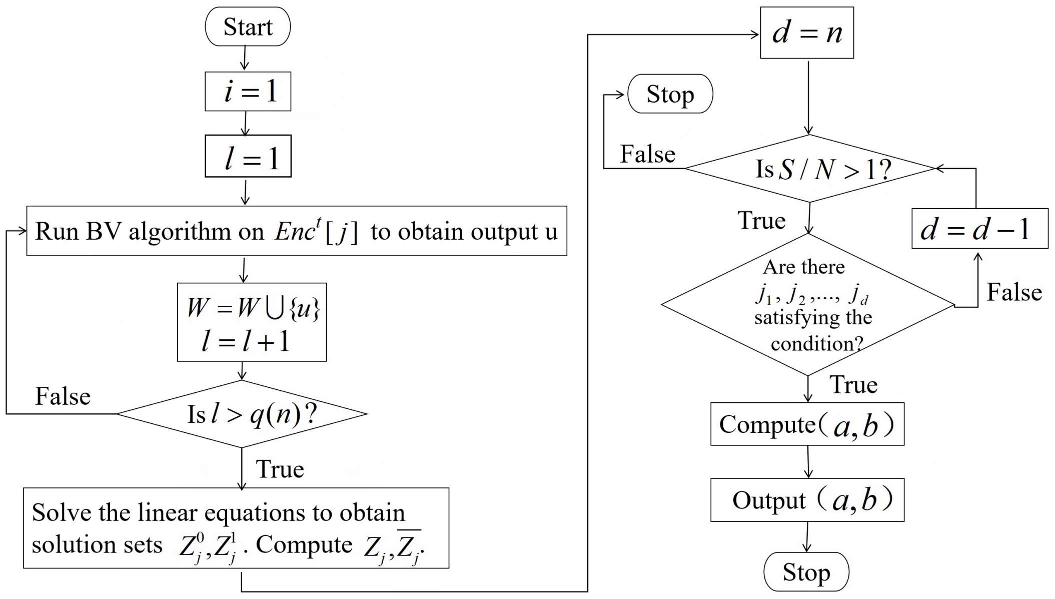
Start
$i = 1$
$l = 1$
Run BV algorithm on $Enc^t[j]$ to obtain output u
$W = W \cup \{u\}$
$l = l + 1$
False
Is $l > q(n)$?
True
Solve the linear equations to obtain solution sets Z_j^0, Z_j^1. Compute $Z_j, \overline{Z_j}$.
$d = n$
Stop
False
Is $S/N > 1$?
True
$d = d - 1$
Are there $j_1, j_2, \ldots, j_d$ satisfying the condition?
False
True
Compute(a, b)
Output (a, b)
Stop

Figure 5. The flowchart of Algorithm 2.

3.2. Analysis of Algorithm 2

We analyze the correctness and efficiency of Algorithm 2. Theorem 3 indicates the correctness of Algorithm 2.

Theorem 3. *Suppose Algorithm 2 outputs (a, b), then with an overwhelming probability, there is a subset $S \subseteq \mathbb{F}_2^m$ satisfying that $|S|/|\mathbb{F}_2^m| > 1 - \frac{1}{\tau(n)}$, and for every key $k \in S$,*

$$\frac{|\{x \in \mathbb{F}_2^n | Enc_k^t(x \oplus a) + Enc_k^t(x) \sim b\}|}{2^n} > \sigma.$$

That is, the differential probability of (a, b) is greater than σ for more than $(1 - \frac{1}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$.

Proof. b has d predicted bits, whose subscripts are $j_1, \cdots, j_d$. Appending m zeros after the vector a gives an $(n + m)$-bit vector $(a \| 0, \cdots, 0)$. Since $a \cdot (u_1, \cdots, u_n) = 0$, it holds that

$$(a \| 0, \cdots, 0) \cdot (u_1, \cdots, u_n, u_{n+1}, \cdots, u_{n+m}) = 0.$$

The $(n + m)$-bit vector $(a \| 0, \cdots, 0)$ can be viewed as the output of Algorithm 2 when it is applied to $Enc^t[j]$ for all $j \in \{j_1, j_2, \cdots, j_d\}$. From Theorem 1, the probability that

$$\frac{|\{z \in \mathbb{F}_2^{n+m} | Enc^t[j](z \oplus (a \| 0, \cdots, 0)) \oplus Enc^t[j](z) = b_j)\}|}{2^{n+m}} > 1 - \epsilon, \ \forall j \in \{j_1, j_2, \cdots, j_d\}$$

holds is greater than $(1-e^{-2q(n)\epsilon^2})^d$. If the above inequality holds, then the number of z that satisfies

$$Enc^t[j]\big(z \oplus (a\|0,\cdots,0)\big) \oplus Enc^t[j](z) = b_j \tag{3}$$

for both $j = j_1$ and $j = j_2$ is greater than $2^{n+m}[2(1-\epsilon)-1] = 2^{n+m}(1-2\epsilon)$. Likewise, the number of z satisfying Equation (3) for all $j = j_1, j_2, j_3$ is greater than $2^{n+m}(1-3\epsilon)$. By induction, the number of z that satisfies Equation (3) for all $j \in \{j_1, j_2, \cdots, j_d\}$ is more than $2^{n+m}(1-d\epsilon)$. Therefore, the probability that

$$\frac{|\{z \in \mathbb{F}_2^{n+m} | Enc^t(\, z \oplus (a\|0,\cdots,0)\,) \oplus Enc^t(z) \sim b)\}|}{2^{n+m}} > 1 - d\epsilon.$$

holds is greater than $(1-e^{-2q(n)\epsilon^2})^d$, which is equivalent to

$$\frac{|\{(x,k) \in \mathbb{F}_2^n \times \mathbb{F}_2^m | Enc_k^t(x \oplus a) \oplus Enc_k^t(x) \sim b\}|}{2^{n+m}} > 1 - d\epsilon. \tag{4}$$

Let

$$Z(k) = \frac{|\{x \in \mathbb{F}_2^n | Enc_k^t(x \oplus a) + Enc_k^t(x) \sim b\}|}{2^n}.$$

Equation (4) indicates that $\mathbb{E}_k[Z(k)] > 1 - d\epsilon$. Here $\mathbb{E}_k[Z(k)]$ is the statistical expectation of $Z(k)$ and the variable k follows the uniform distribution of $\mathbb{F}_2^m$. Therefore, when Equation (4) holds, we have

$$\Pr_k\big[\, Z(k) > 1 - \tau(n)d\epsilon \,\big] > 1 - \frac{1}{\tau(n)}$$

for any polynomial $\tau(n)$. This is because, if not, then $\Pr_{k \leftarrow \mathbb{F}_2^m}[1 - Z(k) \geq \tau(n)d\epsilon] \geq \frac{1}{\tau(n)}$, which means

$$\begin{aligned}&\mathbb{E}_k[Z(k)]\\ =&1-\mathbb{E}_k[1-Z(k)]\\ \leq&1-\frac{1}{\tau(n)}\cdot\tau(n)d\epsilon\\ =&1-d\epsilon.\end{aligned}$$

This leads to a contradiction. Thus, as long as Equation (4) holds, the proportion of the keys satisfying $Z(k) > 1 - \tau(n)d\epsilon$ in $\mathbb{F}_2^m$ must be greater than $(1 - \frac{1}{\tau(n)})$. Let S be a set of all such keys. We have $|S|/|\mathbb{F}_2^m| > 1 - \frac{1}{\tau(n)}$, and for every $k \in S$,

$$Z(k) = \frac{|\{x \in \mathbb{F}_2^n | Enc_k^t(x \oplus a) + Enc_k^t(x) \sim b\}|}{2^n} > 1 - \tau(n)d\epsilon.$$

Let $\epsilon = \frac{1-\sigma}{\tau(n)d}$. Since $q(n) = \frac{1}{2(1-\sigma)^2}\tau(n)^2 n^3$, the probability that Equation (4) holds is larger than $1 - ne^{-n}$. Therefore, with an overwhelming probability, there is a subset $S \subseteq \mathbb{F}_2^m$ satisfying that $|S|/|\mathbb{F}_2^m| > 1 - \frac{1}{\tau(n)}$, and for every $k \in S$,

$$\frac{|\{x \in \mathbb{F}_2^n | Enc_k^t(x \oplus a) + Enc_k^t(x) \sim b\}|}{2^n} > 1 - \tau(n)d\epsilon = \sigma,$$

which means that the differential probability of (a,b) is greater than σ for more than $(1 - \frac{1}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$ $\square$

When implementing a truncated differential attack, the adversaries first choose a polynomial $\tau(n)$ and a parameter σ, then run Algorithm 2 to get (a,b). The polynomial $\tau(n)$ is used to characterize the expected proportion of keys under which (a,b) has high probability. The parameter σ is used characterize the expected differential probability. According to Theorem 3, with an overwhelming probability, for at least $(1-\frac{1}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$ the probability of (a,b) is greater than σ. Then the adversaries can use (a,b) to determine the subkey of the last $r-t$ rounds as in a traditional truncated differential attack. This attack works for at least the $(1-\frac{1}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$. The amount of plaintext pairs required in the counting process is determined by the value of S/N. Based on experimental observations, about 20 to 40 appearances of right plaintext pairs are enough [49]. Therefore, about $\frac{40}{\sigma}$ plaintext pairs are sufficient.

