Classical and Quantum Generic Attacks on 6-round Feistel Schemes

Maya Chartouny[1,2], Benoit Cogliati[1], and Jacques Patarin[1,2]

[1] Thales DIS, Meudon, France
{maya.saab-chartouni, benoit-michel.cogliati, jacques.patarin}@thalesgroup.com
[2] Université Paris-Saclay, UVSQ, CNRS, Laboratoire de mathématiques de Versailles, 78000, Versailles, France

Abstract. In this paper, we describe new quantum generic attacks on 6 rounds balanced Feistel networks with internal functions or internal permutations. In order to obtain our new quantum attacks, we revisit a result of Childs and Eisenberg that extends Ambainis' collision finding algorithm to the subset finding problem. In more details, we continue their work by carefully analyzing the time complexity of their algorithm. We also use four points structures attacks instead of two points structures attacks that leads to a complexity of $\mathcal{O}(2^{8n/5})$ instead of $\mathcal{O}(2^{2n})$. Moreover, we have also found a classical (i.e. non quantum) improved attack on 6 rounds with internal permutations. The complexity here will be in $\mathcal{O}(2^{2n})$ instead of $\mathcal{O}(2^{3n})$ previously known.

Keywords: Feistel ciphers · Pseudo-random permutation · Quantum cryptanalysis · Luby–Rackoff block cipher · Subset finding problem

1 Introduction

In this paper, we will study the classical and the quantum security of random balanced Feistel ciphers, by showing the best known quantum attacks on these schemes. A random Feistel cipher also known as Luby–Rackoff block cipher is a symmetric structure used in the construction of block ciphers. The benefit of the Feistel network is that the same structure can be used for encryption and decryption, and both consist of running a function called a "round function" a fixed number of times. Each round of a (balanced) Feistel scheme takes as input $[L, R]$ that stands for *Left* and *Right* and outputs $[R, L \oplus f(R)]$ where the internal round function f is a secret function from n bits to n bits. When the internal round functions $f_1, \ldots, f_r$ are random functions, or random permutations these schemes are called "random Feistel ciphers" also known as Luby-Rackoff block ciphers. A huge number of papers have been written on these constructions, and the most studied way to build pseudo-random permutation from random function (or random permutations) is the $r-$round Feistel construction.

State of the art. Classical attacks on Feistel's schemes have been widely studied in the literature [Pat01,NVP13,Pat08,Wag99]. Since 2010, the quantum security of these schemes is studied. We will present results that have been published in various papers on these quantum attacks, and we will also present new quantum attacks that we have found.

At least three very different quantum technique can be used to attack the Feistel network. For instance, Simon's quantum algorithm was used for 3 rounds (quantum chosen plaintext attack of [KM10]) and for 4 rounds (quantum chosen plaintext and cipher text attack [IHM+19]). Thanks to Simon's quantum algorithm, which consists in searching for a period, the complexity of these quantum attacks is polynomial (unlike the exponential complexity of non-quantum attacks). Moreover, for 5 rounds (quantum chosen plaintext and quantum plaintext and cipher-text attacks of [CPT22]) Zhandry collision algorithm [Zha13] was used. This algorithm is based on a result of Ambainis [Amb04]. The quantum attacks obtained are still exponential, but with a smaller exponent than for non quantum attacks.

	KPA	CPA	CCA	QCPA	QCCA
Ψ^1	1	1	1	1	1
Ψ^2	$2^{n/2}$	2	2	2	2
Ψ^3	$2^{n/2}$	$2^{n/2}$	3	n	3
Ψ^4	2^n	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$	n
Ψ^5	$2^{3n/2}$	2^n	2^n	$2^{2n/3}$	$2^{2n/3}$
Ψ^6	2^{2n}	2^{2n}	2^{2n}	$2^{8n/5}$(New)	$2^{8n/5}$(New)

Fig. 1: Number of computations to distinguish Feistel schemes (with 1, 2, 3, 4, 5 and 6 rounds) with random internal functions from random permutations.

	KPA	CPA	CCA	QCPA	QCCA
Ψ^1	1	1	1	1	1
Ψ^2	$2^{n/2}$	2	2	2	2
Ψ^3	$2^n(+)$	$2^{n/2}$	3	n	3
Ψ^4	2^n	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$	n
Ψ^5	$2^{3n/2}$	2^n	2^n	$2^{2n/3}$	$2^{2n/3}$
Ψ^6	2^{2n}(New)	2^{2n}(New)	2^{2n}(New)	$2^{8n/5}$(New)	$2^{8n/5}$(New)

Fig. 2: Number of computations to distinguish Feistel schemes (with 1, 2, 3, 4, 5 and 6 rounds) with random internal permutations from random permutations (best known attacks). We write (+) when the complexity is worse than for Feistel networks with internal functions.

Our contribution. In this article, we will present new results. In fact,

- We showed that the classical (i.e. non-quantum) complexity over 6 rounds Feistel schemes with internal permutation is in $\mathcal{O}(2^{2n})$ and not $\mathcal{O}(2^{3n})$ as mentioned in [TP09].
- We developed the existing attack on 6-round Feistel networks with internal functions, providing detailed insights.
- To validate the classical attack on both 6-round Feistel schemes with internal functions and 6-round Feistel schemes with internal permutations, we performed computer simulations to provide empirical evidence of its effectiveness.
- We made a detailed time complexity analysis to Childs and Eisenberg's algorithm.
- By using Childs and Eisenberg's quantum algorithm, we improved attack methods. This allowed us to perform a quantum attack on 6-round Feistel schemes with internal functions as well as internal permutations with a complexity of $\mathcal{O}(2^{8n/5})$ instead of the previously known classical complexity of $\mathcal{O}(2^{2n})$.

2 Preliminaries

Notations. For $n \in \mathbb{N}$, $n \geq 1$, $\{0,1\}^n$ denotes the set of binary strings of length n. Let $\mathcal{F}_{m,n}$ be the set of all functions from $\{0,1\}^m$ to $\{0,1\}^n$. When $m = n$, the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$ will be denoted by $\mathcal{F}_n$. If L and R are elements of $\{0,1\}^n$, we will denote by $[L,R]$ the element of $\{0,1\}^{2n}$ which is the concatenation of L and R. The XOR operator is denoted by $\oplus$ (bit wise addition modulo 2).

Security notions. An adversary's advantage is a measure of how successfully he can attack a cryptographic algorithm, by distinguishing it from an idealized version of that type of algorithm.

Consider an oracle, denoted as F, which represents the function being studied, and another oracle, denoted as G, which simulates an idealized function of the same type. The adversary, denoted as A, is a probabilistic algorithm that takes either F or G as input and produces an output of either 1 or 0. The primary task of adversary A is to differentiate between F and G by making queries to the oracle provided. We can compute the advantage by using this formula:

$$Adv(A) = \left|\Pr[A(F) = 1] - \Pr[A(G) = 1]\right|.$$

A useful lemma. In this paragraph, we state a Chernoff-type bound for binary random variables that are moderately dependent, and do not necessarily share a common bound. This result can be seen as a slight variant of [CLL+14, Lemma 5].

Lemma 1. *Let* $\mathbf{A} = (A_i)_{1 \le i \le q}$ *be a sequence of random variables taking value in* $\{0,1\}$*. Assume that, for any* $1 \le i \le q$*, there exists* $0 \le \varepsilon_i < 1$ *such that, for any binary sequence* $(a_1, \ldots, a_{i-1})$*, one has*

$$\Pr[A_i = 1 \,|\, (A_1, \ldots, A_{i-1}) = (a_1, \ldots, a_{i-1})] \le \varepsilon_i.$$

Then, for any $\delta > 0$*, one has*

$$\Pr\left[\sum_{i=1}^{q} A_i \ge (1+\delta)m\right] \le e^{-\frac{\delta^2}{2+\delta}m},$$

where $m = \sum_{i=1}^{q} \varepsilon_i$.

The proof of this Lemma can straightforwardly be derived from the one of [CLL$^+$14, Lemma 5]. For the sake of completeness, we state it in Appendix A.

3 Feistel schemes

In this section, we recall the definition of a classical (aka balanced) Feistel scheme.

