Quantum Chosen-Cipher Attack on Camellia

Yanjun Li[1,2*], Qi Wang[2], Dingyun Huang[2], Jian Liu[1], Huiqin Xie[2]

[1*]Infomation Industry lnformation Security Evaluation Center, The 15th Research Institute of China Electronics Technology Group Corporation, Beijing, 100083, China.

[2]Department of Cryptography Science and Technology,Beijing Electronic Science and Technology Institute, Beijing, 100070, China.

*Corresponding author(s). E-mail(s): liyjwuyh@163.com;
Contributing authors: wq58416562@163.com;

Abstract

The Feistel structure represents a fundamental architectural component within the domain of symmetric cryptographic algorithms, with a substantial body of research conducted within the context of classical computing environments. Nevertheless, research into specific symmetric cryptographic algorithms utilizing the Feistel structure is relatively scarce in quantum computing environments. This paper builds upon a novel 4-round distinguisher proposed by Ito et al. for the Feistel structure under the quantum chosen-ciphertext attack (qCCA) setting. It introduces a 5-round distinguisher for Camellia. The efficacy of the distinguisher has been empirically validated. Furthermore, this paper combines Grover's algorithm with Simon's algorithm, utilizing an analysis of Camellia's key scheduling characteristics to construct a 9-round key recovery attack on Camellia algorithm. The time complexity for acquiring the correct key bits is $\mathbf{2^{61.5}}$, and it requires 531 quantum bits. This represents the inaugural chosen-ciphertext attack on Camellia under the Q2 model.

Keywords: Feistel cipher, Quantum chosen-ciphertext attacks, Grover's algorithm, Simon's algorithm, Camellia

1 Introduction

In recent years, quantum technology has undergone rapid development, and algorithms related to quantum computing have increasingly posed a threat to the security

of traditional cryptographic algorithms. Shor's algorithm[1], which is based on quantum computing, can break the classical public key cryptographic algorithm RSA in polynomial time. The advent of Grover's algorithm[2] has reduced the complexity of searching for an N-bit block cipher key from $O(N)$ to $O(\sqrt{N})$. Furthermore, Simon's algorithm[3] has facilitated the construction of quantum distinguishers.

Until 2010, quantum attacks on symmetric ciphers were not considered a significant threat. However, when Kuwakado and others[4] first introduced a polynomial distinguisher for a 3-round Feistel cipher under a quantum chosen-plaintext attack (qCPA) setting, this perspective changed. Since then, various quantum attacks on symmetric ciphers have been developed.

Zhandry[5], Kaplan[6], and others have proposed two different models for the quantum cryptanalysis of symmetric ciphers:

Standard Security (Q1 Model): a block cipher is standard secure against quantum adversaries if no efficient quantum algorithm can distinguish the block cipher from pseudorandom permutation (PRP; or a PRF) by making only classical queries.

Quantum Security (Q2 Model): a block cipher is deemed quantum secure against quantum adversaries if no efficient quantum algorithm can distinguish the block cipher from PRP (or a PRF) even by making quantum queries.

This paper assumes that the attackers belong to the Q2 model. Recent studies have analyzed the security of symmetric ciphersunder this model. In 2012, Kuwakado et al.[7] studied the quantum security of the EM cipher under the Q2 model, utilizing Simon's algorithm to construct an efficient distinguisher under the qCPA setting, thereby proving that the quantum version of the EM cipher is not secure. Subsequently, in 2015, Dinur et al.[8] combined meet-in-the-middle and partitioning attacks to enhance attacks on Feistel structures of more than four rounds. In 2016, Kaplan[9] utilized Simon's algorithm to break CBC-MAC, PMAC, and other symmetric cipher systems in polynomial time, demonstrating that Simon's algorithm could be used for slide attacks, providing exponential acceleration. Leander et al.[10] in 2017 first combined Grover's algorithm with Simon's algorithm to construct a quantum attack framework, which was later applied to the analysis of FX structures. Since then, the Grover-meets-Simon algorithm has been extensively utilized by numerous scholars in the quantum analysis of symmetric ciphers. Following Kaplan[9] et al.'s development of quantum slide attacks, Hosoyamada and colleagues[11] in 2019 expanded upon these techniques, introducing a related-key attack on the EM cipher structure and presenting a 2-round key-recovery attack on the EM cipher structure.

Dong et al.[12] combined Grover's algorithm and Simon's algorithm to introduce a new quantum key-recovery attack on Feistel structures with varying rounds. In 2019, Ito[13] and others proposed a novel distinguisher for Feistel structures under a quantum-chosen ciphertext attack (qCCA) setting. This distinguisher can differentiate, in polynomial time, between 4-round Feistel-F, Feistel-KF constructions, and 6-round Feistel-FK structures from random permutations. Subsequently, quantum key-recovery attacks for r-round Feistel-KF and Feistel-FK structures were performed, achieving key-recovery in $O(2^{\frac{(r-4)n}{4}})$ and $O(2^{\frac{(r-6)n}{4}})$ time, respectively. In the same year, Dong et al. [14] conducted a study on quantum distinguisher and key recovery attacks for two generalized Feistel structure(GFS) algorithms. They constructed

$2d-1$ rounds of quantum distinguisher against d-branch Type-1 type GFS and $2d+1$ rounds of quantum distinguisher against 2d-branch Type-2 type GFS. Using these quantum distinguishers, key-recovery attacks were conducted on the Type-1 and Type-2 GFS ciphers over d^2-d+2 and $4d$ rounds, respectively, with time complexities of $O(2^{(\frac{1}{2d^2}-\frac{3}{2d}+2)\cdot\frac{n}{2}})$ and $O(2^{\frac{d^2n}{2}})$.

Ito et al.[15] also designed a polynomial-level quantum distinguisher under the qCPA setting for $3d-3$ rounds configurations of Type-1 GFS, along with a d^2-d+1 rounds version under the qCCA seting. Based on these distinguishers, key-recovery attacks were performed on r-round Type-1 GFS ciphers, with complexities of $O(2^{\left(\frac{d^2}{2}-\frac{3d}{2}+2\right)\cdot\frac{k}{2}+\frac{\left(r-d^2\right)k}{2}})$ and $O(2^{\frac{\left(r-\left(d^2-d+1\right)\right)k}{2}})$.

Ni et al .[16] introduce a $3d-3$ rounds of quantum distinguisher on Type-1 GFS under the Q2 model and also investigated quantum attacks against the CAST-256 block cipher. In 2020, Cid et al. [17] demonstrated a qCPA on contracting-Feistel structures and studied related-key attacks on balanced-Feistel structures. That same year, Hodzic et al.[18] based on Simon's algorithm, developed a construction for 7-round and 8-round quantum distinguishers for generalized Feistel structures under the qCPA setting. In 2021, Li et al.[19] examined the round functions and linear transformation P of Camellia, presenting a 5-round quantum distinguisher and, under the qCPA setting, proposed a 7-round Camellia algorithm key-recovery attack with a complexity of 2^{24}.

Cui et al.[20] initially defined weakly periodic functions, thereby extending the application scope of Simon's algorithm and further constructing several variant Feistel structure distinguishers. They proposed quantum key-recovery attacks for Feistel variants. In 2022, Canale et al.[21] provided an automated periodic function search algorithm under a quantum computing model, implementing key-recovery attacks for the 5-round Feistel-FK structure. In 2023, Xu et al.[22], based on the divisibility of branch output functions, proposed quantum attacks on two types of GFS under the qCPA setting. They constructed quantum distinguishers for an 8-round 4F and the 5-round 2F under the Q2 model, conducting 12-round and 7-round quantum key-recovery attacks, respectively. Additionally, they constructed a 6-round 2F quantum distinguisher on a weak divisibility basis, performing a 8-round quantum key-recovery attack.