For analyzing the complexity, we first calculate the amounts of universal gates and qubits required and then estimate the complexity of the classical computing involved.

In Algorithm 2, BV algorithm is performed on each $Enc^t[j]$ for $q(n)$ times ($j \in \{1,2,\cdots,n\}$). Every call requires the execution of $2(m+n)+1$ Hadamard gates and one quantum circuit of $Enc^t[j]$. Thus, each call requires $2(m+n)+1+|Enc^t[j]|_Q$ quantum universal gates. The total number of Hadamard gates required for Algorithm 2 is

$$\begin{aligned}&q(n)\sum_{j=1}^{n}\big[\,2(m+n)+1\big]\\=&q(n)\big[\,(2m+1)n+2n^2\big]\\=&\frac{1}{2(1-\sigma)^2}\tau(n)^2n^4(2n+2m+1).\end{aligned}$$

Since it holds that

$$\begin{aligned}&q(n)\sum_{j=1}^{n}|Enc^t[j]|_Q\\=&q(n)|Enc^t|_Q\\=&\frac{1}{2(1-\sigma)^2}\tau(n)^2n^3|Enc^t|_Q,\end{aligned}$$

the total number of times Algorithm 2 needs to execute the quantum circuit of Enc^t is $\frac{1}{2(1-\sigma)^2}\tau(n)^2n^3$. In summary, Algorithm 2 requires

$$\frac{1}{2(1-\sigma)^2}\tau(n)^2n^3[2n^2+(2m+1)n+|Enc^t|_Q] \tag{5}$$

universal gates in total. This number is a polynomial of n and m.

Classical computing part is to solve the linear system $\{x\cdot u=i_j|u\in W\}$ for each $j=1,2,\cdots,n$ and $i_j=0,1$. The adversaries need to solve a total of $2n$ systems, and every system has $q(n)$ equations and n unknowns. Therefore, the classical complexity of this part is $O(2q(n)n^3)=O(\frac{1}{(1-\sigma)^2}\tau(n)^2n^6)$. Applying BV algorithm to every $Enc^t[j]$ requires $m+n+1$ qubits. Thus, Algorithm 2 requires

$$q(n)(n+m+1)=\frac{1}{2(1-\sigma)^2}\tau(n)^2n^3(n+m+1) \tag{6}$$

qubits in total.

The parameters involved in Algorithm 2 include the constant σ, polynomial $\tau(n)$, blocksize n and key length m. For the convenience of parameter analysis, we list the quantum resources required for Algorithm 2 in Table 2, then analyze the influence of these parameters on the complexity of Algorithm 2.

Table 2. Quantum resources required for Algorithm 2.

Hadamard Gate	Quantum Execution of Enc^t	Qubit
$\frac{1}{2(1-\sigma)^2}\tau(n)^2n^4(2m+2n+1)$	$\frac{1}{2(1-\sigma)^2}\tau(n)^2n^3$	$\frac{1}{2(1-\sigma)^2}\tau(n)^2n^3(m+n+1)$

The parameter σ is chosen by the adversary and satisfies $0 < \sigma \leq 1$. σ is the lower bound of the probability of truncated differentials desired by the adversary. Since truncated differentials have at least one predicted bit, the probability of any truncated differential of a random permutation is no more than $\frac{1}{2}$. Taking $\sigma = \frac{1}{2}$ is sufficient to ensure that the truncated differential output by Algorithm 2 is an effective differential. When more than one bit is predicted, the value of σ can take a smaller value. Therefore, the coefficient $\frac{1}{2(1-\sigma)^2}$ in Table 2 usually can be seen as a small constant.

The parameter $\tau(n)$ is a polynomial chosen by the adversary. It characterizes the expected proportion of keys under which the output differential has high probability. The larger the value of $\tau(n)$, the more keys are feasible for the attack, but at the same time, the complexity also increases. The adversary can choose $\tau(n)$ based on the expected key proportion and acceptable complexity. Especially, $\tau(n)$ can be chosen as a constant t_0, then the number of Hadamard gates is $O(n^5)$. The number of times Enc^t needs to be executed quantumly is $O(n^3)$ and the number of qubits is $O(n^4)$. Here we omit m because usually $m = O(n)$.

The values of parameters n, m depend on which block cipher is attacked. For common non-lightweight block ciphers, the value of the blocksize n is generally between 128 and 256, the value of the key length m is generally between 128 and 256. For common lightweight block ciphers, the value of the blocksize n is generally between 32 and 128, the value of the key length m is generally between 64 and 256. We take $\sigma = \frac{1}{2}$, $\tau(n) = 2$ as an example and list the values of these parameters of several block ciphers and the corresponding complexity of Algorithm 2 in Table 3.

Table 3. Quantum complexity of Algorithm 2 on specific block ciphers [1].

Block Cipher	n	m	Hadamard Gate	Quantum Execution of Enc^t	Qubit
LBlock	64	80	$2^{35.2}$	$2^{21.0}$	$2^{28.2}$
PRESENT-80	64	80	$2^{35.2}$	$2^{21.0}$	$2^{28.2}$
SPECK32/64	32	64	$2^{30.6}$	$2^{18.0}$	$2^{24.6}$
Simon-32/64	32	64	$2^{30.6}$	$2^{18.0}$	$2^{24.6}$

[1] Complexity is calculated by taking $\sigma = \frac{1}{2}$ and $\tau(n) = 2$.

At present, the largest quantum chip is released by IBM, supporting over 1000-plus qubits [60]. IBM quantum platform supports the quantum circuits of 100-plus qubits. According to Table 3, it is unfeasible to completely implement or simulate Algorithm 2 on a block cipher.

3.3. Simulation

In this subsection, we simulate Algorithm 2 acting on a simple Boolean function. This demonstrates the practicality and correctness of Algorithm 2. Specifically, we choose a Boolean function $F : \mathbb{F}_2^4 \to \mathbb{F}_2^4$, whose truth table in presented in Table 4. Let $F = (F_1, F_2, F_3, F_4)$. To simulate Algorithm 2 with Qiskit, we need to construct the quantum circuit of each component function F_i ($i = 1, 2, 3, 4$), then apply BV algorithm on each F_i to find high-probability differentials of F_i. Using LIGHTER-R tool or manual deduction it is easy to obtain the construction of quantum circuits of all component functions $F_i's$. The code of the simulation is presented on GitHub [61].

After constructing the quantum circuit of F_1 on Qiskit, we use the draw method to generate the quantum circuit diagram of BV algorithm acted on F_1. The circuit diagram is shown in Figure 6. The symbol M denotes the measurement on the computational basis states. We add a dotted box to mark the part of quantum circuit implementing F_1.

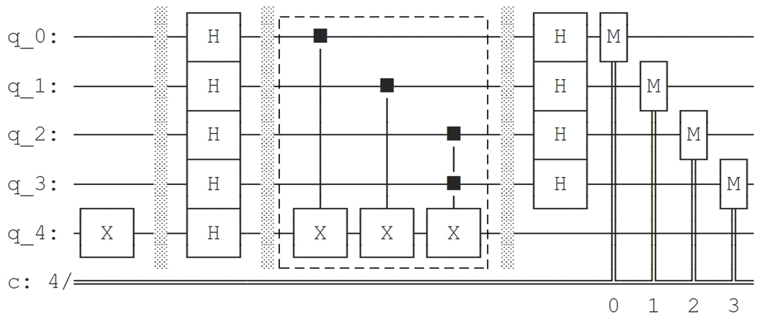

Figure 6. Quantum circuit diagram of BV algorithm acted on F_1 generated by Qiskit.