First round Feistel scheme. Let $f \in \mathcal{F}_n$. The first round balanced Feistel scheme associated with f, denoted by $\Psi(f)$, is the function in $\mathcal{F}_{2n}$ defined by:

$$\forall (L,R) \ \in \{0,1\}^{2n},\ \Psi(f)\Big([L,R]\Big) = [S,T] \iff \begin{cases} S = R, \\ T = L \oplus f(R). \end{cases}$$

For any function f, $\Psi(f)$ is a permutation of $\{0,1\}^{2n}$.
The figure of the Feistel scheme for the first round is given in Figure 3.

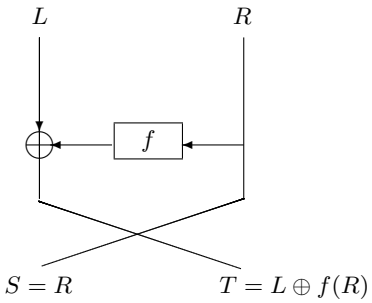

Fig. 3: First round of Feistel scheme.

$r-$round Feistel scheme. Let $f_1, f_2, \ldots, f_r$ be r functions in $\mathcal{F}_n$. The $r-$round balanced Fesitel network associated with $f_1, \ldots, f_r$, denoted by $\Psi^r(f_1, \ldots, f_r)$, is the function in $\mathcal{F}_{2n}$ defined by:

$$\Psi^r(f_1, \ldots, f_r) = \Psi^r(f_r) \circ \cdots \circ \Psi^1(f_1).$$

6−round Feistel scheme. We provide now detailed equations of the Feistel network for the initial six rounds. Let $M = L||R$, then

$$\text{1 round: } \begin{cases} R, \\ L \oplus f_1(R) = X. \end{cases} \qquad \text{4 rounds: } \begin{cases} Z, \\ Y \oplus f_4(Z) = U. \end{cases}$$

$$\text{2 rounds: } \begin{cases} X, \\ R \oplus f_2(X) = Y. \end{cases} \qquad \text{5 rounds: } \begin{cases} U, \\ Z \oplus f_5(U) = S. \end{cases}$$

$$\text{3 rounds: } \begin{cases} Y, \\ X \oplus f_3(Y) = Z. \end{cases} \qquad \text{6 rounds: } \begin{cases} S, \\ U \oplus f_6(S) = T. \end{cases}$$

Alternative notation. We will also introduce variables X^i such that for each round k we have

$$X^k = X^{k-2} \oplus f_k(X^{k-1}). \tag{1}$$

Hence, $X^1 = L$, $X^0 = R$, $X^1 = X$, $X^2 = Y$, $X^3 = Z$, $X^4 = U$, $X^5 = S$.

Theorems. Let $[L_i, R_i]$ be known inputs with $1 \leq i \leq m$ and let $[S_i, T_i] = \Psi^6(f_1, \ldots, f_6)([L_i, R_i])$. We assume that $[L_i, R_i]$ values are pairwise distinct, i.e., we cannot have $L_i = L_j$ and $R_i = R_j$ with $i \neq j$.

Theorem 1. *If we have internal functions, or internal permutations, and if $i \neq j$ we cannot have $X_i^{k-1} = X_j^{k-1}$ and $X_i^k = X_j^k$.*

Proof. This theorem can be proved by induction on k using Equation(1).

Theorem 2. *If we have internal permutations, and if $i \neq j$ we cannot have*

$$\begin{cases} X_i^{k-2} = X_j^{k-2}, & (*) \\ X_i^k = X_j^k. & (**) \end{cases}$$

Proof. By definition, we have that $(**)$ is equivalent to $X_i^{k-2} \oplus f_k(X_i^{k-1}) = X_j^{k-2} \oplus f_k(X_j^{k-1})$ and so $f_k(X_i^{k-1}) = f_k(X_j^{k-1})$ (thanks to Equation $(*)$). Since f_k is a permutation, f_k is injective, we get $X_i^{k-1} = X_j^{k-1}$. Hence, we have:

$$\begin{cases} X_i^{k-2} = X_j^{k-2}, \\ X_i^{k-1} = X_j^{k-1}. \end{cases}$$

From Theorem(1) this is not possible.

4 Classical attacks on 6 rounds Feistel schemes

In this section, we described a four points classical attack on 6 rounds Feistel schemes with internal permutations. We identified an attack as detailed below with a complexity of $\mathcal{O}(2^{2n})$. Moreover, we made a detailed analysis of a four points classical attack on 6−round Feistel schemes with internal functions presented in [Pat01].

The Attack. Let $[L_i, R_i] \in \{0,1\}^{2n}$. We generate m messages $[L_i, R_i]$ with $1 \leq i \leq m$ and $m = \mathcal{O}(2^{2n})$. We then compute $[S_i, T_i] = \Psi^6(f_1, \ldots, f_6)([L_i, R_i])$. We look if among these values we can find four pairwise distinct indices denoted by $1, 2, 3, 4$ such that these eight equations are satisfied:

$$(\mathcal{S}) \begin{cases} R_1 = R_3, \\ R_2 = R_4, \\ S_1 = S_2, \\ S_3 = S_4, \\ L_1 \oplus L_4 = L_2 \oplus L_3, \\ L_1 \oplus S_1 = L_3 \oplus S_3, & (\iff L_2 \oplus S_2 = L_4 \oplus S_4) \\ R_1 \oplus T_1 = R_2 \oplus T_2, \\ R_3 \oplus T_3 = R_4 \oplus T_4. & (\iff T_1 \oplus T_4 = T_2 \oplus T_3) \end{cases}$$

We also have $R_1 \neq R_2, S_1 \neq S_3$ and $T_1 \neq T_2$. We give a visual representation of these 8 equations in Figure (4).

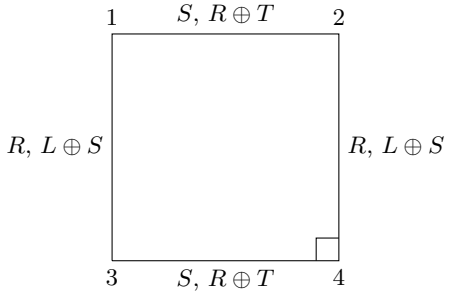

Fig. 4: Representation of the 8 equations of the System $(\mathcal{S})$. The small square in the right-hand corner below represents $L_1 \oplus L_2 \oplus L_3 \oplus L_4 = 0$.

For a 6−round Feistel scheme with internal permutations, we will show that we have about two times more such collision than for truly random permutations. Similarly, for a 6-round Feistel scheme with internal functions, we show that we have 12 times more such collisions than for a truly random permutation. This will give us an attack of $\mathcal{O}(2^{2n})$ in order to distinguish a random permutation from a 6−round Feistel scheme with internal functions or permutations.

Remark 1. This result was already known for internal functions [Pat01]. In fact, when we have internal functions, different kinds of attacks are known with a complexity of $\mathcal{O}(2^{2n})$: 'four points' attacks as described in this paper, or 'two points' attacks. However, this result is new for internal permutations. Note also that for internal permutations four points attacks are in $\mathcal{O}(2^{2n})$ but 'two points' attacks are in $\mathcal{O}(2^{3n})$.

Case of $6-$*round Feistel with internal permutations.*

Theorem 3. *Let us assume that*

$$(1)\ \begin{cases} R_1 = R_3, \\ R_2 = R_4, \\ S_1 = S_2, \\ L_1 \oplus L_4 = L_2 \oplus L_3. \end{cases}$$

Let us also assume that

$$(2)\ \begin{cases} X_1 = X_4, \\ Y_1 = Y_2, \\ Z_1 = Z_3, \\ U_1 = U_4. \end{cases}$$

Then, we will necessary have for a $6-$*round Feistel schemes with internal functions (and hence also for internal permutations):*

$$(3)\ \begin{cases} S_3 = S_4, \\ L_1 \oplus S_1 = L_3 \oplus S_3, \\ R_1 \oplus T_1 = R_2 \oplus T_2, \\ R_3 \oplus T_3 = R_4 \oplus T_4. \end{cases}$$

Note that the equations of System (2) *are compatible with Theorem (1) and Theorem (2) of Section (3).*

Proof. Suppose we have System (1) and System (2), let us show that System (3) is verified.