In the same year, Sun et al.[23] studied the security of Type-1 generalized Feistel structures, constructing a quantum distinguisher for a d-branch d^2-1 round Type-1 GFS structure under the qCCA setting. Furthermore, under the qCPA setting for the Type-1 block cipher CAST-256, they introduced a 17-round quantum distinguisher and constructed a quantum key-recovery attack with a complexity of $O(2^{\frac{37(r-17)}{2}})$.

Based on the findings of prior research, it is apparent that studies conducted under the qCCA model remain inadequate, and numerous aspects remain unexplored. Building upon quantum distinguishers for block cipher structures, research on quantum key-recovery attack techniques primarily relies on the quantum attack frameworks of the Grover-meets-Simon algorithm. The effectiveness of these attacks is notably influenced by the specific key scheduling algorithms employed by the block ciphers.

The paper presents a 5-round distinguisher for Camellia algorithm [24] under the qCCA setting. This is achieved by studying the round function and the characteristics of the key scheduling algorithm. Additionally, a key recovery attack model is proposed and the complexity of nine rounds of Camellia recovery key under the quantum computing model is analyzed using the distinguisher.

2 Preliminaries

2.1 Notation

The following notations are used in this study:

X_i : Output on the left side of the i-th round in the Feistel structure
X_{i-1} : Output on the right side of the i-th round
F_i : Round function of the i-th round in the Feistel structure
k_i : Round key for the i-th round

2.2 Brief description of Camellia algorithm

Camellia algorithm [24], jointly designed by NTT and Mitsubishi Electric in 2000, is known for its high security and efficient performance on both hardware and software platforms. It was selected as a winning algorithm in the European NESSIE project in 2003, recommended in Japan's CRYPTREC initiative the same year, became an IETF standard in 2004, adopted as an ISO/IEC standard in 2005.

Camellia is based on a Feistel structure with a block length of 128 bits and supports key lengths of 128, 192, and 256 bits, corresponding to 18, 24, and 32 rounds respectively.

2.2.1 Camellia encryption Transformation

The encryption transformation of Camellia involves differing initial and final whitening keys, with FL/FL^{-1} functions inserted every 6 rounds. For the 128-bit key version, the process consists of three 6-round Feistel structures and two rounds of FL/FL^{-1} functions. Below, Camellia with a 128-bit key is described as shown in Figure 1.

The plaintext M is 128 bits, the whitening key $kw_i(1 \leq i \leq 4)$ is 64 bits, and the round key $k_i(1 \leq i \leq 18)$ is 64 bits. The key $kl_i(1 \leq i \leq 4)$ utilized in each FL/FL^{-1} function is 32 bits, and the final output ciphertext C is 128 bits. The specific encryption process is as follows:

1. Plaintext Whitening

   A 128-bit plaintext M undergoes an XOR operation with the whitening key $kw_1 \| kw_2$, resulting in two parts: the left 64 bits X_{-1} and the right 64 bits X_0, such that $M \oplus (kw_1 \| kw_2) = X_{-1} \| X_0$.
2. Round Iteration

   For each round of the Feistel structure, let X_i denote the left output of the i-th round, and X_{i-1} denote the right output of the i-th round. For $i = 1, 2, \ldots, 18$, excluding $i = 6$ and $i = 12$, the i-th round transformation is performed as follows:

$$X_i = X_{i-2} \oplus F\left(X_{i-1}, k_i\right) \tag{1}$$

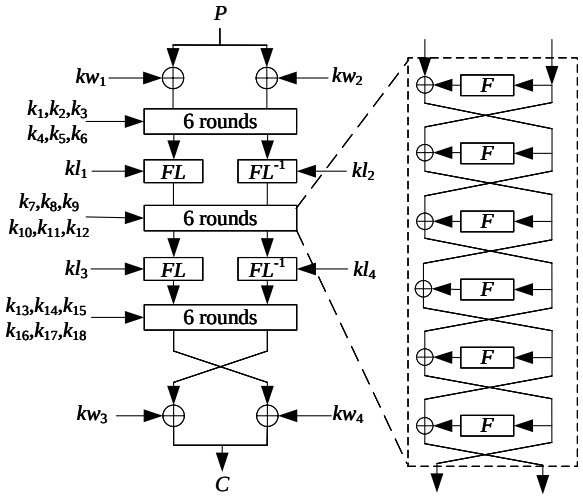

Fig. 1 Encryption process of Camellia

For $i = 6, 12$, the transformation is as follows:

$$\begin{aligned} X'_i &= X_{i-2} \oplus F\left(X_{i-1}, k_i\right) \quad X_i = FL\left(X'_i, kl_{i/3-1}\right) \\ X_{i-1} &= FL^{-1}(X_{i-1}, kl_{i/3}) \end{aligned} \tag{2}$$

3. Pre-whitening of ciphertext output

The final round output $X_{18}\|X_{17}$ is XORed with the whitening keys $kw_3\|kw_4$, producing the whitened ciphertext $C = (X_{18}\|X_{17}) \oplus (kw_3\|kw_4)$.

The FL and FL^{-1} transformations are defined as follows:

$$\begin{aligned} &FL : F_2^{64} \to F_2^{64}, \\ &(X_L \|X_R, kl_L\| \, kl_R) \mapsto Y_L\|Y_R, \\ &Y_R = ((X_L \cap kl_L) \ll 1) \oplus X_R, \quad Y_L = (Y_R \cup kl_R) \oplus X_L \\ &FL^{-1} : \mathrm{F}_2^{64} \to \mathrm{F}_2^{64}, \\ &(Y_L \|Y_R, kl_R\| \, kl_L) \mapsto X_L\|X_R, \\ &X_L = (Y_R \cup kl_R) \oplus Y_L, \quad X_R = ((X_L \cap kl_L) \ll 1) \oplus Y_R. \end{aligned} \tag{3}$$

where $\cap$ represents bitwise logical "AND" operation; $\cup$ represents bitwise logical "OR" operation. In the Feistel structure, the function F during step 2 utilizes an SP-structure design that incorporates round key XOR operations, S-box lookups, and the permutation **P**. The final output of function F is formed by the outputs of eight S-boxes after undergoing the permutation **P**, as depicted in Figure 2. The specific steps involved are outlined below:

1. Round Key XOR
   A 64-bit input is divided into 8 bytes. Each byte is then XORed with a corresponding round key byte before proceeding to the next step.
2. S-Box Lookup
   The XORed bytes sequentially query 8 S-boxes in the order of $s1, s2, s3, s4, s2, s3, s4, s1$.
3. Permutation $\mathbf{P}$
   The output from the S-boxes undergoes a linear transformation, described as follows.

$$\begin{aligned} y_1 &= x_1 \oplus x_3 \oplus x_4 \oplus x_6 \oplus x_7 \oplus x_8; y_5 = x_1 \oplus x_2 \oplus x_6 \oplus x_7 \oplus x_8 \\ y_2 &= x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_7 \oplus x_8; y_6 = x_2 \oplus x_3 \oplus x_5 \oplus x_7 \oplus x_8 \\ y_3 &= x_1 \oplus x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_8; y_7 = x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_8 \\ y_4 &= x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7; y_8 = x_1 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \end{aligned} \tag{4}$$