Table 4. Truth table of F.

x_1	x_2	x_3	x_4	F	x_1	x_2	x_3	x_4	F
0	0	0	0	0000	1	0	0	0	1001
0	0	0	1	0010	1	0	0	1	1111
0	0	1	0	0110	1	0	1	0	1111
0	0	1	1	1101	1	0	1	1	0000
0	1	0	0	1111	1	1	0	0	0111
0	1	0	1	1101	1	1	0	1	0001
0	1	1	0	1000	1	1	1	0	0000
0	1	1	1	0011	1	1	1	1	1111

The measurement results simulated by Qiskit are shown in Figure 7. They only take four values: 1100, 1110, 1101 and 1111. Then solving the equation

$$\begin{cases} (1100)\cdot x = 0 \\ (1110)\cdot x = 0 \\ (1101)\cdot x = 0 \\ (1111)\cdot x = 0 \end{cases}$$

gives a fundamental solution system: $\{(1100)\}$. The solution set of the above equation is $Z_1^0 = \{(1100),(0000)\}$. The solution set of the equation

$$\begin{cases} (1100)\cdot x = 1 \\ (1110)\cdot x = 1 \\ (1101)\cdot x = 1 \\ (1111)\cdot x = 1 \end{cases}$$

is $Z_1^1 = \{(1000),(0100)\}$. According to step 9 of Algorithm 2, we let $Z_1 = Z_1^0 \cup Z_1^1 = \{(1100),(0000),(1000),(0100)\}$.

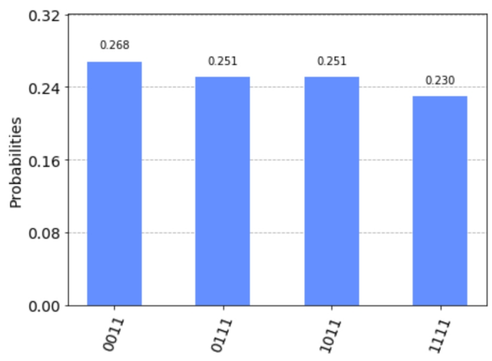

Figure 7. Measurement results on F_1 simulated by Qiskit.

By employing a similar method, we construct the quantum circuit of F_2 on Qiskit and use the draw method to generate the quantum circuit diagram of BV algorithm acted on F_2. The circuit diagram is shown in Figure 8. We add a dotted box to mark the part of quantum circuit implementing F_2.

The measurement results simulated by Qiskit are shown in Figure 9. They only take four values: 0110, 1110, 0111 and 1111. Solving the equation

$$\begin{cases} (0110) \cdot x = 0 \\ (1110) \cdot x = 0 \\ (0111) \cdot x = 0 \\ (1111) \cdot x = 0 \end{cases}$$

gives a fundamental solution system: $\{(0110)\}$. The solution set of the above equation is $Z_2^0 = \{(0110), (0000)\}$. The solution set of the equation

$$\begin{cases} (0110) \cdot x = 1 \\ (1110) \cdot x = 1 \\ (0111) \cdot x = 1 \\ (1111) \cdot x = 1 \end{cases}$$

is $Z_2^1 = \{(0100), (0010)\}$. According to step 9 of Algorithm 2, we let $Z_2 = Z_2^0 \cup Z_2^1 = \{(0110), (0000), (0100), (0010)\}$.

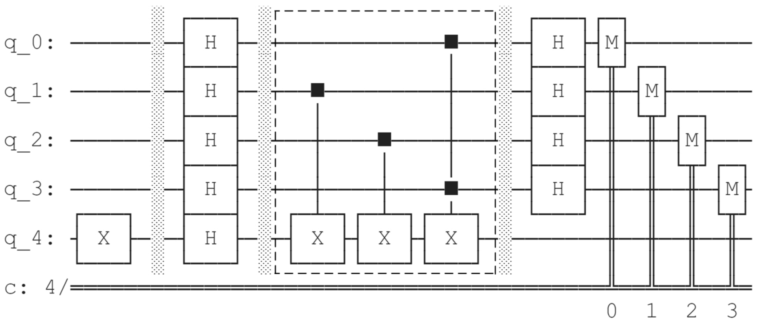

Figure 8. Quantum circuit diagram of BV algorithm acted on F_2 generated by Qiskit.

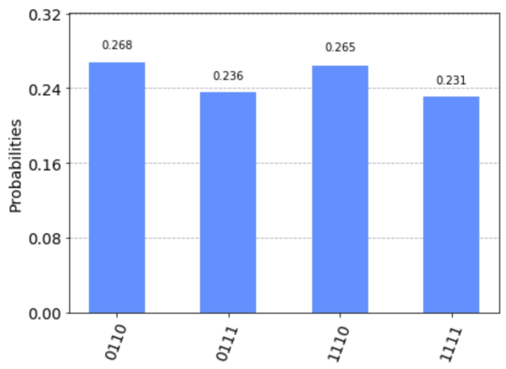

Figure 9. Measurement results on F_2 simulated by Qiskit.

Similarly, we construct the quantum circuit of F_3 on Qiskit and use the draw method to generate the quantum circuit diagram of BV algorithm acted on F_3. The circuit diagram is shown in Figure 10. We add a dotted box to mark the part of quantum circuit implementing F_3.

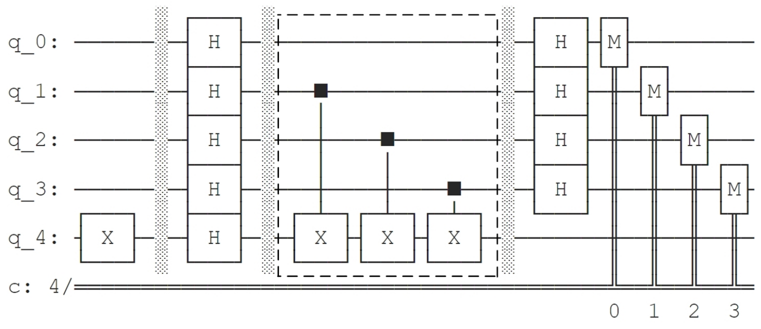

Figure 10. Quantum circuit diagram of BV algorithm acted on F_3 generated by Qiskit.

The measurement results simulated by Qiskit are shown in Figure 11. They only take one value: 0111. Solving the equation $(0111) \cdot x = 0$ gives a fundamental solution system: $\{(1000), (0101), (0011)\}$. The solution set of this equation is $Z_3^0 = \{(0000), (1000), (0101),$ $(0011), (1101), (0110), (1011), (1110)\}$. The solution set of the equation $(0111) \cdot x = 1$ is $Z_3^1 = \{(0001), (1001), (0100), (0010), (1100), (0111), (1010), (1111)\}$. According to step 9 of Algorithm 2, we let

$$Z_3 = Z_3^0 \cup Z_3^1 = \{(0000), (1000), (0101), (0011), (1101), (0110), (1011), (1110), (0001), \\ (1001), (0100), (0010), (1100), (0111), (1010), (1111)\}.$$

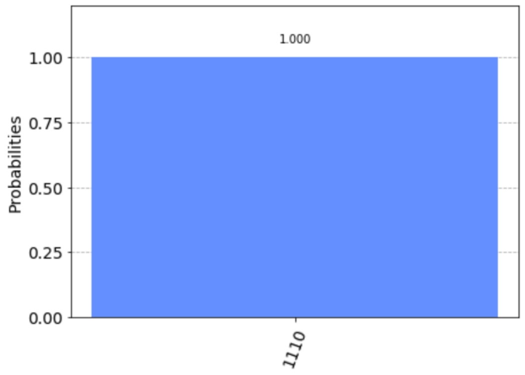

Figure 11. Measurement results on F_3 simulated by Qiskit.

Then we construct the quantum circuit of F_4 on Qiskit and use the draw method to generate the quantum circuit diagram of BV algorithm acted on F_4. The circuit diagram is shown in Figure 12. We add a dotted box to mark the part of quantum circuit implementing F_4.

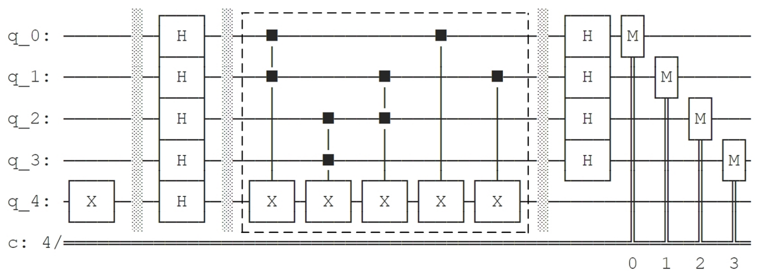

Figure 12. Quantum circuit diagram of BV algorithm acted on F_4 generated by Qiskit.