From $X_1 = X_4$ we get $X_2 = X_3$. In fact, $X_1 = X_4$ is equivalent to $L_1 \oplus f_1(R_1) = L_4 \oplus f_1(R_4)$. Since $R_1 = R_3$, $R_4 = R_2$ and $L_1 \oplus L_4 = L_2 \oplus L_3$, we get, $L_2 \oplus f_1(R_2) = L_3 \oplus f_1(R_3)$, i.e., $X_2 = X_3$.

Similarly, from $Y_1 = Y_2$ we get $Y_3 = Y_4$. Indeed, $Y_1 = Y_2$ means $R_1 \oplus f_2(X_1) = R_2 \oplus f_2(X_2)$. Since $R_1 = R_3$, $R_2 = R_4$, $X_1 = X_4$ and $X_2 = X_3$, we get $R_3 \oplus f_2(X_3) = R_4 \oplus f_2(X_4)$, i.e., $Y_3 = Y_4$.

Also, from $Z_1 = Z_3$ we get $Z_2 = Z_4$. In fact, $Z_1 = Z_3$ means $X_1 \oplus f_3(Y_1) = X_3 \oplus f_3(Y_3)$. Since $X_1 = X_4, X_3 = X_2, Y_1 = Y_2$, and $Y_3 = Y_4$, we obtain $X_2 \oplus f_3(Y_2) = X_4 \oplus f_3(Y_4)$, i.e., $Z_2 = Z_4$.

Moreover, from $U_1 = U_4$ we obtain $U_2 = U_3$. In fact, $U_1 = U_4$ is equivalent to $Y_1 \oplus f_4(Z_1) = Y_4 \oplus f_4(Z_4)$. Since $Z_1 = Z_3$, $Z_4 = Z_2$, $Y_1 = Y_2$ and $Y_4 = Y_3$, we get $Y_2 \oplus f_4(Z_2) = Y_3 \oplus f_4(Z_3)$, i.e., $U_2 = U_3$.

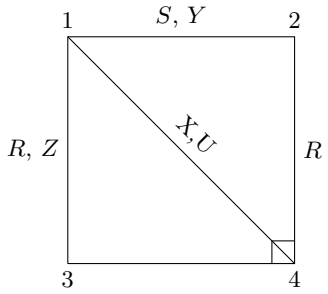

Fig. 5: Representation of our 8 starting equations. The small square in the right-hand corner below represents $L_1 \oplus L_2 \oplus L_3 \oplus L_4 = 0$.

Similarly, from $S_1 = S_2$ we get $S_3 = S_4$. Indeed, $S_1 = S_2$ means $Z_1 \oplus f_5(U_1) = Z_2 \oplus f_5(U_2)$. Since $Z_1 = Z_3$, $Z_2 = Z_4$, $U_1 = U_4$ and $U_2 = U_3$ we get $Z_3 \oplus f_5(U_3) = Z_4 \oplus f_5(U_4)$ which means $S_3 = S_4$.

Now from $S_1 = S_2$ we get:

$$U_1 \oplus U_2 = T_1 \oplus T_2, \tag{2}$$

because $T_1 = U_1 \oplus f_6(S_1)$ and $T_2 = U_2 \oplus f_6(S_2)$.

Similarly from $S_3 = S_4$ we get $U_3 \oplus U_4 = T_3 \oplus T_4$, i.e.,

$$T_3 \oplus T_4 = U_1 \oplus U_2, \tag{3}$$

because $U_4 = U_1$ and $U_3 = U_2$.

Similarly from $U_1 = U_4$ we get $S_1 \oplus S_4 = Z_1 \oplus Z_4$, i.e.,

$$S_1 \oplus S_3 = Z_1 \oplus Z_2, \tag{4}$$

since $S_4 = S_3$ and $Z_4 = Z_2$.

Similarly from $Z_1 = Z_3$ we get $U_1 \oplus U_3 = Y_1 \oplus Y_3$, i.e.,

$$U_1 \oplus U_2 = Y_1 \oplus Y_3, \tag{5}$$

since $U_3 = U_2$.

Similarly from $Y_1 = Y_2$ we get:

$$Z_1 \oplus Z_2 = X_1 \oplus X_2. \tag{6}$$

Similarly from $X_1 = X_4$ we get $Y_1 \oplus Y_4 = R_1 \oplus R_4$, i.e.,

$$Y_1 \oplus Y_3 = R_1 \oplus R_2, \tag{7}$$

because $Y_4 = Y_3$ and $R_4 = R_2$.

Similarly from $R_1 = R_3$ we get $X_1 \oplus X_3 = L_1 \oplus L_3$, i.e.,

$$X_1 \oplus X_2 = L_1 \oplus L_3, \tag{8}$$

because $X_3 = X_2$.

From Equations (2), (5) and (7) we obtain:

$$T_1 \oplus T_2 = R_1 \oplus R_2 \quad (= Y_1 \oplus Y_3).$$

From Equations (3),(5) and (7) we obtain $R_1 \oplus R_2 = T_3 \oplus T_4$. Since $R_1 = R_3$ and $R_2 = R_4$ we get:

$$R_3 \oplus R_4 = T_3 \oplus T_4.$$

Also from Equations (4),(6) and (8) we obtain:

$$S_1 \oplus S_3 = L_1 \oplus L_3 \quad (= Z_1 \oplus Z_2 = X_1 \oplus X_2).$$

Therefore, we have proved that the System (3) is verified.

From Theorem (3), we see that we will have at least about 2 times more solutions of the System ($\mathcal{S}$) when we have a 6-round Feistel scheme with internal permutations than when we have a random permutation. In fact, for a 6−round Feistel scheme, the System ($\mathcal{S}$) can appear randomly or from the equations of Figure (5). In both cases we have 8 equations on 4 indices. This gives an attack in $\mathcal{O}\left(\frac{q^4}{2^{8n}}\right)$, i.e., when $q \approx \mathcal{O}(2^{2n})$.

Case of 6−*round Feistel with internal functions.* This attack also works on 6−round Feistel with internal functions. In fact, instead of about 2 more times solutions we will have here about 12 more times solutions. These solutions come from random solution of the System ($\mathcal{S}$) plus these 11 families listed below. In fact, in Theorem (3), we can replace the System (2) by any system of these 11 families and still prove in a similar way that System (3) is verified. This still gives an attack in $\mathcal{O}(2^{2n})$.

$$\text{Family 1: } \begin{cases} X_1 = X_2, \\ Y_1 = Y_3, \\ Z_1 = Z_2, \\ U_1 = U_3. \end{cases} \qquad \text{Family 2: } \begin{cases} X_1 = X_2, \\ Y_1 = Y_3, \\ Z_1 = Z_2, \\ U_1 = U_4. \end{cases}$$

$$\text{Family 3: } \begin{cases} X_1 = X_2, \\ Y_1 = Y_3, \\ Z_1 = Z_4, \\ U_1 = U_3. \end{cases} \qquad \text{Family 4: } \begin{cases} X_1 = X_2, \\ Y_1 = Y_4, \\ Z_1 = Z_2, \\ U_1 = U_3. \end{cases}$$

$$\text{Family 5: } \begin{cases} X_1 = X_2, \\ Y_1 = Y_4, \\ Z_1 = Z_2, \\ U_1 = U_4. \end{cases} \qquad \text{Family 6: } \begin{cases} X_1 = X_2, \\ Y_1 = Y_4, \\ Z_1 = Z_3, \\ U_1 = U_4. \end{cases}$$

$$\text{Family 7:} \begin{cases} X_1 = X_4, \\ Y_1 = Y_2, \\ Z_1 = Z_3, \\ U_1 = U_4. \end{cases} \qquad \text{Family 8:} \begin{cases} X_1 = X_4, \\ Y_1 = Y_2, \\ Z_1 = Z_4, \\ U_1 = U_3. \end{cases}$$

$$\text{Family 9:} \begin{cases} X_1 = X_4, \\ Y_1 = Y_3, \\ Z_1 = Z_2, \\ U_1 = U_3. \end{cases} \qquad \text{Family 10:} \begin{cases} X_1 = X_4, \\ Y_1 = Y_3, \\ Z_1 = Z_2, \\ U_1 = U_4. \end{cases}$$

$$\text{Family 11:} \begin{cases} X_1 = X_4, \\ Y_1 = Y_3, \\ Z_1 = Z_4, \\ U_1 = U_3. \end{cases}$$

Note that these families are compatible with Theorem (1) of Section (3). Note also that Family (7) is the same as System (2).