The final output order is $(y_8, y_7, y_6, y_5, y_4, y_3, y_2, y_1)$. The diffusion layer $\mathbf{P}$ and its inverse $\mathbf{P}^{-1}$ have the following coefficient matrices:

$$\mathbf{P} = \begin{bmatrix} 0&1&1&1&1&0&0&1 \\ 1&0&1&1&1&1&0&0 \\ 1&1&0&1&0&1&1&0 \\ 1&1&1&0&0&0&1&1 \\ 0&1&1&1&1&1&1&0 \\ 1&0&1&1&0&1&1&1 \\ 1&1&0&1&1&0&1&1 \\ 1&1&1&0&1&1&0&1 \end{bmatrix}, \mathbf{P}^{-1} = \begin{bmatrix} 1&1&1&0&1&0&0&1 \\ 0&1&1&1&1&1&0&0 \\ 1&0&1&1&0&1&1&0 \\ 1&1&0&1&0&0&1&1 \\ 0&1&1&1&0&1&1&1 \\ 1&0&1&1&1&0&1&1 \\ 1&1&0&1&1&1&0&1 \\ 1&1&1&0&1&1&1&0 \end{bmatrix} \tag{5}$$

2.2.2 Key expansion algorithm

The round keys used in the encryption process are generated from a 256 -bit initial key $k_{L(128)} \| k_{R(128)}$. Initially, $k_{L(128)} \| k_{R(128)}$ is input into the Feistel structure with round constants $\Sigma 1, \Sigma 2, \Sigma 3, \Sigma 4, \Sigma 5, \Sigma 6$. After four rounds, 128-bit $k_{A(128)}$ is generated, and after six rounds, 128 -bit $k_{B(128)}$ is produced. The structure of the algorithm is shown in Figure 3. The six round constants involved in generating $k_{A(128)}$ and $k_{B(128)}$ are:

$$\begin{aligned} \Sigma_1 &= 0xA09E667F3BCC908B \\ \Sigma_2 &= 0xB67AE8584CAA73B2 \\ \Sigma_3 &= 0xC6EF372FE94F82BE \\ \Sigma_4 &= 0x54FF53A5F1D26F1C \\ \Sigma_5 &= 0x10E527FADE682D1D \\ \Sigma_6 &= 0xB05688C2B3E6C1FD \end{aligned} \tag{6}$$

In the 128-bit version of the master key, the 256-bit initial key is defined as: $k_{L(128)} \left\| k_{R(128)} = k \right\| 0$. The round keys for each round are derived by shift transformations of the initial key K_L and K_A, as summarized in Table 1.

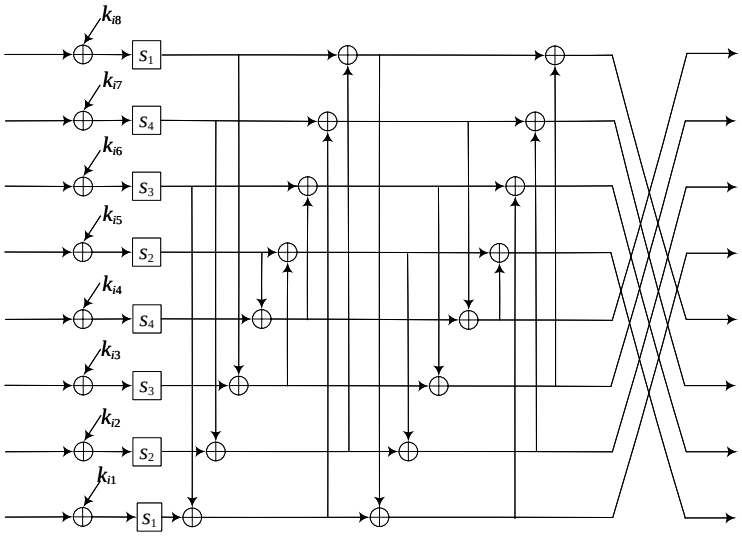

Fig. 2 The round function of Camellia

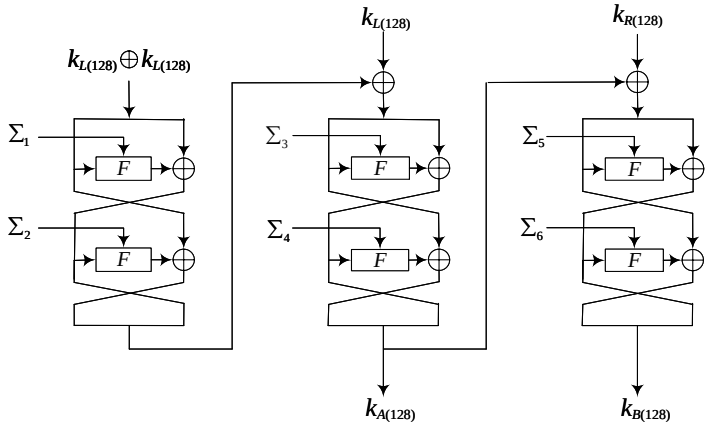

Fig. 3 The key expansion algorithm of Camellia

2.3 Related algorithms

In this section, we offer a concise overview of classical quantum algorithms, specifically Simon's algorithm, Grover's algorithm, and the Grover-meets-Simon algorithm.

Table 1 Wheel keys for each wheel

Round	Round key	Round key value	Round	Round key	Round key value
Pre-whitening	$kw_{1(64)}$	$(k_L <<< 0)_{L(64)}$	F(Round10)	$k_{10(64)}$	$(k_L <<< 60)_{R(64)}$
Pre-whitening	$kw_{2(64)}$	$(k_L <<< 0)_{R(64)}$	F(Round11)	$k_{11(64)}$	$(k_A <<< 60)_{L(64)}$
F(Round1)	$k_{1(64)}$	$(k_A <<< 0)_{L(64)}$	F(Round12)	$k_{12(64)}$	$(k_A <<< 60)_{R(64)}$
F(Round2)	$k_{2(64)}$	$(k_A <<< 0)_{R(64)}$	FL	$kl_{3(64)}$	$(k_L <<< 77)_{L(64)}$
F(Round3)	$k_{3(64)}$	$(k_L <<< 15)_{L(64)}$	FL^{-1}	$kl_{4(64)}$	$(k_L <<< 77)_{R(64)}$
F(Round4)	$k_{4(64)}$	$(k_L <<< 15)_{R(64)}$	F(Round13)	$k_{13(64)}$	$(k_L <<< 94)_{L(64)}$
F(Round5)	$k_{5(64)}$	$(k_A <<< 15)_{L(64)}$	F(Round14)	$k_{14(64)}$	$(k_L <<< 94)_{R(64)}$
F(Round6)	$k_{6(64)}$	$(k_A <<< 15)_{R(64)}$	F(Round15)	$k_{15(64)}$	$(k_A <<< 94)_{L(64)}$
FL	$kl_{1(64)}$	$(k_A <<< 30)_{L(64)}$	F(Round16)	$k_{16(64)}$	$(k_A <<< 94)_{R(64)}$
FL^{-1}	$kl_{2(64)}$	$(k_A <<< 30)_{R(64)}$	F(Round17)	$k_{17(64)}$	$(k_L <<< 111)_{L(64)}$
F(Round7)	$k_{7(64)}$	$(k_L <<< 45)_{L(64)}$	F(Round18)	$k_{18(64)}$	$(k_L <<< 111)_{R(64)}$
F(Round8)	$k_{8(64)}$	$(k_L <<< 45)_{R(64)}$	post-whitening	$kw_{3(64)}$	$(k_A <<< 111)_{L(64)}$
F(Round9)	$k_{9(64)}$	$(k_A <<< 45)_{L(64)}$	post-whitening	$kw_{4(64)}$	$(k_A <<< 111)_{R(64)}$