The measurement results simulated by Qiskit are shown in Figure 13. All vectors in $\mathbb{F}_2^4$ appear in the measurement results. The system of linear equations $\{u \cdot x = 0 | u \in \mathbb{F}_2^4\}$ has only one solution $Z_4^0 = \{(0000)\}$. The solution set of the system of linear equations $\{u \cdot x = 1 | u \in \mathbb{F}_2^4\}$ is the empty set, that is, $Z_4^1 = \Phi$. According to step 9 of Algorithm 2, we let $Z_4 = Z_4^0 \cup Z_4^1 = \{(0000)\}$

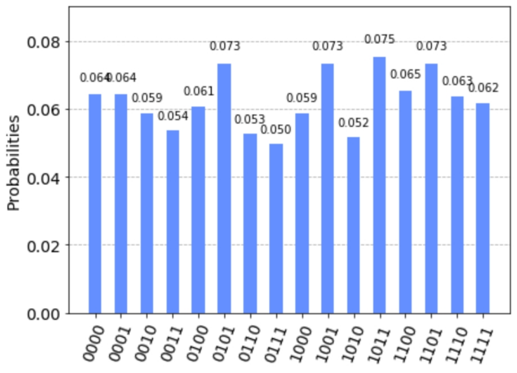

Figure 13. Measurement results on F_4 simulated by Qiskit.

Since $Z_1 \cap Z_2 \cap Z_3 \cap Z_4 = \{(0000)\}$, $Z_1 \cap Z_2 \cap Z_3 = \{(0000), (0100)\}$ and $(0100) \in Z_1^1 \cap Z_2^1 \cap Z_3^1$, Algorithm 2 chooses $a = (0100)$ and let $b = (111*)$, then output (a, b). It is easy to verify that $F(x \oplus (0100)) \oplus F(x) \sim b$ holds for all $x \in \mathbb{F}_2^4$. The probability of the truncated differential (a, b) is one. This indicates that Algorithm 2 can indeed find high-probability truncated differentials.

4. Quantum Boomerang Attack

4.1. Quantum Algorithm for Finding Boomerang Distinguisher

Since its proposal in 1999, the boomerang attack [52] has been widely used as a cryptanalysis method. The principle of boomerang cryptanalysis is to connect two differential paths having a high probability such that the adversaries can attack more rounds. This attack was proposed because, when constructing the differential characteristics of block ciphers, the probability of the differential rapidly decreases as the round number increases. It works in cases where it is difficult to find a $(t_1 + t_2)$-round differential characteristic of some block ciphers that has high probability, while it is possible to find t_1-round and t_2-round differential characteristics having high probability.

Suppose $Enc_k^r(x) = Enc_k^{t_2} \circ Enc_k^{t_1}(x)$, where $t_1 + t_2 = r$, $Enc_k^{t_1}$ has a p_1-probability truncated differential $(\overline{\Delta}x, \overline{\Delta}y)$ and the inverse function $Enc_k^{t_2\,-1}$ of $Enc_k^{t_2}$ has a p_2-probability truncated differential $(\overline{\nabla}x, \overline{\nabla}y)$. As shown in Figure 14, P, P', Q, Q' are four plaintexts and the corresponding ciphertexts under Enc_k^r are C, C', D, D', respectively. P, P' are said to satisfy the differential $(\overline{\Delta}x, \overline{\Delta}y)$ of $Enc_k^{t_1}$, if $P \oplus P' \sim \overline{\Delta}x$ and $Enc_k^{t_1}(P) \oplus Enc_k^{t_1}(P') \sim \overline{\Delta}y$. If both P, P' and Q, Q' satisfy the differential $(\overline{\Delta}x, \overline{\Delta}y)$ of $Enc_k^{t_1}$, and both C, D and C', D' satisfy the differential $(\overline{\nabla}x, \overline{\nabla}y)$ of $Enc_k^{t_2^{-1}}$, then (P, P', Q, Q') is called a right quadruple. Such two differentials are called a boomerang distinguisher of Enc_k^r. Quadruples can be generated via the following method:

1. Choose two plaintexts (P, P') satisfying $P \oplus P \sim \overline{\Delta}x$ and denote the corresponding ciphertexts as (C, C').
2. Compute $D = C \oplus \overline{\nabla}y$ and $D' = C' \oplus \overline{\nabla}y$, and decrypt D, D' to obtain the corresponding plaintexts Q, Q'.
3. Test whether it holds that $Q \oplus Q' \sim \overline{\Delta}x$.

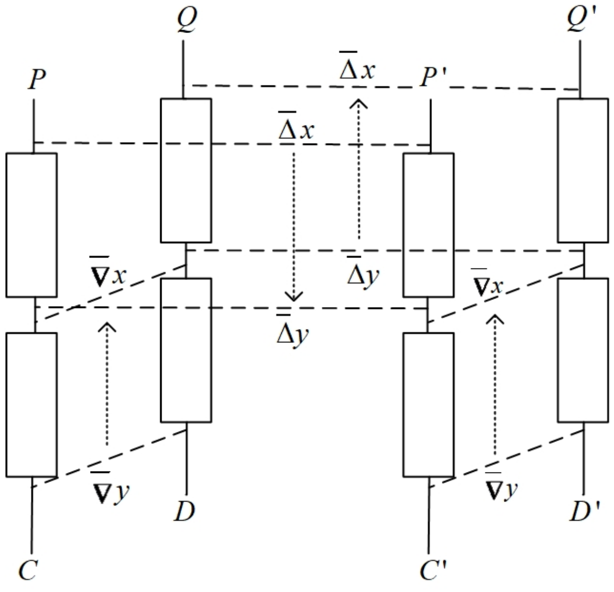

Figure 14. Boomerang attack.

The probability that (P, P') satisfies differential $(\overline{\Delta}x, \overline{\Delta}y)$ is p_1. The probability of (C, D) satisfying the differential $(\overline{\nabla}x, \overline{\nabla}y)$ is p_2. The probability of (C', D') satisfying the differential $(\overline{\nabla}x, \overline{\nabla}y)$ is also p_2. Under these three conditions, it naturally holds that $Enc_k^{t_1}(Q) \oplus Enc_k^{t_1}(Q') \sim \overline{\Delta}y$, so that the probability of $Q \oplus Q' \sim \overline{\Delta}x$ is p_1. In summary, the probability of a quadruple generated by the above method being a right quadruple is $(p_1p_2)^2$. For a random permutation, this probability is 2^{-d}, where d is the number of determined bits of $\overline{\Delta}x$. If $(p_1p_2)^2 > 2^{-d}$, then the block cipher can be distinguished from a random permutation through data analysis. A boomerang distinguisher can be used to search for the subkey involved in the last several rounds of the attacked block cipher.

The key to a boomerang attack is to find the boomerang distinguisher, namely, a t_1-round truncated differential of $Enc_k^{t_1}$ and a t_2-round truncated differential of $Enc_k^{t_2^{-1}}$ that have a high probability. Thus, the essence of boomerang attack is to find two truncated differentials that have a high probability, which can be achieved using Algorithm 2. According to these analysis, we propose Algorithm 3 for finding boomerang distinguishers.

Steps 4–22 of Algorithm 3 are used to find a truncated differential (a, b) of Enc^{t_1}. The probability of (a, b) is larger than σ for at least $1 - 1/\tau(n)$ proportion of keys in $\mathbb{F}_2^m$. Steps 26–43 are used to find a truncated differential (α, β) of $Enc^{t_2^{-1}}$. The probability of (α, β) is also larger than σ for at least $1 - 1/\tau(n)$ proportion of keys in $\mathbb{F}_2^m$. Steps 23–25 are to determine whether the truncated differential of Enc^{t_1} has been found. If $\eta = 0$, no satisfied truncated differential of Enc^{t_1} is found, then break out of the current loop and try the next t_1. If (a, b) and (α, β) are successfully found, these two differentials form a boomerang distinguisher $\{(a,b),(\alpha,\beta)\}$ of Enc_k^r. The probability of the corresponding quadruple is σ^4 for more than $1-2/\tau(n)$ proportion of keys in $\mathbb{F}_2^m$. The flowchart of Algorithm 3 is presented in Figure 15.