Remark 2. Family (7) is the only family that is compatible with the case of 6−round Feistel schemes with internal permutations because of Theorem (3).

Remark 3. We found an even number of solutions since if we exchange indices 1 and 3 and indices 2 and 4, we also obtain a solution of $(\mathcal{S})$. Alternatively, it would have been possible to impose that index 1 is strictly less than index 3.

Remark 4. Insted of the 8 equations of $(\mathcal{S})$ we may want to compute solutions such that

$$(\mathcal{S}') \begin{cases} R_1 = R_2, \\ R_3 = R_4, \\ S_1 = S_3, \\ S_2 = S_4, \\ T_1 \oplus T_4 = T_2 \oplus T_3, \\ T_1 \oplus R_1 = T_3 \oplus R_3, \\ S_1 \oplus L_1 = S_2 \oplus L_2, \\ L_1 \oplus L_2 \oplus L_3 \oplus L_4 = 0. \end{cases}$$

We use here the fact that the inverse of a Feistel scheme is another Feistel scheme when we permute the left and right part, i.e., there is a symmetry R/S and L/T. However $(\mathcal{S}')$ is just the same as $(\mathcal{S})$ when we permute the indices 1 and 2.

5 Complexity of classical attacks on 6 rounds Feistel schemes

In this section, we will show that the complexity of the attack for 6-round Feistel schemes with random functions or with random permutations is in $\mathcal{O}(2^{2n})$.

First, we select indices 1 and 2 such that:

$$\begin{cases} S_2 = S_1, \\ R_2 \oplus T_2 = R_1 \oplus T_1. \end{cases}$$

We expect to find $\frac{m^2}{2^{2n}}$ possibilities with $m = 2^{2n}$. Indeed, it's possible to do this in $\mathcal{O}(m)$ computations and $\mathcal{O}(m)$ memory by storing all the m values $(S_i, R_i \oplus T_i)$ in a hash table and looking for collisions.

Moreover, we find index 3 such that:

$$\begin{cases} R_3 = R_1, \\ L_3 \oplus S_3 = L_1 \oplus S_1. \end{cases}$$

We expect to find $\frac{m}{2^{2n}}$ possibilities with $m = 2^{2n}$. In fact, it is possible to do this in $\mathcal{O}(1)$ computations and $\mathcal{O}(m)$ memory by storing all the m values $(R_i, L_i \oplus S_i)$ in a hash table and looking for $(R_1, L_1 \oplus S_1)$.

Finally, we look for index 4 such that:

$$\begin{cases} R_4 = R_2, \\ L_4 = L_1 \oplus L_2 \oplus L_3. \end{cases}$$

It is possible to do this in $\mathcal{O}(1)$ and in this case we don't need a hash table.

At the end we have at most m choices of pairwise distinct indices $(1, 2, 3, 4)$. Among these we keep those that give $R_1 \neq R_2$ and $L_1 \neq L_3$. Therefore, the total complexity is in $\mathcal{O}(2^{2n})$ computations and $\mathcal{O}(2^{2n})$ memory.

6 Computer Simulations

Let $\mathcal{N}_0$ be the number of solutions of the System $(\mathcal{S})$ for random permutations on $2n$ bits and σ_0 be the corresponding standard deviation. Let $\mathcal{N}_1$ be the number of solutions of the System $(\mathcal{S})$ for random 6−round Feistel schemes with internal permutations on $2n$ bits and σ_1 be the corresponding standard deviation. Let $\mathcal{N}_2$ be the number of solutions of the System $(\mathcal{S})$ for random 6−round Feistel schemes with internal functions on $2n$ bits and σ_2 be the corresponding standard deviation. Let Adv_1 be the advantage to distinguish a random permutation from a 6−round Feistel scheme with internal permutations. Let Adv_2 be the advantage to distinguish a random permutation from a 6−round Feistel scheme with internal functions. We did computer simulations for 10 000 permutations.

Simulations for n=7. We have obtained the following results:
$\mathcal{N}_0 \approx 0.486$ and $\sigma_0 \approx 0.9853$,
$\mathcal{N}_1 \approx 0.999$ and $\sigma_1 \approx 1.4116$,
$\mathcal{N}_2 \approx 5.9566$ and $\sigma_2 \approx 3.6751$.
As expected, $\mathcal{N}_1 \approx 2\mathcal{N}_0$ and $\mathcal{N}_2 \approx 12\mathcal{N}_0$. More precisely, we found for 10 000

trials the following:

Number of solutions	Random permutations	Feistel with internal permutations	Feistel with internal functions
0	7 839	6 081	580
2	1 915	2 994	1 634
4	225	787	2 192
6	19	126	2 167
8	2	11	1 539
10	0	1	952
12	0	0	508
14	0	0	235
16	0	0	119
18	0	0	47
20	0	0	19
22	0	0	6
24	0	0	2

Moreover, we have $\mathrm{Adv}_1 = 0.1758$ and $\mathrm{Adv}_2 = 0.754$.

Simulations for n=8. We have obtained the following results:
$\mathcal{N}_0 \approx 0.5062$ and $\sigma_0 \approx 1.0024$,
$\mathcal{N}_1 \approx 1.0126$ and $\sigma_1 \approx 1.4259$,
$\mathcal{N}_2 \approx 6.0098$ and $\sigma_2 \approx 3.5957$.
As expected $\mathcal{N}_1 \approx 2\mathcal{N}_0$ and $\mathcal{N}_2 \approx 12\mathcal{N}_0$. More precisely, we found for 10 000 trials:

Number of solutions	Random permutations	Feistel with internal permutations	Feistel with internal functions
0	7 756	6 023	530
2	1 980	3 069	1 570
4	242	754	2 198
6	21	133	2 154
8	1	19	1 670
10	0	1	980
12	0	1	510
14	0	0	224
16	0	0	99
18	0	0	40
20	0	0	17
22	0	0	5
24	0	0	3

Moreover, we have $\mathrm{Adv}_1 = 0.1733$ and $\mathrm{Adv}_2 = 0.7636$

Remark 5. We don't use here the property that Ψ^6 are even permutations. Hence, our advantages are also the advantages for distinguishing Ψ^6 from random even permutations (or from random odd permutations). Testing whether a permutation is even requires knowing all the inputs/outputs (or all but one), whereas our advantage attacks change a little bit if we don't know a small number of inputs/outputs.

7 Quantum attack

In this section, we start by recalling in Section 7.1 Ambainis' quantum collision finding algorithm given in [Amb04]. We then recall the quantum algorithms for subset finding in Section 7.2 and carefully give the time complexity of the algorithm. We describe the data structures from Ambainis' algorithm in Section 7.3. Finally, we apply it to the problem of distinguishing 6-round Feistel schemes from a random permutation in Section 7.4.

7.1 Recalling Ambainis' algorithm

In this section, we explain how to solve the k-distinctness problem for a fixed value of k. Specifically, given N values $x_1, \ldots, x_N$ that belong to a set $\mathcal{X}$ of size M, are there k distinct indices $1 \leq i_1, \ldots, i_k \leq N$ such that $x_{i_1} = \cdots = x_{i_k}$? In [Amb04], Ambainis proves the following result.

Theorem 4 ([Amb04]). *Let $r \geq k$, $r = o(N)$. There is a quantum algorithm that solves element k-distinctness in time $\widetilde{\mathcal{O}}\left(\max\left(\frac{N^{k/2}}{r^{(k-1)/2}}, r\right)\right)$, with $\mathcal{O}\left(\max\left(\frac{N^{k/2}}{r^{(k-1)/2}}, r\right)\right)$ queries, using $\widetilde{\mathcal{O}}(r)$ qubits of memory.*

We will give a rough overview of the algorithm and the necessary conditions that are required to prove Theorem 4, and in particular with respect to the time complexity of the algorithm.

Remark 6. We will only describe the quantum algorithm that solves the k-distinctness problem in the specific case where there only exists a single k-collision. Ambainis gives a generic method to derive an algorithm that solves the general problem using the restricted one as a subroutine, and without changing the complexity.