2.3.1 Simon's algorithm

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}^n$, which is guaranteed to satisfy $f(x) = f(y) \Leftrightarrow x \oplus y \in \{0, s\}$. It means that the function has a period s and we need to find s. Classically, the optimal time to find period s is $O\left(2^{n/2}\right)$.Nevertheless, Simon [3] introduced an algorithm that significantly expedites this process, requiring only $O(n)$ queries to determine s. This algorithm comprises the following five steps:

1. Initialize two n-bit quantum registers in state $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$ and apply the Hadamard transform to the first register to obtain the corresponding superposition state.

$$H^{\otimes n}|0\rangle|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|0\rangle \tag{7}$$

2. Conduct a quantum query on function f and map it to the current state.

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|f(x)\rangle \tag{8}$$

3. Measure the second register, reducing the first register to the following state:

$$\frac{1}{\sqrt{2}}(|z\rangle + |z \oplus s\rangle) \tag{9}$$

4. Apply the Hadamard transform to the first register to obtain

$$\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{y\cdot z} \left(1 + (-1)^{y\cdot s}\right) |y\rangle \tag{10}$$

5. In this superposition state, the amplitudes corresponding to $y \cdot s = 1$ are zero. As a result, for any measurement of y, it is always true that $y \cdot s = 0$. By iterating this process $O(n)$ times, a set of linear equations can be constructed. Solving this system of equations results in determining the value of s.

At ISIT2010, Kuwawkado et al.[4] presented a quantum distinguishing attack on a three-round Feistel cipher constructed using Simon's algorithm. As illustrated in Figure 4, α_0 and α_1 are arbitrary constants.

$$\begin{aligned} &f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n \\ &b, x \to \alpha_b \oplus X_2, (X_3, X_2) = E(\alpha_b, x) \\ &f(b,x) = F_2(F_1(\alpha_b) \oplus x) \end{aligned} \tag{11}$$

Let f be a periodic function satisfying $f(b,x) = f(b \oplus 1, x \oplus F_1(\alpha_0) \oplus F_1(\alpha_1))$. Subsequently, the period $s = 1 \| F_1(\alpha_0) \oplus F_1(\alpha_1)$ can be obtained in polynomial time by employing Simon's algorithm.

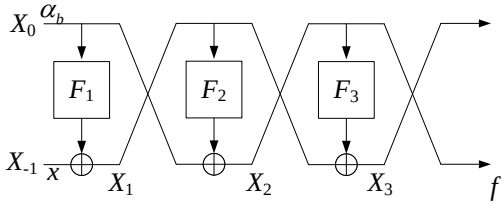

Fig. 4 The periodic function of 3-round Feistel structure under qCPA setting

2.3.2 Grover's algorithm

When dealing with an unordered set of $N = 2^n$ elements, Grover's algorithm [2] is employed to pinpoint a unique element that fulfills certain criteria. Specifically, a quantum oracle O is used, which performs the operation $O|x\rangle = (-1)^{f(x)}|x\rangle$, where $f(x) = 0$ for all x except x_0 within the range $0 \leq x < 2^n$, and $f(x_0) = 1$. The goal is to determine x_0. The most efficient classical algorithm for searching this unordered data has a time complexity of $O(N)$. However, Grover's algorithm, executed on a quantum computer, dramatically reduces this to merely $O(\sqrt{N})$ operations. The algorithm proceeds as follows:

1. Initialize an n-bit register $|0\rangle^{\otimes n}$ and apply the Hadamard transform to the first register to achieve the corresponding superposition state, as shown in equation (12):

$$H^{\otimes n}|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle = |\varphi\rangle \tag{12}$$

2. Construct a quantum oracle $O : |x\rangle \to (-1)^{f(x)}|x\rangle$, if x is the correct state, then $f(x) = 1$; otherwise, $f(x) = 0$.

3. Define the Grover iteration: $(2|\varphi\rangle\langle\varphi| - I)O$, and iterate this operation $R \approx \pi\sqrt{2^n}/4$ times:

$$[(2|\varphi\rangle\langle\varphi| - I)O]^R|\varphi\rangle \approx |x_0\rangle \tag{13}$$

4. Return x_0.

2.3.3 Grover-meets-Simon

During the 2017 ASIACRYPT conference, Leander et al.[10] presented a quantum key recovery attack approach that integrates Grover's algorithm with Simon's algorithm, specifically targeting the FX structure, depicted in Figure 5. The FX structure fulfills the given equation:

$$\mathrm{Enc}(x) = E_{k_0}\left(x + k_1\right) + k_2 \tag{14}$$

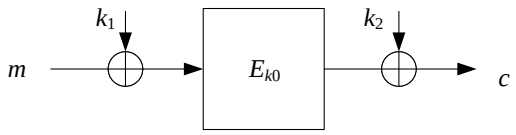

Fig. 5 The structure of FX

Reference [10] constructs the function $f(k, x) = \mathrm{Enc}(x) + E_k(x) = E_{k_0}\left(x + k_1\right) + k_2 + E_k(x)$. When the correct key guess $k = k_0$, it holds that $f(k, x) = f\left(k, x + k_1\right)$. However, for $k \neq k_0$, the function is not periodic. Under the qCPA setting, Reference [10] employs a combination of Simon's algorithm and Grover's algorithm to attack the FX structure.

Based on the work of Leander [10], Hosoyamada et al [25], and Dong et al [12] added a few rounds behind the 3-round Feistel structured distinguisher shown in Figure 4 for recovering the keys of the Feistel encryption algorithm for the r rounds, with a time complexity of $O\left(2^{(r-3)n/2}\right)$.

3 Construction of periodic functions

This chapter presents a brief description of the periodic function of 4-round Feistel structure proposed by ITO et al[13]. Additionally, a periodic function for Camellia algorithm is constructed and verified to be correct.

3.1 The periodic function of 4-round Feistel structure

At RSA 2019, Ito et al. introduced the design of periodic functions for a 4-round Feistel cipher. They developed a quantum distinguisher and presented a key recovery attack against the Feistel structure. We will now describe the method for constructing

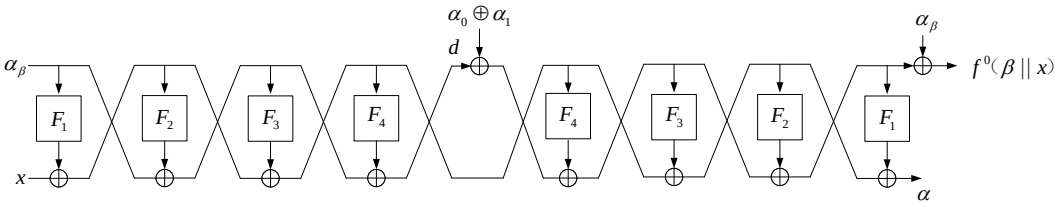

Fig. 6 The periodic function of 4-round Feistel structure

a periodic function for a 4-round Feistel structure,as proposed in Reference [13]. Figure 6 illustrates the structure of the periodic function.