Algorithm 3 Quantum algorithm for finding boomerang distinguishers.

```
Input: The quantum circuit of $Enc^r$, a polynomial $\tau(n)$ and a constant $\sigma$ ($0 < \sigma < 1$)
       chosen by the adversaries.
Output: a boomerang distinguisher of $Enc^r$.
 1: Let $q(n) = \frac{1}{2(1-\sigma)^2}\tau(n)^2 n^3$;
 2: Define a set $W := \Phi$;
 3: for $t_1 = 1,2,\cdots,r-1$ do
 4:     for $j = 1,2,\cdots,n$ do
 5:         for $l = 1,\cdots,q(n)$ do
 6:             Apply BV algorithm to $Enc^{t_1}[j]$ to get an output $u = (u_1,\cdots,u_n,\cdots,u_{n+m})$;
 7:             Let $W = W \cup \{(u_1,\cdots,u_n)\}$;
 8:         end for
 9:         Solve the linear equation $\{u \cdot x = i_j | u \in W\}$ to get solution sets $Z_j^{i_j}$ for $i_j = 0,1$;
10:         Compute the set $Z_j = Z_j^0 \cup Z_j^1$;
11:         Compute the set $\overline{Z}_j = \{(a,i_j) | a \in Z_j^{i_j}, i_j = 0,1\}$;
12:         Let $W = \Phi$;
13:     end for
14:     Let $\eta = 0$;
15:     for $d = n, n-1\cdots,1$ do
16:         if $S/N = 2^d\sigma > 1$ then
17:             if there are $d$ different subscripts $j_1,\cdots,j_d$ s.t. $Z_{j_1} \cap \cdots \cap Z_{j_d} \supsetneq \{\vec{0}\}$ then
18:                 Choose at random a vector $a \in Z_{j_1} \cap \cdots \cap Z_{j_d}$, and for $j = 1,\cdots,n$, let
                        $$b_j = \begin{cases} i_j, & j \in \{j_1,\cdots,j_d\} \\ *, & j \notin \{j_1,\cdots,j_d\}, \end{cases}$$
                    where $i_j$ denotes the bit appended to $a$ in the set $\overline{Z}_j$, i.e., $(a,i_j) \in \overline{Z}_j$;
19:                 Let $b = (b_1,\cdots,b_n)$. Let $\eta = 1$ and break out of the current loop;
20:             end if
21:         end if
22:     end for
23:     if $\eta = 0$ then
24:         Break out of the current loop;
25:     end if
26:     for $j = 1,2,\cdots,n$ do
27:         for $l = 1,\cdots,q(n)$ do
28:             Apply BV algorithm to $Enc^{t_2}{}^{-1}[j]$ to get output $u = (u_1,\cdots,u_n,\cdots,u_{n+m})$;
29:             Let $W = W \cup \{(u_1,\cdots,u_n)\}$;
30:         end for
31:         Solve the linear equation $\{u \cdot x = i_j | u \in W\}$, to get solution sets $V_j^{i_j}$ for
            $i_j = 0,1$, respectively;
32:         Compute the set $V_j = V_j^0 \cup V_j^1$;
33:         Compute the set $\overline{V}_j = \{(a,i_j) | a \in V_j^{i_j}, i_j = 0,1\}$;
34:         Let $W = \Phi$;
35:     end for
```

Algorithm 3 *Cont.*

```
36:     for $d = n, n-1 \cdots, 1$ do
37:         if $S/N = 2^d\sigma > 1$ then
38:             if there are $d$ different subscripts $j_1, \cdots, j_d$ s.t. $V_{j_1} \cap \cdots \cap V_{j_d} \supsetneq \{\vec{0}\}$ then
39:                 Choose at random a vector $\alpha \in V_{j_1} \cap \cdots \cap V_{j_d}$, and for $j = 1, \cdots, n$, let
                    $$\beta_j = \begin{cases} i_j, & j \in \{j_1, \cdots, j_d\} \\ *, & j \notin \{j_1, \cdots, j_d\}, \end{cases}$$
                    where $i_j$ denotes the bit appended to $\alpha$ in the set $\overline{V}_j$, i.e., $(\alpha, i_j) \in \overline{V}_j$;
40:                 Let $\beta = (\beta_1, \cdots, \beta_n)$. Output $[(a,b), (\alpha, \beta), t_1]$ and stop;
41:             end if
42:         end if
43:     end for
44: end for
45: Output "No" and stop;
```

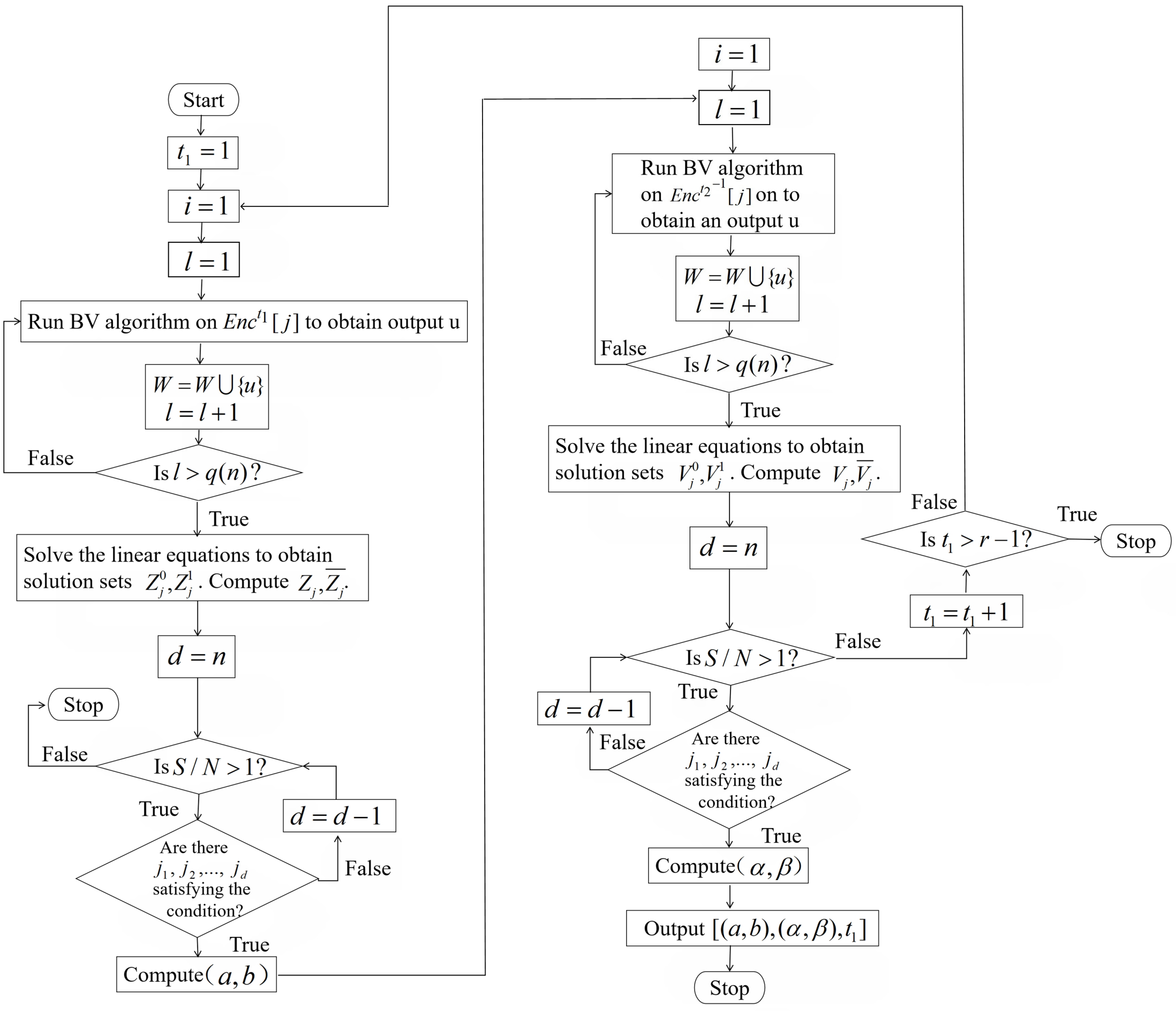
Start
$t_1 = 1$
$i = 1$
$l = 1$
Run BV algorithm on $Enc^{t_1}[j]$ to obtain output u
$W = W \bigcup \{u\}$
$l = l + 1$
Is $l > q(n)$?
False
True
Solve the linear equations to obtain solution sets Z_j^0, Z_j^1. Compute $Z_j, \overline{Z}_j$.
$d = n$
Stop
False
Is $S/N > 1$?
True
$d = d - 1$
Are there $j_1, j_2, \ldots, j_d$ satisfying the condition?
False
True
Compute(a, b)
$i = 1$
$l = 1$
Run BV algorithm on $Enc^{t_2^{-1}}[j]$ on to obtain an output u
$W = W \bigcup \{u\}$
$l = l + 1$
False
Is $l > q(n)$?
True
Solve the linear equations to obtain solution sets V_j^0, V_j^1. Compute $V_j, \overline{V}_j$.
$d = n$
False
Is $t_1 > r - 1$?
True
Stop
$t_1 = t_1 + 1$
False
Is $S/N > 1$?
True
$d = d - 1$
False
Are there $j_1, j_2, \ldots, j_d$ satisfying the condition?
True
Compute(α, β)
Output $[(a,b),(\alpha,\beta),t_1]$
Stop

Figure 15. The flowchart of Algorithm 3.