From now on, we assume that our goal is to find the unique tuple of indices that forms a k-collision. From a high-level, the algorithm considers the set $\mathcal{T}$ of tuples of the form $(\mathbf{S}, x_{\mathbf{S}}, y)$ where:

- $\mathbf{S} = \{y_1, \ldots, y_r\}$ is a subset of $\{1, \ldots, N\}$ of size r,
- $x_{\mathbf{S}} = (x_{y_1}, \ldots, x_{y_r}) \in \mathcal{X}^r$,
- $y \in \{1, \ldots, N\} \setminus \mathbf{S}$.

It then works as a quantum walk over the set of all these tuple by taking the following steps:

1. Build a uniform superposition of all tuples $(\mathbf{S}, x_{\mathbf{S}}, y)$.
2. Repeat the following procedure $t_1 = \mathcal{O}\left((N/r)^{k/2}\right)$ times:
   (a) If there exists k distinct indices $i_1, \ldots, i_k$ in $\mathbf{S}$ such that $x_{i_1} = \cdots = x_{i_k}$, apply the conditional phase flip.
   (b) Perform $t_2 = \mathcal{O}(\sqrt{r})$ times the following quantum walk step:
      i. Apply a diffusion operator that will create a superposition of all y values such that $y \in \{1, \ldots, N\} \setminus \mathbf{S}$[3].
      ii. Add y to $\mathbf{S}$ and modify the corresponding $x_{\mathbf{S}}$ vector accordingly.
      iii. In order to map the current vector back to $\mathcal{T}$, we need to remove an element from $\mathbf{S}$ and to shrink the vector x accordingly. This is done by creating a superposition, over all possible values of $y' \in \mathbf{S}$, of all tuples of the form $(\mathbf{S} \setminus \{y'\}, x_{\mathbf{S}\setminus\{y'\}}, y')$[3].
3. Measure the final state and check if $\mathbf{S}$ contains a collision.

Theorem 4 states that this algorithm indeed finds a collision with constant probability. Moreover, it has a *query* complexity of r to create the initial superposition, and $t_1 \times t_2$ for the second part of the algorithm, which leads to an optimal value of $r = \mathcal{O}\left(N^{k/(k+1)}\right)$. In the following sections, we discuss the time complexity of the algorithm in more details.

7.2 Childs and Eisenberg algorithm

We start by noting that Ambainis' algorithm does not necessarily deal with collisions. Indeed, it is a quantum search algorithm that finds a k-tuple of elements in a set that satisfy a specific relation. Moreover, it does so by relying on a quantum walk over a r-tuple of elements in this same set. In particular, the query complexity of Ambainis' algorithm could be generalized to *any* relation that involves exactly k indices, using exactly the same procedure, and with the same *query* complexity [CE05]. A critical part of the analysis from [Amb04] is that the *time* complexity of the collision-finding algorithm actually matches its query complexity. In order to achieve this, Ambainis relies on a clever combination of data structures in order to satisfy the following constraints:

- all manipulation of the data structures can be made in a reversible way,
- the time complexity of search, insertion and deletion has to be (poly-)logarithmic,
- the time complexity of *checking if the relation is satisfied* has to be smaller than $t_2 = \mathcal{O}(\sqrt{r})$.

The last point stems from the fact that step 2.a of the algorithm (the conditional phase flip) should not be more time-consuming than step 2.b (t_2 iterations of the quantum walk). This remark calls for the following definition.

[3] We stress that this superposition is not uniform, which is critical in the proof of Theorem 4. However, the corresponding amplitudes will not be relevant for our discussion, which is why we chose to ignore them for the sake of simplicity.

Definition 1. *A k-ary relation $\mathcal{R}$ over $\mathcal{X}^k$ is said to be (f, g, r)-testable if there exists a data structure to represent subsets of $\{1, \ldots, |\mathcal{X}|\}$ such that the following properties hold:*

- *it uses $\widetilde{\mathcal{O}}(s)$ qubits of memory to store a subset of elements of size $s \leq r + 1$,*
- *insertion and deletion of an element is (poly)-logarithmic and requires a single query,*
- *testing if there exists a tuple $(i_1, \ldots, i_k)$ of k distinct elements in a set of size r such that $(x_{i_1}, \ldots, x_{i_k}) \in \mathcal{R}$ takes a time at most $f(r)$, and requires at most $g(r)$ queries.*

Then, one has the following result.

Theorem 5. *[CE05] Let $r \geq k$, $r = o(N)$, and let $\mathcal{R}$ be a (f, g, r)-testable k-ary relation over $\mathcal{X}^k$ for some functions f and g. There is a quantum algorithm that outputs a k-tuple $(i_1, \ldots, i_k)$ of distinct elements such that $(x_{i_1}, \ldots, x_{i_k}) \in \mathcal{R}$ with $\mathcal{O}\left(\max\left(\left(\frac{N}{r}\right)^{k/2}(\sqrt{r} + g(r)), r\right)\right)$ queries using $\widetilde{\mathcal{O}}(r)$ qubits of memory.*

Proof. Childs and Eisenberg proved this theorem in [CE05].

Corollary 1. *This quantum algorithm takes $\widetilde{\mathcal{O}}\left(\max\left(\left(\frac{N}{r}\right)^{k/2}(\sqrt{r} + f(r)), r\right)\right)$ in time.*

Remark 7. In [Amb04], Ambainis proves that the k- collision relation is actually $\left(\widetilde{\mathcal{O}}(1), 0, r\right)$-testable when $r = o(N)$.

7.3 On data structures

In [Amb04], Ambainis' shows that actual running time of the quantum algorithm is in $\widetilde{\mathcal{O}}(N^{k/k+1})$. We will recall in this section the data structures that Ambainis relies on for the k-distinctness problem to achieve this running time.

Required operations. The required operations are the following:

1. Adding y to $\mathbf{S}$ and storing x_y.
2. Removing y from $\mathbf{S}$ and erasing x_y.
3. Checking if $\mathbf{S}$ contains the solution to perform the conditional phase flip.
4. Applying the diffusion operator.
5. Storing the same set $\mathbf{S}$ in the same way independently of how $\mathbf{S}$ was created.
6. Using classical subroutines that must terminate in fixed time t.

The data structures used is a combination between hash table and a skip list. In the following, v is set as a variable that tracks the count of distinct $x \in [M]$ for which the set $\mathbf{S}$ contains $i_1, \ldots, i_k$ such that $x_{i_1} = \cdots = x_{i_k}$.

Hash table. The hash table is used for storing pairs (i, x_i) and retrieving them when searching for x_i corresponding to a given i.

The hash table consists of r buckets, each equipped with memory for $\lceil \log N \rceil$ entries and $\lfloor \log r \rfloor$ counters. To determine the placement of entries, the set $\{1, \ldots, N\}$ is hashed into the r buckets using the function $h(i) = \lfloor i \cdot r/N \rfloor + 1$. The j-th bucket organizes pairs (i, x_i) for $i \in \mathbf{S}$ such that $h(i) = j$, in ascending order of i. In cases where there are more than $\lceil \log N \rceil$ entries with $h(i) = j$, only $\lceil \log N \rceil$ of them are stored in the bucket. Ambainis demonstrates that the probability of this occurrence is small.

As for the counters, denoted as $d_1, \ldots, d_{\lfloor \log r \rfloor}$, the counter d_1 in the j-th bucket counts the number of $i \in \mathbf{S}$ such that $h(i) = j$. Additionally, d_l for $l > 1$ is used only if j is divisible by 2^l. In such instances, it counts the number of $i \in \mathbf{S}$ satisfying $j - 2^l + 1 \leq h(i) \leq j$.

Skip list. The skip list is used for organizing the x_i in ascending order. Consequently, when adding a new element i to $\mathbf{S}$, we can verify whether x_i matches any of the x_j values for $j \in \mathbf{S}$. For a more detailed understanding of how this skip list operates, please refer to [Pug90],[Amb04].

Each element $i \in \mathbf{S}$ is assigned a random level l_i ranging from 0 to $l_{\max} = \lceil \log N \rceil$. To chose the level independently for each i, we define the levels using $l_{\max}$ functions $h_1, h_2, ..., h_{l_{\max}} : [N] \to \{0, 1\}$. If $h_1(i) = \ldots = h_l(i) = 1$ but $h_{l+1}(i) = 0$ then $i \in [N]$ is associated to level l. If $h_1(i) = \ldots = h_{l_{\max}}(i) = 1$, it implies that the element $i \in [N]$ is associated with level $l_{\max}$. The reason for assigning the level in this manner is due to a drawback in the straightforward implementation.