The 4-round Feistel-F encryption structure is denoted as FF_4, and its corresponding decryption structure is represented by FF_4^{-1}. The round functions of Feistel-F are signified by $F_1, \ldots, F_4 \in \mathrm{Func}\,(\,n/2\,)$. The plaintext input $(a, b) \in \left(\{0,1\}^{n/2}\right)^2$ to FF_4, which outputs the ciphertext $(c, d) \in \left(\{0,1\}^{n/2}\right)^2$. The encryption structure $(a, b) \mapsto (c, d)$ is defined as follows:

$$\begin{aligned} c &= a \oplus F_1(b) \oplus F_3\left(b \oplus F_2\left(a \oplus F_1(b)\right)\right) \\ d &= b \oplus F_2\left(a \oplus F_1(b)\right) \oplus F_4\left(a \oplus F_1(b) \oplus F_3\left(b \oplus F_2\left(a \oplus F_1(b)\right)\right)\right) \end{aligned} \tag{15}$$

The decryption structure $(c, d) \mapsto (a, b)$ is defined as follow:

$$\begin{aligned} a &= c \oplus F_3\left(d \oplus F_4(c)\right) \oplus F_1\left(d \oplus F_4(c) \oplus F_2\left(c \oplus F_3\left(d \oplus F_4(c)\right)\right)\right) \\ b &= d \oplus F_4(c) \oplus F_2\left(c \oplus F_3\left(d \oplus F_4(c)\right)\right) \end{aligned} \tag{16}$$

For the input plaintext (α_β, x), the encryption and decryption structures can be further simplified.The simplified structure is depicted in Figure 7. The function $f^o(\beta \| x)$ is described as:

$$\begin{aligned} f^o(\beta \| x) = \alpha_0 \oplus \alpha_1 \oplus F_2\left(x \oplus F_1\left(\alpha_\beta\right)\right) \oplus F_2\left(x \oplus F_1\left(\alpha_\beta\right) \oplus F_3\left(\alpha_\beta \oplus F_2\left(x \oplus F_1\left(\alpha_\beta\right)\right)\right)\right. \\ \left. \oplus F_3\left(\alpha_\beta \oplus \alpha_0 \oplus \alpha_1 \oplus F_2\left(x \oplus F_1\left(\alpha_\beta\right)\right)\right)\right) \end{aligned} \tag{17}$$

Theorem 1. *Define* $Z_{(\beta \| x)} = F_1\left(\alpha_\beta\right) \oplus x$. *Then* $Z_{(\beta \| x)}$ *is a periodic function with a period* $s = 1 \| \left(F_1\left(\alpha_0\right) \oplus F_1\left(\alpha_1\right)\right)$.

Proof. $Z_{((\beta \| x) \oplus s)} = x \oplus F_1\left(\alpha_0\right) \oplus F_1\left(\alpha_1\right) \oplus F_1\left(\alpha_{(\beta \oplus 1)}\right) = x \oplus F_1\left(\alpha_\beta\right) = Z_{(\beta \| x)}$. □

It is straightforward to demonstrate that $Z_{(\beta \| x)}$ is a periodic function. The output function $f^\circ(\beta \| x)$ can be described as follows:

$$f^o(\beta \| x) = \alpha_0 \oplus \alpha_1 \oplus F_2\left(Z_{\beta \| x}\right) \oplus F_2\left(Z_{\beta \| x} \oplus F_3\left(\alpha_0 \oplus F_2\left(Z_{\beta \| x}\right)\right) \oplus F_3\left(\alpha_1 \oplus F_2\left(Z_{\beta \| x}\right)\right)\right) \tag{18}$$

It can be deduced that $f^o(\beta \| x)$ is a periodic function with a period of s.

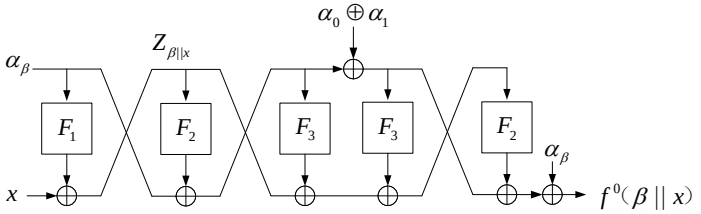

Fig. 7 The equivalent structure of 4-round Feistel structure periodic function

3.2 Construction of periodic functions for Camellia

In this section, we aim to construct a periodic function for Camellia, leveraging the periodic function construction methods outlined in the preceding section. To commence, we construct the 5-round periodic function structure as illustrated in Figure 8.

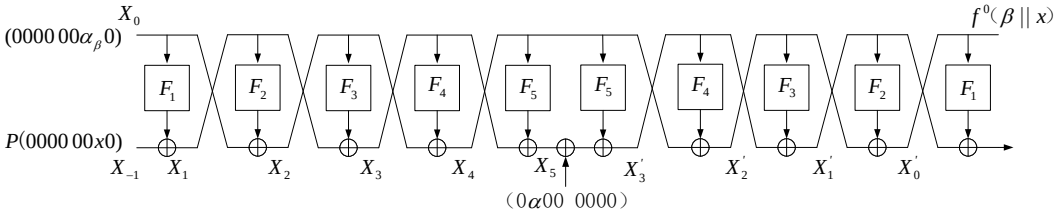

Fig. 8 The construction of Camellia's periodic function

In the illustrated structure, F_5 and F_1 do not participate in the construction of the periodic function $f^o(\beta\|x)$, thus the structure can be further simplified as shown in Figure 9.

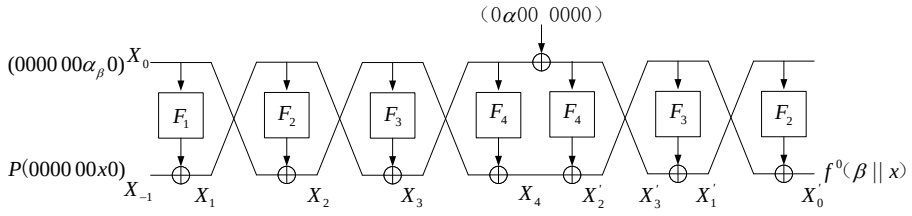

Fig. 9 The quivalent structure of Camellia periodic function

Let the inputs for the two branches be $(\mathbf{000000}\alpha_\beta\mathbf{0})$ and $\mathbf{P}(\mathbf{000000}x\mathbf{0})$, where $\mathbf{0}$ represents a sequence of eight zero bits. The variables α_β and x are both byte variables,

both of which are located at the sixth byte. Byte indexing starts at 0 in this article. The input $\mathbf{P}(\mathbf{000000}x\mathbf{0})$ is obtained from $(\mathbf{000000}x\mathbf{0})$ through a permutation $\mathbf{P}$.