4.2. Analysis of Algorithm 3

The process of Algorithm 3 is actually to call Algorithm 2 on Enc^{t_1} and $Enc^{t_2\,-1}$, respectively, for all $t_1 = 1, 2, \cdots, n$ and $t_2 = r - t_1$. Therefore, the total number of Hadamard gates required for Algorithm 3 is

$$\begin{aligned}&\sum_{t_1=1}^{r-1} \frac{1}{2(1-\sigma)^2}\tau(n)^2 n^3 \left[4n^2 + (4m+2)n\right]\\ =&\frac{r-1}{(1-\sigma)^2}\tau(n)^2 n^4 (2n+2m+1).\end{aligned}$$

Since it holds that

$$\begin{aligned}&\sum_{t_1=1}^{r-1} \frac{1}{2(1-\sigma)^2}\tau(n)^2 n^3 (|Enc^{t_1}|_Q + |Enc^{t_2\,-1}|_Q)\\ =&\sum_{t_1=1}^{r-1} \frac{1}{2(1-\sigma)^2}\tau(n)^2 n^3 (|Enc^{t_1}|_Q + |Enc^{t_2}|_Q)\\ =&\sum_{t_1=1}^{r-1} \frac{1}{2(1-\sigma)^2}\tau(n)^2 n^3 |Enc^{r}|_Q\\ =&\frac{r-1}{2(1-\sigma)^2}\tau(n)^2 n^3 |Enc^{r}|_Q,\end{aligned}$$

the total number of times Algorithm 3 needs to execute the quantum circuit of Enc^r is $\frac{r-1}{2(1-\sigma)^2}\tau(n)^2 n^3$. This number is a polynomial of n and m. In summary, Algorithm 3 requires

$$\frac{r-1}{2(1-\sigma)^2}\tau(n)^2 n^3 (4n^2 + 4mn + 2n + |Enc^r|_Q) \tag{7}$$

universal gates in total.

Algorithm 3 calls Algorithm 2 on Enc^{t_1} and $Enc^{t_2\,-1}$, respectively, for all $t_1 = 1, 2, \cdots, r-1$ and $t_2 = r - t_1$. Thus, Algorithm 3 requires

$$\frac{1}{2(1-\sigma)^2}\tau(n)^2 n^3 (m+n+1) \times 2 \times (r-1) = \frac{r-1}{(1-\sigma)^2}\tau(n)^2 n^3 (m+n+1) \tag{8}$$

qubits in total.

The parameters involved in Algorithm 3 include the constant σ, polynomial $\tau(n)$, blocksize n, number of rounds r and key length m. For the convenience of parameter analysis, we list the numbers of quantum resources required for Algorithm 3 in Table 5, then analyze the influence of parameters on the complexity of Algorithm 3.

Table 5. Quantum resources required for Algorithm 3.

Hadamard Gate	**Quantum Execution of *Enc^r***	**Qubit**
$\frac{r-1}{(1-\sigma)^2}\tau(n)^2 n^4 (2m+2n+1)$	$\frac{r-1}{2(1-\sigma)^2}\tau(n)^2 n^3$	$\frac{r-1}{(1-\sigma)^2}\tau(n)^2 n^3 (m+n+1)$

Similar to Algorithm 2, the parameters σ of Algorithm 3 are chosen by the adversary. σ^4 is the lower bound of the probability of boomerang distinguishers desired by the adversary. Since the probability of any boomerang distinguisher of a random permutation is no more than $\frac{1}{2^4}$, taking $\sigma = \frac{1}{2}$ is sufficient to ensure that the boomerang distinguisher output by Algorithm 3 is an effective distinguisher.

The parameter $\tau(n)$ is a polynomial chosen by the adversary. It characterizes the expected proportion of keys under which the output boomerang distinguisher has high probability. Specifically, $\tau(n)$ can be chosen as a constant t_0, then the number of Hadamard gates is $O(rn^5)$, the number of times Enc^r needs to be executed quantumly is $O(rn^3)$ and the number of qubits is $O(rn^4)$. Here, we omit m because usually $m = O(n)$.

The values of parameters n, r, m depend on which block cipher is attacked. For common non-lightweight block ciphers, the value of the blocksize n is generally between 128 and 256, the value of the round number r is generally between 10 and 40 and the value of the key length m is generally between 128 and 256. For common lightweight block ciphers, the value of the blocksize n is generally between 32 and 128, the value of the round number r is generally between 32 and 80 and the value of the key length m is generally between 64 and 256. We take $\sigma = \frac{1}{2}$, $\tau(n) = 2$ as an example and list the values of these parameters of several common block ciphers and the corresponding complexity of Algorithm 3 in Table 6.

Table 6. Quantum complexity of Algorithm 3 on specific block ciphers [1].

Block Cipher	n	r	m	Hadamard Gate	Quantum Execution of Enc^r	Qubit
LBlock	64	32	80	$2^{41.1}$	$2^{26.0}$	$2^{34.1}$
PRESENT-80	64	31	80	$2^{41.1}$	$2^{25.9}$	$2^{34.1}$
SPECK32/64	32	22	64	$2^{36.0}$	$2^{22.4}$	$2^{30.0}$
Simon-32/64	32	32	64	$2^{36.5}$	$2^{23.0}$	$2^{30.6}$

[1] Complexity is calculated by taking $\sigma = \frac{1}{2}$ and $\tau(n) = 2$.

Since IBM quantum platform only supports operations of 100-plus qubits, according to Table 6, it is unfeasible to completely implement or simulate Algorithm 3 on a block cipher.

5. Results

We apply BV algorithm to truncated differential cryptanalysis and boomerang cryptanalysis and propose two quantum algorithms for finding high-probability truncated differentials and boomerang distinguishers, respectively.

For truncated differential cryptanalysis, we propose Algorithm 2 for finding truncated differentials that have high probability. Given the quantum circuit of a block cipher Enc^t, Algorithm 2 takes the key as a part of the input and repeats running BV algorithm on each component function of Enc^t to find truncated differentials of each $Enc^t[j]$, then obtains a truncated differential of Enc^t by searching for a common input difference of as many component functions as possible. When executing Algorithm 2, the adversary first chooses parameters σ and $\tau(n)$, Algorithm 2 is then run to obtain a truncated differential. We use quantum information theory and probability theory to rigorously prove that the probability of the truncated differential output by Algorithm 2 must be greater than σ for more than $(1 - \frac{1}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$. Algorithm 2 can be run by Q1 quantum adversaries and the complexity is at polynomial level. We take $\sigma = \frac{1}{2}$ as an example and list numbers of universal gates and qubits of Algorithm 2 under different values of $\tau(n)$ in Table 7. The values in Table 7 are obtained according to Equations (5) and (6).

Table 7. Quantum complexity of Algorithm 2 under $\sigma = \frac{1}{2}$.