Insertion and deletion. To include i in the set $\mathbf{S}$, we first query the value x_i. Then, we calculate $h(i)$ and add (i, x_i) into the $h(i)$-th bucket. If the bucket already contains some entries, we may move some of them to ensure that, after the insertion of (i, x_i), the entries are still in the order of increasing i. Following this, we increment the counter d_1 for the $h(i)$-th bucket and the counter d_l for the $(\lceil \frac{h(i)}{2^l} \rceil \cdot 2^l)$-th bucket, for each $l \in \{2, \ldots, \lfloor \log r \rfloor\}$. Following that, we modify the skip list as follows: first, run a search to locate the last element preceding i. Then, for every level $l \in \{0, ..., l_i\}$ assign j_l to be the level-l pointer of i_l. Set the level-l pointer of i to be equal to j_l and the level-l pointer of i_l to be equal to i. Once the update is finished, we use the skip list to identify the smallest j where $x_j = x_i$. We then use level-0 pointers to determine whether the count of j with $x_j = x_i$ is less than k, exactly k, or exceeds k. If there are exactly k such j, we increase v by 1.

An element i can be deleted from $\mathbf{S}$ by running this procedure in reverse.

Checking for k-collisions. To find k-collisions in $\mathbf{S}$, we simply verify whether $v > 0$.

Diffusion transform. Based on Grover's result [Gro96], Ambainis demonstrates that the algorithm requires approximately $\widetilde{\mathcal{O}}(1)$ time for the diffusion process.

Uniqueness. $\mathbf{S}$ is always stored in the same way. The values $i \in \mathbf{S}$ are consistently hashed to buckets by h, and within each bucket, the entries are arranged in increasing order of i. The counters, responsible to count the number of entries in the buckets, are uniquely defined by the set $\mathbf{S}$. Similarly, the configuration of the skip list is also uniquely established.

Guaranteed running time. Ambainis demonstrates that the probability of lookup, insertion, or deletion of an element taking more than $\widetilde{\mathcal{O}}(1)$ steps is negligible for any set $\mathbf{S}$. In such instances, the algorithm is adjusted for lookup, insertion, or deletion, causing them to abort after $\widetilde{\mathcal{O}}(1)$ steps. It is highlighted in [Amb04] that this modification has no significal impact on the entire quantum search algorithm.

7.4 Application to the case of Feistel networks

We use the same type of data structure mentioned in the Section 7.3, but with a few small adjustments.

In the following, v is set as a variable that tracks the count of distinct solutions of the System $(\mathcal{S})$ that are included in $\mathbf{S}$.

Skip list. For a Feistel scheme with 6 rounds, we need two skip lists instead of one. Specifically, the first one is used to store $S||R \oplus T$. The second skip list stores $R||L \oplus S$.

Insertion and deletion. To include i in the set $\mathbf{S}$, we first query the value $O(L, R)$[4]. Then, we calculate $h(i)$ and add $(i, (S_i, T_i))$ into the $h(i)$-th bucket. After that, we update the skip list and then we update the value of v by adding the value of tuple of indices that satisfy the System $(\mathcal{S})$ and include the new index i. This can take the form of $(i, j, k, l) \in \mathbf{S}^4$ that satisfies the System $(\mathcal{S})$, or $(j, i, k, l) \in S^4$ that satisfies the System $(\mathcal{S})$, or $(j, k, i, l) \in \mathbf{S}^4$ that satisfies the System $(\mathcal{S})$, or $(j, k, l, i) \in \mathbf{S}^4$ that satisfies the System $(\mathcal{S})$. For example, to verify if (i, j, k, l) is a solution, it suffices to check whether $S_j||R_j \oplus T_j = S_i||R_i \oplus T_i$ and $R_k||L_k \oplus S_k = R_i||L_i \oplus S_i$ and $R_l||L_l = R_j||L_i \oplus L_j \oplus L_k$. Another instance is checking if (j, i, k, l) is a solution, for which the verification involves ensuring that $S_i||R_i \oplus T_i = S_j||R_j \oplus T_j$ and $R_l||L_l \oplus S_l = R_j||L_j \oplus S_j$ and $S_k||T_k = S_l||T_i \oplus T_j \oplus T_l$.

An element i can be deleted from $\mathbf{S}$ by running this procedure in reverse.

Checking for k-collisions. To find collisions in $\mathbf{S}$, we simply verify whether $v > 0$.

Guaranteed running time. We need a guaranteed running time when testing if a new entry can be query $1, 2, 3$, of a tuple of solution. Let ℓ_1 (resp. ℓ_2) be the length of the longest $[S, R \oplus T]$ (resp. $[R, L \oplus S]$) line[5], where we say that we

[4] The oracle O is implemented either with a uniformly random permutation P or a 6-round Feistel scheme Ψ.

[5] This is defined over the full message space.

have a $[A, B]$ line of length ℓ if there exists pairwise distinct indices $i_1, \ldots, i_\ell$ such that $(A_{i_1}, B_{i_1}) = \cdots = (A_{i_\ell}, B_{i_\ell})$. In this case, the complexity of checking if a new entry triggers the completion of a new tuple is $\widetilde{\mathcal{O}}(\ell_1\ell_2)$. However, for our attack to work, we need to be able to guarantee that the insertion or deletion of an element runs in time $\widetilde{\mathcal{O}}(1)$, except with a small probability over the uniformly random sampling of the $2n$-bit permutation or the 6 n-bit random functions. In Appendix B, we provide a complete proof of the fact that $\ell_1, \ell_2 = \widetilde{\mathcal{O}}(1)$ with a probability that is close to 1, which is sufficient to conclude the analysis of our quantum attack. While Appendix B is quite long, the ideas behind the proof are quite simple, and we briefly outline them here. In the case of a random permutation, the experiment is close to the classical balls-into-bins problem (with a moderate dependence between throws), which explains why the length of a line is logarithmic with overwhelming probability. In the case of a Feistel network, collisions can come from two possible avenues:

- purely random collisions that occur from a degree of freedom during the evaluation of the Feistel network;
- structural collisions that are forced due to internal collisions with previous queries.

The first category behaves close to a standard balls-into-bins problem, and also yield lines of logarithmic length with overwhelming probability. The second type of collisions, although less numerous, are more difficult to analyze. However, the intuition behind our result is that, in such a structural collision brings at least two independent equations per query involved in the collision. Hence, the average number of such collisions is close to 1, which means that it will be logarithmic with a probability that is also close to 1. This is sufficient to ensure that Child and Eisenberg's algorithm will find solutions to the System (4) with a non-negligible probability, and with a query complexity of $O(2^{8n/5})$ and a time complexity of $\widetilde{\mathcal{O}}(2^{8n/5})$.

References

Amb04. Andris Ambainis. Quantum walk algorithm for element distinctness. In *45th FOCS*, pages 22–31. IEEE Computer Society Press, October 2004.

CE05. Andrew M. Childs and Jason M. Eisenberg. Quantum algorithms for subset finding. *Quantum Inf. Comput.*, 5(7):593–604, 2005.

CLL+14. Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John P. Steinberger. Minimizing the two-round Even-Mansour cipher. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 39–56. Springer, Heidelberg, August 2014.

CPT22. Maya Chartouny, Jacques Patarin, and Ambre Toulemonde. Quantum cryptanalysis of 5 rounds feistel schemes and benes schemes. *IACR Cryptol. ePrint Arch.*, page 1015, 2022.

Gro96. Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

IHM+19. Gembu Ito, Akinori Hosoyamada, Ryutaroh Matsumoto, Yu Sasaki, and Tetsu Iwata. Quantum chosen-ciphertext attacks against feistel ciphers. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, volume 11405 of *Lecture Notes in Computer Science*, pages 391–411. Springer, 2019.

KM10. Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round feistel cipher and the random permutation. In *IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010, Austin, Texas, USA, Proceedings*, pages 2682–2685. IEEE, 2010.

NVP13. Valérie Nachef, Emmanuel Volte, and Jacques Patarin. Differential attacks on generalized Feistel schemes. In Michel Abdalla, Cristina Nita-Rotaru, and Ricardo Dahab, editors, *CANS 13*, volume 8257 of *LNCS*, pages 1–19. Springer, Heidelberg, November 2013.