After the first round function transformation, we get the output X_1 as

$$X_1 = F_1(X_0) \oplus X_{-1} = F_1(\mathbf{000000}\alpha_\beta\mathbf{0}) \oplus \mathbf{P}(\mathbf{000000}x\mathbf{0}) = \mathbf{P}(\mathbf{000000} * \mathbf{0}) \quad (19)$$

where $* = s_1(\alpha_\beta) \oplus x$, s_i is the s transformation of the i-th round function. Given the characteristics of Camellia algorithm's P permutation matrix, we get the output X_2 as:

$$X_2 = F_2(X_1) \oplus X_0 = \mathbf{P}(\mathbf{00}\Delta\Delta\Delta\Delta\Delta\mathbf{0}) \oplus (\mathbf{000000}\alpha_\beta\mathbf{0}) \quad (20)$$

Within this structure, each symbol Δ is distinct and consists solely of bytes that are related to $*$. The output obtained above continues to participate in the round function operation, and we can obtain the subsequent output as shown in the following equations:

$$\begin{aligned} X_3 &= X_1 \oplus F_3(X_2) = \mathbf{P}(\mathbf{000000} * \mathbf{0}) \oplus F_3(\mathbf{P}(\mathbf{00}\Delta\Delta\Delta\Delta\Delta\mathbf{0}) \oplus (\mathbf{000000}\alpha_\beta\mathbf{0})) \\ &= \mathbf{P}(\mathbf{000000} * \mathbf{0}) \oplus F_3((\Delta\Delta\Delta\Delta\Delta\Delta\Delta\Delta) \oplus (\mathbf{000000}\alpha_\beta\mathbf{0})) \\ &= \mathbf{P}(\mathbf{000000} * \mathbf{0}) \oplus F_3(\Delta\Delta\Delta\Delta\Delta\Delta?\Delta) \\ &= \mathbf{P}(\mathbf{000000} * \mathbf{0}) \oplus \mathbf{P}(\Delta\Delta\Delta\Delta\Delta\Delta?\Delta) \\ &= \mathbf{P}(\Delta\Delta\Delta\Delta\Delta\Delta?\Delta) \end{aligned} \quad (21)$$

The symbol ? is used to indicate a byte where the periodicity of the function cannot be determined.

$$\begin{aligned} X_4 &= F_4(X_3) \oplus X_2 = F_4(\mathbf{P}(\Delta\Delta\Delta\Delta\Delta\Delta?\Delta)) \oplus \mathbf{P}(\mathbf{00}\Delta\Delta\Delta\Delta\Delta\mathbf{0}) \oplus (\mathbf{000000}\alpha_\beta\mathbf{0}) \\ &= F_4(\Delta\Delta?????\Delta) \oplus \mathbf{P}(\mathbf{00}\Delta\Delta\Delta\Delta\Delta\mathbf{0}) \oplus (\mathbf{000000}\alpha_\beta\mathbf{0}) \end{aligned} \quad (22)$$

$$\begin{aligned} X_2' &= F_4(X_3 \oplus (\mathbf{0}\alpha\mathbf{000000})) \oplus X_4 = F_4((\Delta\Delta?????\Delta) \oplus (\mathbf{0}\alpha\mathbf{000000})) \oplus X_4 \\ &= \mathbf{P}(\Delta\alpha'?????\Delta) \oplus \mathbf{P}(\Delta\Delta?????\Delta) \oplus \mathbf{P}(\mathbf{00}\Delta\Delta\Delta\Delta\Delta\mathbf{0}) \oplus (\mathbf{000000}\alpha_\beta\mathbf{0}) \\ &= \mathbf{P}(\Delta\alpha'\Delta\Delta\Delta\Delta\Delta\Delta) \oplus (\mathbf{000000}\alpha_\beta\mathbf{0}) \end{aligned} \quad (23)$$

$$X_1' = F_3(X_2') \oplus X_3' = \mathbf{P}(\Delta\Delta\Delta\Delta\Delta\Delta?\Delta) \oplus X_4 \oplus (\mathbf{0}\alpha\mathbf{000000}) = \mathbf{P}(\Delta\Delta\Delta\Delta\Delta\Delta?\Delta) \quad (24)$$

$$X_0' = X_2' \oplus F_2(X_1') = \mathbf{P}(\Delta\alpha'\Delta\Delta\Delta\Delta\Delta\Delta) \oplus (\mathbf{000000}\alpha_\beta\mathbf{0}) \oplus S_2\mathbf{P}(\mathbf{P}(\Delta\Delta\Delta\Delta\Delta\Delta?\Delta)) \quad (25)$$

Perform the $\mathbf{P}^{-1}$ operation on both sides of the above equation, we obtain:

$$\begin{aligned}\mathbf{P}^{-1}X_0' &= (\Delta\alpha'\Delta\Delta\Delta\Delta\Delta\Delta) \oplus \mathbf{P}^{-1}(\mathbf{000000}\alpha_\beta\mathbf{0}) \oplus S_2(\Delta\Delta?????\Delta) \\ &= (\Delta\alpha'\Delta\Delta\Delta\Delta\Delta\Delta) \oplus (\mathbf{00}\alpha_\beta\alpha_\beta\alpha_\beta\alpha_\beta\mathbf{0}\alpha_\beta) \oplus (\Delta\Delta?????\Delta) \\ &= (\Delta\Delta??????)\end{aligned} \tag{26}$$

As derived above, the 0 th and 1st bytes of $\mathbf{P}^{-1}X_0'$ relate only to the variable $*$, thus they can be used to construct a periodic function. Define the output byte $\mathbf{P}^{-1}(X_0')_1$ as the function $f^o(\beta, x)$, i.e., $f^o(\beta, x) = \left(\mathbf{P}^{-1}(X_0')_1\right)$. The theorem states as follows:

Theorem 2. *The period of the function $f^o(\beta, x)$ is $s = 1\|s_1(\alpha_0) \oplus s_1(\alpha_1)$.*

$$f^o(\beta, x) = f^o(\beta \oplus 1, x \oplus s_1(\alpha_0) \oplus s_1(\alpha_1))$$

Proof. When $\beta = 0$, for $f^o(\beta\|x), * = s(\alpha_0) \oplus x = s(\alpha_{0\oplus 1}) \oplus x \oplus s(\alpha_0) \oplus s(\alpha_1) = s(\alpha_0) \oplus x$. Since the value of $f^o(\beta\|x)$ relates only to $*$, it follows that $f^o(0\|x) = f^o(0 \oplus 1\|x \oplus s(\alpha_0) \oplus s(\alpha_1))$.

When $\beta = 1$, for $f^o(\beta\|x), * = s(\alpha_1) \oplus x = s(\alpha_{1\oplus 1}) \oplus x \oplus s(\alpha_0) \oplus s(\alpha_1) = s(\alpha_1) \oplus x$. Since the value of $f^o(\beta\|x)$ relates only to $*$, it follows that $f^o(1\|x) = f^o(1 \oplus 1\|x \oplus s(\alpha_0) \oplus s(\alpha_1))$. □

It can be demonstrated that the function $f^o(\beta\|x)$ exhibits periodicity. In accordance with Theorem 2 of the literature [13], a distinguisher against the 5-round Camellia can be constructed using the function $f^o(\beta\|x)$.

3.3 Experimental validation

In this subsection, we conducted targeted experiments to validate the correctness of the periodic functions developed in the preceding section.In accordance with the Camellia algorithm criteria outlined in the literature[24], we have constructed the 5-round periodic function structure presented in Subsection 3.2 on the Python 3.7 environment. Our primary aim was to verify the equation $f^\circ(\beta, x) = f^\circ(\beta \oplus 1, x \oplus s_1(\alpha_0) \oplus s_1(\alpha_1))$. Consequently, we gathered a diverse array of data sets for periodicity testing, and the results consistently upheld the accuracy of the periodic function. As illustrated in Figure10, the periodic function is demonstrated to be correct with a specific set of data.