$\tau(n)$	Universal Gate	Qubit
2	$8n^3(2n^2 + 2mn + n + \lvert Enc^t\rvert_Q)$	$8n^3(n + m + 1)$
$n/2$	$\frac{1}{2}n^5(2n^2 + 2mn + n + \lvert Enc^t\rvert_Q)$	$\frac{1}{2}n^5(n + m + 1)$
n	$2n^5(2n^2 + 2mn + n + \lvert Enc^t\rvert_Q)$	$2n^5(n + m + 1)$

For boomerang cryptanalysis, we propose Algorithm 3 for finding boomerang distinguishers. Given the quantum circuit of a block cipher Enc^r, Algorithm 3 traverses the value of t_1 from 1 to $r - 1$ and calls Algorithm 2 to find the truncated differentials of Enc^{t_1}

and Enc^{t_2}, respectively, where $t_2 = r - t_1$. When executing Algorithm 3, the adversary also needs to choose parameters σ and $\tau(n)$, then runs Algorithm 3 to obtain a boomerang distinguisher of Enc^r. The probability of generating a right quadruple of this boomerang distinguisher is greater than σ^4 for more than $(1 - \frac{2}{\tau(n)})$ proportion of keys in $\mathbb{F}_2^m$. Algorithm 3 can be run by Q1 quantum adversaries and the complexity is at polynomial level. We take $\sigma = \frac{1}{2}$ as an example, and list number of universal gates and qubits of Algorithm 3 under different values of $\tau(n)$ in Table 8. The values in Table 8 are obtained according to Equations (7) and (8).

Table 8. Quantum complexity of Algorithm 3 under $\sigma = \frac{1}{2}$.

$\tau(n)$	Universal Gate	Qubit
2	$8(r-1)n^3(4n^2+4mn+2n+\lvert Enc^r\rvert_Q)$	$16(r-1)n^3(n+m+1)$
$n/2$	$\frac{1}{2}(r-1)n^5(4n^2+4mn+2n+\lvert Enc^r\rvert_Q)$	$(r-1)n^5(n+m+1)$
n	$2(r-1)n^5(4n^2+4mn+2n+\lvert Enc^r\rvert_Q)$	$4(r-1)n^5(n+m+1)$

Both Algorithm 2 and Algorithm 3 can be executed in Q_1 model. As shown in Tables 7 and 8, the quantum complexity of both algorithms are at the polynomial level. They show the superiority of quantum computing in cryptanalysis.

6. Conclusions

In this study, we further explored the superior computing power of quantum algorithms when applied to the field of cryptanalysis. We used BV algorithm to enhance two variants of differential cryptanalysis: truncated differential cryptanalysis and boomerang cryptanalysis. We constructed two quantum algorithms that can find truncated differentials and boomerang distinguishers of block ciphers. We prove with an overwhelming probability, that the truncated differentials or boomerang distinguishers found by our algorithms have a high probability for the most keys in the key space.

The complexity of our algorithms is at the polynomial level and adversaries can realize them in Q1 model. Compared to many proposed quantum attack algorithms [14–16,18,19,46] which demand quantum queries, our algorithms are more practical for realization. Classical automatic tools for searching truncated differentials with high probability or boomerang distinguishers were unable to consider all the details of S-boxes when the S-boxes were not small-scale. For example, in the case of the widely used 8-bit S-boxes, the classical searching tools can only work for extremely few rounds. In comparison, our algorithms fully utilize the strengths of quantum computing to compensate for this shortcoming. Their quantum circuits strictly compute the S-boxes when performing the operator U_{Enc^t} and only have polynomial quantum gates. Moreover, classical truncated differential and boomerang attacks are unable to consider the influence of key scheduling in the attack model of single-key, but the proposed algorithms incorporate the key scheduling into the operator U_{Enc^t} and thus fully consider the impact of the key scheduling. We believe the study of quantum cryptanalysis is crucial for the design of quantum-secure cryptosystems in order to prepare for the arrival of quantum computers.

For further research, reducing the quantum complexity of the proposed algorithms is a meaningful direction. It would also be interesting to explore the possible applications of quantum algorithms in other cryptanalytic tools such as integral and algebraic attacks. Quantum key distribution technique uses quantum systems to generate and distribute keys. The quantum algorithms proposed in this paper are used to attack traditional block ciphers that encrypt classical information. Investigating a combination of the proposed algorithms with quantum key distribution technique may be an interesting research direction.

Author Contributions: Conceptualization, H.X. and L.Y.; Formal analysis, H.X.; Funding acquisition, H.X.; Investigation, H.X.; Methodology, H.X.; Validation, H.X. and L.Y.; Visualization, H.X.; Writing—original draft, H.X.; Writing—review and editing, H.X. and L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Beijing Natural Science Foundation (no.4234084) and Open Research Fund of Key Laboratory of Cryptography of Zhejiang Province (no.ZCL21012).

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cirac, J.I.; Zoller, P. Quantum computations with cold trapped ions. *Phys. Rev. Lett.* **1995**, *74*, 4091–4094. [CrossRef] [PubMed]
2. Wendin, G. Quantum information processing with superconducting circuits: A review. *Rep. Prog. Phys.* **2017**, *80*, 106001. [CrossRef]
3. Malinowski, M.; Allcock, D.T.C.; Ballance, C.J. How to wire a 1000-qubit trapped-ion quantum computer. *PRX Quantum* **2023**, *4*, 040313. [CrossRef]
4. Jain, S.; Sägesser, T.; Hrmo, P.; Torkzaban, C.; Stadler, M.; Oswald, R.; Axline, C.; Bautista-Salvador, A.; Ospelkaus, C.; Kienzler, D.; et al. Penning micro-trap for quantum computing. *Nature* **2024**, *627*, 510–514. [CrossRef]
5. Leung, P.H.; Landsman, K.A.; Figgatt, C.; Linke, N.M.; Monroe, C.; Brown, K.R. Robust 2-qubit gates in a linear ion crystal using a frequency-modulated driving force. *Phys. Rev. Lett.* **2018**, *120*, 020501. [CrossRef]
6. Bao, Z.; Li, Y.; Wang, Z.; Wang, J.; Yang, J.; Xiong, H.; Song, Y.; Wu, Y.; Zhang, H.; Duan, L. A cryogenic on-chip microwave pulse generator for large-scale superconducting quantum computing. *Nat. Commun.* **2024**, *15*, 5958. [CrossRef] [PubMed]
7. Zhang, Y.; Ge, Y.Q.; Liu, Y. Simulation of Kitaev chain using one-dimensional chain of superconducting qubits and environmental effects on topological states. *J. Appl. Phys.* **2024**, *136*, 064401. [CrossRef]
8. Aumentado, J.; Catelani, G.; Serniak, K. Quasiparticle poisoning in superconducting quantum computers. *Phys. Today* **2023**, *76*, 34–39. [CrossRef]
9. Arute, F.; Arya, K.; Babbush, R.; Bacon, D.; Bardin, J.C.; Barends, R.; Biswas, R.; Boixo, S.; Brandao, F.G.S.L.; Buell, D.A.; et al. Quantum supremacy using a programmable superconducting processor. *Nature* **2019**, *574*, 505–510. [CrossRef] [PubMed]
10. Edman, B.T. A Hardware-Focused Tour of IBM's 127-Qubit Eagle Processor. *Vanderbilt Undergrad. Res. J.* **2024**, *14*, 21–30.
11. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134.
12. Grover, L.K. A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 212–219.
13. Simon, D.R. On the power of quantum computation. *SIAM J. Comput.* **1997**, *10*, 1474–1483. [CrossRef]
14. Kuwakado, H.; Morii, M. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In Proceedings of the IEEE International Symposium on Information Theory, Austin, TX, USA, 13–18 June 2010; pp. 2682–2685.
15. Santoli, T.; Schaffner, C. Using Simon's algorithm to attack symmetric-key cryptographic primitives. *Quantum Inf. Comput.* **2017**, *17*, 65–78. [CrossRef]
16. Kaplan, M.; Leurent, G.; Leverrier, A.; Naya-Plasencia, M. Breaking symmetric cryptosystems using quantum period finding. In Proceedings of the CRYPTO'16: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2016; pp. 207–237.
17. Leander, G.; May, A. Grover Meets Simon–Quantumly Attacking the FX-construction. In Proceedings of the ASIACRYPT'17: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 161–178.
18. Dong, X.; Wang, X. Quantum key-recovery attack on Feistel structures. *Sci. China Inf. Sci.* **2018**, *10*, 240–246. [CrossRef]
19. Dong, X.; Wang, X. Quantum cryptanalysis on some generalized Feistel schemes. *Sci. China Inf. Sci.* **2019**, *62*, 22501:1–22501:12. [CrossRef]
20. Jaques, S.; Naehrig, M.; Roetteler, M.; Virdia, F. Implementing Grover Oracles for Quantum Key Search on AES and LowMC. In Proceedings of the EUROCRYPT'20: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; pp. 280–310.
21. Zhang, Z.; Wu, W.; Sui, H.; Wang, B. Quantum attacks on type-3 generalized Feistel scheme and unbalanced Feistel scheme with expanding functions. *Chin. J. Electron.* **2023**, *32*, 209–216. [CrossRef]
22. Xiang, Z.; Wang, X.; Yu, B.; Sun, B.; Zhang, S.; Zeng, X.; Shen, X.; Li, N. Links between Quantum Distinguishers Based on Simon's Algorithm and Truncated Differentials. *IACR Trans. Symmetric Cryptol.* **2024**, *2024*, 296–321. [CrossRef]
23. Bernstein, E.; Vazirani, U. Quantum complexity theory. *SIAM J. Comput.* **1997**, *26*, 1411–1473. [CrossRef]
24. Li, H.; Yang, L. Quantum differential cryptanalysis to the block ciphers. In Proceedings of the International Conference on Applications and Techniques in Information Security, Beijing, China, 4–6 November 2015; pp. 44–51.