Pat01. Jacques Patarin. Generic attacks on Feistel schemes. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 222–238. Springer, Heidelberg, December 2001.

Pat08. Jacques Patarin. Generic attacks on feistel schemes. *IACR Cryptol. ePrint Arch.*, page 36, 2008.

Pug90. William Pugh. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.

TP09. Joana Treger and Jacques Patarin. Generic attacks on feistel networks with internal permutations. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, volume 5580 of *Lecture Notes in Computer Science*, pages 41–59. Springer, 2009.

Wag99. David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *FSE'99*, volume 1636 of *LNCS*, pages 156–170. Springer, Heidelberg, March 1999.

Zha13. Mark Zhandry. A note on the quantum collision and set equality problems. *CoRR*, abs/1312.1027, 2013.

A Proof of Lemma 1

Let $\mathbf{B} = (B_i)_{1 \leq i \leq q}$ be independent random variables such that

$$\Pr[B_i = 1] = \varepsilon_i \qquad \text{and} \qquad \Pr[B_i = 0] = 1 - \varepsilon_i,$$

which is to say that $B_i \sim \mathsf{Ber}_{\varepsilon_i}$, where Ber_p denotes the Bernoulli distribution of parameter p.

Following [CLL+14], we will use a coupling argument to prove that, for any r, one has

$$\Pr\left[\sum_{i=1}^{q} A_i \geq r\right] \leq \Pr\left[\sum_{i=1}^{q} B_i \geq r\right].$$

Let us consider the sampling process outlined in Algorithm 1. By construction, one has $(a_1, \ldots, a_q) \sim \mathbf{A}$. Moreover, we also have $(b_1, \ldots, b_q) \sim \mathbf{B}$. Indeed, let us

Algorithm 1 Sampling procedure for the coupling argument of the proof of Lemma 1

for $i = 1$ to q **do**
  $p \leftarrow \Pr[A_i = 1 \,|\, (A_1, \ldots, A_{i-1}) = (a_1, \ldots, a_{i-1})]$
  $a_i \leftarrow \mathsf{Ber}_p$
  if $a_i = 1$ **then**
    $b_i \leftarrow 1$
  else
    $p'_i \leftarrow \frac{\varepsilon_i - p}{1-p}$
    $b_i \leftarrow \mathsf{Ber}_{p'_i}$
  end if
end for
return $((a_1, \ldots, a_q), (b_1, \ldots, b_q))$

fix any $i = 1, \ldots, q$ and any binary sequence $(b_1, \ldots, b_{i-1})$. Then, one has

$$\begin{aligned}\Pr[b_i = 1] &= \Pr[a_i = 1] + \Pr\left[(a_i = 0) \wedge \mathsf{Ber}_{p'_i} = 1\right] \\ &= p + (1-p)\frac{\varepsilon_i - p}{1-p} \\ &= \varepsilon_i.\end{aligned}$$

Moreover, since we assume $\varepsilon_i < 1$, then $p \leq \varepsilon_i < 1$, and $\mathsf{Ber}_{p'_i}$ is well-defined.

We now remark that, in the sampling process, if $a_i = 1$, then b_i is also set to 1. This implies that, for any r, we indeed have

$$\sum_{i=1}^{q} a_i \leq r \implies \sum_{i=1}^{q} b_i \leq r,$$

which in turn gives

$$\Pr\left[\sum_{i=1}^{q} A_i \geq r\right] \leq \Pr\left[\sum_{i=1}^{q} B_i \geq r\right].$$

Remark that $\mathbf{B}$ is a tuple of independent Bernoulli random variables, that are not necessarily identically distributed. Thus, applying a classical Chernoff bound tailored to this case yields

$$\Pr\left[\sum_{i=1}^{q} A_i \geq (1+\delta)m\right] \leq \Pr\left[\sum_{i=1}^{q} B_i \geq (1+\delta)m\right] \leq e^{-\frac{\delta^2}{2+\delta}m}$$

for any $\delta > 0$.

B On the maximum line length

B.1 Permutation case.

In this section, we consider the random variable ℓ which corresponds to the length of the longest $[R, L \oplus S]$ line (the other case can be treated similarly by

symmetry). Let us start by fixing any arbitrary ordering over $\{0,1\}^n$ such that

$$\{0,1\}^n = \{L_1, \ldots, L_{2^n}\}.$$

For any $A, B \in \{0,1\}^n$, let $\ell_{A,B}$ be the number of indices $i \in \{1, \ldots, 2^n\}$ such that $P(L_i||A) = (B \oplus L_i)||*$. Then, one has $\ell = \max_{A,B}(\ell_{A,B})$ and, for any c,

$$\Pr[\ell \geq c] \leq \sum_{A,B} \Pr[\ell_{A,B} \geq c].$$

Let us fix any $A, B \in \{0,1\}^n$ and, for all $i = 1, \ldots, 2^n$, let $\mathcal{Y}_i$ be the random variable that is equal to 1 if $P(L_i||A) = (B \oplus L_i)||*$, and to 0 otherwise. It is easy to see that

$$\ell_{A,B} = \sum_{i=1}^{2^n} \mathcal{Y}_i.$$

Let us fix any $i \in \{1, \ldots, 2^n\}$, and any $(y_1, \ldots, y_{i-1}) \in \{0,1\}^{i-1}$. Then, one has

$$\Pr[\mathcal{Y}_i = 1 \,|\, (\mathcal{Y}_1, \ldots, \mathcal{Y}_{i-1}) = (y_1, \ldots, y_{i-1})] \leq \varepsilon_i,$$

where

$$\varepsilon_i = \frac{2^n}{2^{2n} - i + 1} = \frac{1}{2^n} + \frac{i-1}{2^{3n} - (i-1)2^n}.$$

Hence, one has

$$1 \leq 1 + \frac{(2^n - 1)}{2^{2n+1}} \leq \sum_{i=1}^{2^n} \varepsilon_i \leq 1 + \frac{1}{2^{n+1} - 2} \leq \frac{3}{2}.$$

Let us now apply Lemma 1 to $(Y_i)_{1 \leq i \leq 2^n}$. One has

$$\Pr\left[\ell_{A,B} \geq \frac{3}{2}(1+\delta)\right] \leq \Pr\left[\ell_{A,B} \geq (1+\delta)\sum_{i=1}^{2^n} \varepsilon_i\right] \leq e^{-\frac{\delta^2}{2+\delta}\sum_{i=1}^{2^n} \varepsilon_i} \leq e^{-\frac{\delta^2}{2+\delta}}.$$

Using the fact that $n \geq 1$ and $e \geq 2$, and taking $\delta = \frac{9n}{2}$, we get

$$\Pr\left[\ell_{A,B} \geq 9n\right] \leq \Pr\left[\ell_{A,B} \geq \frac{3}{2}(1 + \frac{9n}{2})\right] \leq e^{-3n} \leq 2^{-3n}.$$

Hence, after summing over all possible values for A, B, we finally have

$$\Pr[\ell \geq 9n] \leq \frac{1}{2^n}.$$

B.2 Feistel case.

Let us fix $n \geq 14$. We will split the study in two parts, depending on the number of fresh inputs of the queries that appear in the line. Let ℓ' be the length of

the longest free line in $[R, L \oplus S]$ (i.e. a line such that, for each involved query, the intermediate input of *at least one function* among f_3, f_4 and f_5 is fresh). Let N be the number of queries that are part of a $[R, L \oplus S]$ line and whose intermediate inputs of f_3, f_4 and f_5 all collide with another query from the same line. It is clear that $\ell \leq \ell' + N$. We will prove that $\Pr[\ell' \geq 4n] \leq \frac{1}{2^{3n}}$, and $\Pr[N \geq 4n] \leq \frac{1+\frac{6}{2^n}}{4n}$. Overall, this will prove that

$$\Pr[\ell \geq 8n] \leq \Pr[\ell' \geq 4n] + \Pr[N \geq 4n] \leq \frac{1}{2^{3n}} + \frac{1+\frac{6}{2^n}}{4n}.$$

First scenario. Let us fix any $A, B \in \{0,1\}^n$, $i \in \{1, \ldots, 2^n\}$ and any $(y_1, \ldots, y_{i-1}) \in \{0,1\}^{i-1}$. For any set $I = \{i_1, \ldots, i_k\}$ of $4n$ pairwise distinct indices in $\{1, \ldots, 2^n\}$, we let $\ell_{A,B}(I)$ denote the event

$$\Psi^6(L_{i_j}||A) = (B \oplus L_{i_j})|| * \qquad \text{for } i = 1, \ldots, 4n.$$