The input plaintext group data is $(\mathbf{000000}\alpha_\beta\mathbf{0}), \mathbf{P}(\mathbf{000000}x\mathbf{0})$, where $x = (00000000), \alpha_0 = (00000000)$, and $\alpha_1 = (00000001)$. When $\beta = 0$, for $f^\circ(\beta, x)$, the plaintext input comprises solely zeros. The output for the first byte, denoted as $\left(\mathbf{P}^{-1}\right)X_0'$, is (01101100).

For $f^\circ(\beta \oplus 1, x \oplus s_1(\alpha_0) \oplus s_1(\alpha_1))$, the plaintext group data is updated to $(0000000\alpha_1), \mathbf{P}(000000x \oplus s_1(\alpha_0) \oplus s_1(\alpha_1)\,0\,)$, and remarkably, the output for the first byte, again denoted as $\left(\mathbf{P}^{-1}\right)X_0'$, remains unchanged at (01101100).

Given the experimental verification outlined above, Theorem 1 is confirmed to be accurate.

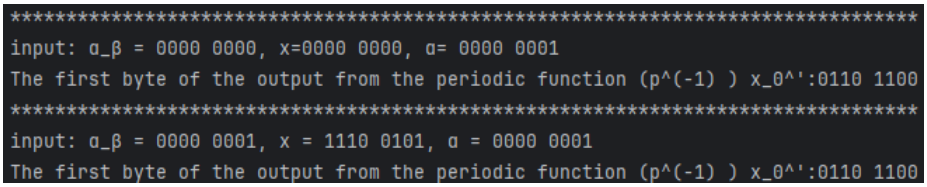

Fig. 10 Experimental result

4 The attack model for Camellia

This section presents a key-recovery attack model for Camellia algorithm under the qCCA setting, leveraging the 5-round distinguisher proposed in Subsection 3.2. Here is an outline of our attack methodology:

1. Implement a 9-round encryption oracle, denoted as $\mathcal{E}$, and decrypt the input of the periodic function f_{in} over two rounds. The resulting intermediate value after the 2 -rounds and the subkeys for the 2 -rounds as the input to the circuit $\mathcal{E}$. Then, the output of $\mathcal{E}$ undergoes two rounds of decryption and is XORed with a constant.
2. Implement a quantum circuit $\mathcal{D}$ that computes the inverse of $\mathcal{E}$. Encrypt the results from the previous step over two rounds, inputting the intermediate value along with subkeys into the circuit $\mathcal{D}$. Subsequently, encrypt $\mathcal{D}$'s output over two rounds to acquire the periodic function output f_{out} .
3. Guess the keys for the two rounds preceding and succeeding the quantum circuits $\mathcal{E}$ and $\mathcal{D}$.
4. For each key guess, check its correctness with the following procedure.
   a) Apply the 5 -round distinguisher to $\mathcal{E}$ and $\mathcal{D}$.
   b) If the distinguisher returns "this is a random permutation", then judge that the guess is wrong. Otherwise judge that the guess is correct.

4.1 9-round key-recovery attack on Camellia

Utilizing the established attack model, a 9-round key-recovery attack is executed, as depicted in Figure 11. The periodic function input f_{in} is configured as $[\mathbf{P}(\mathbf{000000}x\mathbf{0}), (\mathbf{000000}\alpha_\beta\mathbf{0})]$, with the output f_{out} being $\mathbf{P}^{-1}((X_7)_1)$.To decrypt f_{in} through two rounds and retrieve the plaintext, the keys K_1 and $K_{2,\{2,3,4,5,6\}}$ need to be guessed. Subsequently, the ciphertext is obtained after nine rounds of encryption by the oracle. In addition, the keys to be guessed for the decryption of the ciphertext for two rounds are $K_{9,\{0,3,4,5,6\}}$ and $K_{8,7}$. After xOR the obtained intermediate state with the constant $[(\mathbf{00000000}), (\mathbf{0}\alpha\mathbf{000000})]$, it needs to guess the keys $K_{8,7}$ and $K_{9,\{0,3,5,6,7\}}$ to encrypt for two rounds to obtain the ciphertext. The plaintext is then obtained by nine rounds of decryption oracle. Finally, the keys K_1 and $K_{2,1}$ must be guessed in order to encrypt the plaintext for two rounds, thereby obtaining the output f_{out} .

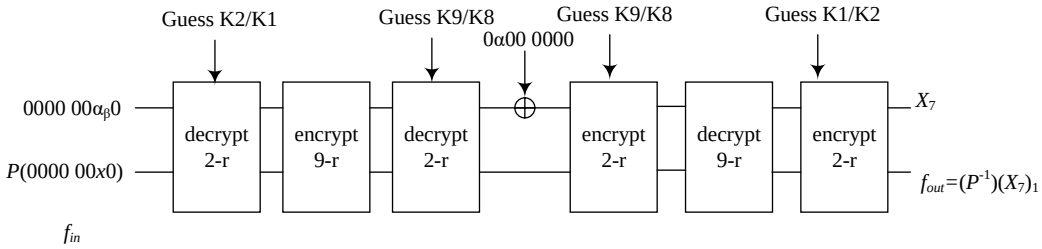

Fig. 11 The key-recovery attack model of Camellia

Due to the characteristics of Camellia's key scheduling, where round keys for F (Round 1), F (Round 2), and F (Round 9) are derived from cyclic shifts of K_A, many key bits are repeated. We guess all key bits for Round 1 and bytes 1-6 for Round 2. Round 9 requires guesses for 0 and 3-7 bytes of the key, with 45 bits being repetitions. As shown in Figure 12,the orange portion of the figure indicates the 3-bit non-repeating key positions that must be guessed in round 9. Thus, the actual key bits to guess are $K_{A[0-63]}, K_{A[72-119]}, K_{L[37-44]}, K_{A[69,70,71]}$, totaling $64+ 48 + 8 + 3 = 123$ bits.

r	Key bits																													
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
2	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
n	…																															
9	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108

Fig. 12 Camellia round key duplicate bits

Define $g : F_2^{64} \times F_2^{48} \times F_2^{8} \times F_2^{32} \times F_2^{(8+1)} \rightarrow F_2^{8}$ satisfying $\left(K_{A[0-63]}, K_{A[72-119]}, K_{L[37-44]}, K_{A[69,70,71]}, y\right) \rightarrow f(y)$, where $y = f^o(\beta, x)$. If the key guess is correct, the following holds true:

$$\begin{aligned} &g\left(K_{\mathrm{A}[0-63]}, K_{\mathrm{A}[72-119]}, K_{\mathrm{L}[37-44]}, K_{\mathrm{A}[69,70,71]}, y\right) = \\ &g\left(K_{\mathrm{A}[0-63]}, K_{\mathrm{A}[72-119]}, K_{\mathrm{L}[37-44]}, K_{\mathrm{A}[69,70,71]}, y \oplus s\right) \end{aligned} \tag{27}$$

If $f_{\mathrm{in}} \rightarrow f_{\mathrm{out}}$ is a periodic function, then the period of this function can be determined by inputing it into Simon's algorithm. The guess is correct; otherwise, the guess is incorrect.