25. Xie, H.; Yang, L. Using Bernstein-Vazirani algorithm to attack block ciphers. *Des. Codes Cryptogr.* **2019**, *86*, 1161–1182. [CrossRef]
26. Chen, H.; Li, Y.; Abla, P.; Li, Z.; Jiao, L.; Wang, M. Quantum Algorithm for Finding Impossible Differentials and Zero-Correlation Linear Hulls of Symmetric Ciphers. In Proceedings of the Australasian Conference on Information Security and Privacy, Brisbane, Australia, 5–7 July 2023; pp. 431–451.
27. Zhou, B.M.; Yuan, Z. Quantum Attacks without Superposition Queries: The Offline Bernstein-Vazirani Meets Grover Algorithm. In Proceedings of the 2nd International Conference on Computing, Communication, Perception and Quantum Technology, Xiamen, China, 4–7 August 2023; pp. 68–71.
28. Zhou, Q.; Lu, S.; Zhang, Z.; Sun, J. Quantum differential cryptanalysis. *Quantum Inf. Process.* **2015**, *14*, 2101–2109. [CrossRef]
29. Kaplan, M.; Leurent, G.; Leverrier, A.; Naya-Plasencia, M. Quantum differential and linear cryptanalysis. *IACR Trans. Symmetric Cryptol.* **2016**, *2016*, 71–94. [CrossRef]
30. Shi, R.; Xie, H.; Feng, H.; Yuan, F.; Liu, B. Quantum zero correlation linear cryptanalysis. *Quantum Inf. Process.* **2022**, *21*, 293. [CrossRef]
31. Hosoyamada, A.; Sasaki, Y. Finding Hash Collisions with Quantum Computers by Using Differential Trails with Smaller Probability than Birthday Bound. In Proceedings of the EUROCRYPT'20: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; pp. 249–279.
32. Dong, X.; Sun, S.; Shi, D.; Gao, F.; Wang, X.; Hu, L. Quantum Collision Attacks on AES-Like Hashing with Low Quantum Random Access Memories. In Proceedings of the ASIACRYPT'20: International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, Republic of Korea, 7–11 November 2020; pp. 727–757.
33. Denisenko, D. Quantum differential cryptanalysis. *J. Comput. Virol. Hacking Tech.* **2022**, *18*, 3–10. [CrossRef]
34. Hosoyamada, A. Quantum Speed-Up for Multidimensional (Zero Correlation) Linear Distinguishers. In Proceedings of the 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, 4–8 December 2023; pp. 311–345.
35. Xu, Y.S.; Cai, B.B.; Yuan, Z.; Qin, S.J.; Gao, F.; Wen, Q.Y. Quantum Differential Meet-In-The-Middle Attack and Some Applications to Lightweight Ciphers. *Adv. Quantum Technol.* **2024**, 2400157. [CrossRef]
36. Roetteler, M.; Steinwandt, R. A note on quantum related-key attacks. *Inf. Process. Lett.* **2015**, *115*, 40–44. [CrossRef]
37. Hosoyamada, A.; Aoki, K. On quantum related-key attacks on iterated Even-Mansour ciphers. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2019**, *102*, 27–34. [CrossRef]
38. Xie, H.; Yang, L. A quantum related-key attack based on the Bernstein-Vazirani algorithm. *Quantum Inf. Process.* **2020**, *19*, 240. [CrossRef]
39. Zhang, P. Quantum Related-Key Attack Based on Simon's Algorithm and Its Applications. *Symmetry* **2023**, *15*, 972. [CrossRef]
40. Wu, H.; Feng, X. Quantum related-key differential cryptanalysis. *Quantum Inf. Process.* **2024**, *23*, 269. [CrossRef]
41. Zou, J.; Wei, Z.; Sun, S.; Liu, X.; Wu, W. Quantum circuit implementations of AES with fewer qubits. In Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, Republic of Korea, 7–11 December 2020; pp. 697–726.
42. Kanazawa, N.; Egger, D.J.; Ben-Haim, Y.; Zhang, H.; Shanks, W.E.; Aleksandrowicz, G.; Wood, C.J. Qiskit experiments: A python package to characterize and calibrate quantum computers. *J. Open Source Softw.* **2023**, *8*, 5329. [CrossRef]
43. Tudorache, A.G. Graph Generation for Quantum States Using Qiskit and Its Application for Quantum Neural Networks. *Mathematics* **2023**, *11*, 1484. [CrossRef]
44. Khaleel, F.A.; Tawfeeq, S.K. Implementation of a modified noise-free and noisy multistage quantum cryptography protocol using QISKIT. *Quantum Stud. Math. Found.* **2024**, 1–12. [CrossRef]
45. Dasu, V.A.; Baksi, A.; Sarkar, S.; Chattopadhyay, A. Lighter-r: Optimized reversible circuit implementation for sboxes. In Proceedings of the 32nd IEEE International System-on-Chip Conference (SOCC), Singapore, 3–6 September 2019; pp. 260–265.
46. Kuwakado, H.; Morii, M. Security on the quantum-type Even-Mansour cipher. In Proceedings of the 2012 International Symposium on Information Theory and Its Applications, Honolulu, HI, USA, 28–31 October; pp. 312–316.
47. Frixons, P.; Naya-Plasencia, M.; Schrottenloher, A. Quantum boomerang attacks and some applications. In Proceedings of the 28th International Conference on Selected Areas in Cryptography, Virtual Event, 29 September–1 October 2021; pp. 332–352.
48. Zou, H.; Zou, J.; Luo, Y. New results on quantum boomerang attacks. *Quantum Inf. Process.* **2023**, *22*, 171. [CrossRef]
49. Biham, E.; Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.* **1991**, *4*, 3–72. [CrossRef]
50. Biham, E.; Biryukov, A.; Shamir, A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; pp. 12–23.
51. Knudsen, L.R. Truncated and higher order differentials. In *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings 2*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 196–211.
52. Wagner, D. The boomerang attack. In *Fast Software Encryption: 6th International Workshop, FSE'99 Rome, Italy, March 24-26, 1999 Proceedings*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 156–170.
53. Nielsen, M.; Chuang, I. *Quantum Computation and Quantum Information*, 1st ed.; Cambridge University Press: Cambridge, UK, 2000.
54. Damgård, I.; Funder, J.; Nielsen, J.B.; Salvail, L. Superposition attacks on cryptographic protocols. In Proceedings of the International Conference on Information Theoretic Security, Cham, Switzerland, 28–30 November 2013; pp. 142–161.

55. Boneh, D.; Zhandry, M. Secure signatures and chosen ciphertext security in a quantum computing world. In Proceedings of the CRYPTO'13: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2013; pp. 361–379.
56. Gagliardoni, T.; Hlsing, A.; Schaffner, C. Semantic security and indistinguishability in the quantum world. In Proceedings of the CRYPTO'16: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2016; pp. 60–89.
57. Li, H.; Yang, L. A quantum algorithm to approximate the linear structures of Boolean functions. *Math. Struct. Comput. Sci* **2018**, *28*, 1–13. [CrossRef]
58. Knudsen, L.R.; Berson, T.A. Truncated differentials of SAFER. In *Fast Software Encryption: Third International Workshop Cambridge, UK, February 21–23 1996 Proceedings 3*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 15–26.
59. Knudsen, L.R.; Robshaw, M.J. Truncated differentials and Skipjack. In Proceedings of the CRYPTO'99: 19th Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; pp. 165–180.
60. Davide, C. IBM releases first-ever 1,000-qubit quantum chip. *Nature* **2023**, *624*, 238.
61. Simulation-with-Qiskit. Available online: https://github.com/huiqinxie/Simulation-with-Qiskit (accessed on 22 August 2024).