As outlined in the literature [10], the formula for the number of quantum bits required for a key recovery attack is as follows:

$$\text{sum } = n_k + n_{\text{in}} \times l + n_{\text{out}} \times l, l = 2(\tilde{n} + \sqrt{n}) \tag{28}$$

where sum represents the total number of quantum bits required, n_k the length of the key, n_{in} the input length of the periodic function, n_{out} the output length of the periodic function, and $\tilde{n}$ the length of the period. For Camellia's guessed keys $K_A[0-63], K_A[72-119], K_L[37-44], K_A[69,70,71]$, we have that:

$$\begin{aligned} &n_k = 64 + 48 + 8 + 3 = 123, n_{\text{in}} = 8 + 1 = 9, \tilde{n} = 8 + 1 = 9 \\ &n_{\text{out}} = 8, l = 2(9 + \sqrt{9}) = 24, \text{ sum } = 123 + 9 \times 24 + 8 \times 24 = 531 \end{aligned} \tag{29}$$

The whole attack needs $123 + 9 \times 2(9 + \sqrt{9}) + 8 \times 2(9 + \sqrt{9}) = 531$ qubits. Based on Grover's algorithm, an exhaustive search is performed on a 123-bit key. If the function outputs a period s using Simon's algorithm, then the guessed key is correct. The required time complexity is $2^{\frac{n_k}{2}} = 2^{61.5}$.

5 Conslusion and future work

Ito et al. studied the 4-round Feistel structure quantum distinguisher under the Q2 model, but did not consider the algorithm's internal structure .This paper investigates quantum key-recovery attacks against Camellia under the Q2 model. Initially, based on the round function and key scheduling features of Camellia, a 5-round qCCA distinguisher is proposed. Subsequently, utilizing this distinguisher, we present a 9-round quantum key -recovery attack based on the Grover-meets-Simon algorithm, achieving a time complexity of $2^{61.5}$. Compared to previous quantum analyses of Camellia, our attack performs better and is the first to conduct a quantum chosen-ciphertext key-recovery attack on Camellia.

The ability of symmetric cryptographic algorithms against quantum attacks is garnering increasing attention. In addition to performing quantum security analyses on existing symmetric cryptographic algorithms, scholars are also considering quantum security as an essential criterion in proposing new symmetric cryptographic algorithms. Future research will concentrate on the construction of quantum distinguishers with an increased number of rounds under the quantum computing model. Additionally, the time complexity of quantum key-recovery attacks may be reduced through the exploration of quantum algorithms. Furthermore, the efficacy of symmetric cryptographic structural schemes against quantum attacks will be further investigated.

Acknowledgments. This work was supported by the Fund of Yunnan Key Laboratory of Blockchain Application Technology (Grant No.202305AG340008) and the Beijing Natural Science Foundation (Grant No.4234084).

Data availability. The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The author declares that there are no conflicts of interest.

References

[1] Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review **41**(2), 303–332 (1999)

[2] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 212–219 (1996)

[3] Simon, D.R.: On the power of quantum computation. SIAM journal on computing **26**(5), 1474–1483 (1997) https://doi.org/10.1137/S0097539796298637

[4] Kuwakado, H., Morii, M.: Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: 2010 IEEE International Symposium on Information Theory, pp. 2682–2685 (2010). https://doi.org/10.1109/ISIT.2010.5513654 . IEEE

[5] Zhandry, M.: How to construct quantum random functions. In: 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science, pp. 679–687 (2012). https://doi.org/10.1109/FOCS.2012.37 . IEEE

[6] Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. arXiv preprint arXiv:1510.05836 (2015) https://doi.org/10.48550/arXiv.1510.05836

[7] Kuwakado, H., Morii, M.: Security on the quantum-type even-mansour cipher. In: 2012 International Symposium on Information Theory and Its Applications, pp. 312–316 (2012). IEEE

[8] Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: New attacks on feistel structures with improved memory complexities. In: Annual Cryptology Conference, pp. 433–454 (2015). https://doi.org/10.1007/978-3-662-47989-6_21 . Springer

[9] Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36, pp. 207–237 (2016). https://doi.org/10.1007/978-3-662-53008-5_8 . Springer

[10] Leander, G., May, A.: Grover meets simon–quantumly attacking the fx-construction. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23, pp. 161–178 (2017). https://doi.org/10.1007/978-3-319-70697-9_6 . Springer

[11] Hosoyamada, A., Aoki, K.: On quantum related-key attacks on iterated even-mansour ciphers. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **102**(1), 27–34 (2019) https://doi.org/10.1587/transfun.E102.A.27

[12] Dong, X., Wang, X.: Quantum key-recovery attack on feistel structures. Science China Information Sciences **61**(10), 102501 (2018) https://doi.org/10.1007/s11432-017-9468-y

[13] Ito, G., Hosoyamada, A., Matsumoto, R., Sasaki, Y., Iwata, T.: Quantum chosen-ciphertext attacks against feistel ciphers. In: Topics in Cryptology–CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings, pp. 391–411 (2019). https://doi.org/10.1007/978-3-030-12612-4_20 . Springer

[14] Dong, X., Li, Z., Wang, X.: Quantum cryptanalysis on some generalized feistel schemes. Science China Information Sciences **62**(2), 22501 (2019) https://doi.org/10.1007/s11432-017-9436-7

[15] Ito, G., Iwata, T.: Quantum distinguishing attacks against type-1 generalized feistel ciphers (2019). Consulté le: 25 avril 2023

[16] Ni, B., Dong, X.: Improved quantum attack on type-1 generalized feistel schemes and its application to cast-256. Cryptology ePrint Archive (2019)

[17] Cid, C., Hosoyamada, A., Liu, Y., Sim, S.M.: Quantum cryptanalysis on contracting feistel structures and observation on related-key settings. In: Progress in Cryptology–INDOCRYPT 2020: 21st International Conference on Cryptology in India, Bangalore, India, December 13–16, 2020, Proceedings 21, pp. 373–394 (2020). https://doi.org/10.1007/978-3-030-65277-7_17 . Springer

[18] Hodžić, S., Knudsen, L.R.: A quantum distinguisher for 7/8-round sms4 block cipher. Quantum Information Processing **19**(11), 411 (2020) https://doi.org/10.1007/s11128-020-02929-6

[19] Li, Y., Lin, H., Liang, M., Sun, Y.: A new quantum cryptanalysis method on block cipher camellia. IET Information Security **15**(6), 487–495 (2021) https://doi.org/10.1049/ise2.12037

[20] Cui, J., Guo, J., Ding, S.: Applications of simon's algorithm in quantum attacks on feistel variants. Quantum Information Processing **20**, 1–50 (2021) https://doi.org/10.1007/s11128-021-03027-x

[21] Canale, F., Leander, G., Stennes, L.: Simon's algorithm and symmetric crypto: Generalizations and automatized applications. In: Annual International Cryptology Conference, pp. 779–808 (2022). Springer

[22] Xu, Y., Du, X., Jia, M., Wang, X., Zou, J.: Quantum attacks on generalized feistel networks based on the strong–weak separability. Quantum Information Processing **22**(10), 375 (2023) https://doi.org/10.1007/s11128-023-04135-6

[23] Sun, H.-W., Cai, B.-B., Qin, S.-J., Wen, Q.-Y., Gao, F.: Quantum attacks on type-1 generalized feistel schemes. Advanced Quantum Technologies **6**(10), 2300155 (2023) https://doi.org/10.1002/qute.202300155

[24] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-bit block cipher suitable for multiple platforms—design and-analysis. In: Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000 Waterloo, Ontario, Canada, August 14–15, 2000 Proceedings 7, pp. 39–56 (2001). https://doi.org/10.1007/3-540-44983-3_4 . Springer

[25] Hosoyamada, A., Sasaki, Y.: Quantum demiric-selçuk meet-in-the-middle attacks: applications to 6-round generic feistel constructions. In: Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings 11, pp. 386–403 (2018). https://doi.org/10.1007/978-3-319-98113-0_21 . Springer