If we rewrite this equality in terms of the intermediate variables X_{i_j} and Y_{i_j}, this equality becomes

$$f_3\left(Y_{i_j}\right) \oplus f_5\left(Y_{i_j} \oplus f_4\left(X_{i_j} \oplus f_3\left(Y_{i_j}\right)\right)\right) = B \oplus f_1(A). \tag{9}$$

Let us fix $j \in \{1, \ldots, 4n\}$, and let us assume that Eq. (9) holds for all $j' < j$. The event "Eq. (9) holds for the j-th query" is included in the following set of events:

1. the input to function f_5 is fresh (i.e. it is does not collide with any previous inputs to f_5), and Eq. (9) holds;
2. there exists $j' < j$ such that

$$\begin{aligned} Y_{i_j} \oplus f_4\left(X_{i_j} \oplus f_3\left(Y_{i_j}\right)\right) &= Y_{i_{j'}} \oplus f_4\left(X_{i_{j'}} \oplus f_3\left(Y_{i_{j'}}\right)\right), \\ f_3(Y_{i_j}) &= f_3(Y_{i_{j'}}) \qquad \text{(from Eq. (9))}, \end{aligned}$$

   but $X_{i_j} \oplus f_3\left(Y_{i_j}\right)$ is fresh;
3. there exists $j', j'' < j$ such that

$$\begin{aligned} Y_{i_j} \oplus f_4\left(X_{i_j} \oplus f_3\left(Y_{i_j}\right)\right) &= Y_{i_{j'}} \oplus f_4\left(X_{i_{j'}} \oplus f_3\left(Y_{i_{j'}}\right)\right), \\ X_{i_j} \oplus f_3\left(Y_{i_j}\right) &= X_{i_{j''}} \oplus f_3\left(Y_{i_{j''}}\right), \\ f_3(Y_{i_j}) &= f_3(Y_{i_{j'}}) \qquad \text{(from Eq. (9))}, \end{aligned}$$

   but Y_{i_j} is fresh.

Overall, in all cases, at least one value will be chosen uniformly at random in $\{0,1\}^n$. Indeed, one has $X_{i_j} = L_{i_j} \oplus f_1(A)$ and $Y_{i_j} = A \oplus f_2(X_{i_j})$. This means that the X_i values are always pairwise distinct[6], and the Y_i values are chosen

[6] Recall that $X_i = L_i \oplus f_1(R_i) = L_i \oplus f_1(A)$. This means, in a $[R, L \oplus S]$ line, all L and X values have to be pairwise distinct.

uniformly and independently at random. We now consider all the probabilities of all three cases occurring.
Case 1: Since the input to f_5 is fresh, this case occurs with a probability lower than

$$\frac{1}{2^n}.$$

Case 2: There are at most $j-1$ possible choices for j'. Since the input to f_4 is fresh, the first equation holds with a probability 2^{-n}. For the second equation, two cases can occur: either Y_{i_j} is fresh (which means that the second equation occurs with a probability 2^{-n}), or Y_{i_j} is not fresh, which occurs with a probability at most $(j-1)/2^n$. Overall, the probability of this case occurring is lower than

$$\frac{(j-1)+(j-1)^2}{2^{2n}} \leq \frac{2(4n)^2}{2^{2n}} \leq \frac{1}{2 \cdot 2^n},$$

since $2(4n)^2 \leq 2^n/2$ when $n \geq 14$.
Case 3: One has $X_{i_j} \oplus f_3\left(Y_{i_j}\right) = X_{i_{j''}} \oplus f_3\left(Y_{i_{j''}}\right)$. This means that the first equation can be rewritten as

$$Y_{i_j} \oplus f_4\left(X_{i_{j''}} \oplus f_3\left(Y_{i_{j''}}\right)\right) = Y_{i_{j'}} \oplus f_4\left(X_{i_{j'}} \oplus f_3\left(Y_{i_{j'}}\right)\right),$$

which happens with a probability 2^{-n} over the choice of Y_{i_j}. Since Y_{i_j} is fresh, the third equation will also occur with a probability at most 2^{-n}. Overall, this case happens with probability smaller than

$$\frac{(j-1)^2}{2^{2n}} \leq \frac{1}{2 \cdot 2^n}.$$

Hence, the probability that Eq. (9) holds for the j-th query is smaller than $2/2^n$. Thus, taking a product over j yields

$$\Pr[\ell_{A,B}(I) \geq 4n] \leq \frac{2^{4n}}{2^{4n^2}}.$$

Summing over all possible choices for I, A, B yields

$$\Pr[\ell' \geq 4n] \leq \frac{2^{(4n+2)n+4n}}{(4n)! \cdot 2^{4n^2}} = \frac{2^{6n}}{(4n)!}.$$

One has $(4n)! \geq (2n)^{2n} \cdot (2n)! \geq 2^{2n} n^{3n}$, which gives

$$\Pr[\ell' \geq 4n] \leq \frac{2^{4n}}{n^{3n}} \leq \left(\frac{3}{n}\right)^{3n} \leq \frac{1}{2^{3n}}$$

since $n \geq 14$.

Second scenario. Here, we want to upper-bound the probability that there exist 4 indices i_1, i_2, i_3, i_4 such that:

- all indices belong to the same $[R, L \oplus S]$ line;
- $Y_{i_4} = Y_{i_3}$;
- $X_{i_4} \oplus f_3(Y_{i_4}) = X_{i_2} \oplus f_3(Y_{i_2})$ (i.e. $Z_{i_4} = Z_{i_2}$);
- $Y_{i_4} \oplus f_4(X_{i_4} \oplus f_3(Y_{i_4})) = Y_{i_1} \oplus f_4(X_{i_1} \oplus f_3(Y_{i_1}))$ (i.e. $U_{i_4} = U_{i_1}$).
- all indices are the first indices (for an arbitrary ordering of the queries) that satisfies these conditions; this means that all queries i_1, i_2, i_3 have at least one input to f_3, f_4, or f_5 that is fresh.

In particular, these conditions imply that

- $i_4 \neq i_3, i_2, i_1$;
- $f_3(Y_{i_4}) = f_3(Y_{i_3}) = f_3(Y_{i_1})$ (since $U_{i_4} = U_{i_1}$ and $L_{i_4} \oplus S_{i_4} = L_{i_1} \oplus S_{i_1}$, which corresponds to Eq (9);
- $i_3 \neq i_2$ and $i_2 \neq i_1$ since otherwise the queries would be identical due to the bijectivity of Feistel networks and the fact that the collision of two successive intermediate values imply a full-state collision (but it may happen that $i_3 = i_1$);
- $Y_{i_4} \neq Y_{i_2}$, since otherwise $X_{i_4} = X_{i_2}$, which is impossible as $i_4 \neq i_2$.

We now divide our analysis in several subcases.

1. All 4 indices are pairwise distinct. The equality $Y_{i_4} = Y_{i_3}$ happens with probability 2^{-n} due to the fact that all queries belong to the same line, which implies that all X values are pairwise distinct, and thus all Y values are uniformly random and independent. Now, the equality $X_{i_4} \oplus f_3(Y_{i_4}) = X_{i_2} \oplus f_3(Y_{i_2})$ becomes $X_{i_4} \oplus f_3(Y_{i_3}) = X_{i_2} \oplus f_3(Y_{i_2})$ and three possible cases can occur:
   (a) Y_{i_3} is fresh, which means that the equality occurs with probability 2^{-n}. Since $Y_{i_4} = Y_{i_3} \neq Y_{i_1}$, the equality $f_3(Y_{i_4}) = f_3(Y_{i_1})$ occurs with a probability 2^{-n}. Let us now study the equality $Y_{i_4} \oplus f_4(X_{i_4} \oplus f_3(Y_{i_4})) = Y_{i_1} \oplus f_4(X_{i_1} \oplus f_3(Y_{i_1}))$, which is equivalent to $Y_{i_3} \oplus f_4(X_{i_2} \oplus f_3(Y_{i_2})) = Y_{i_1} \oplus f_4(X_{i_1} \oplus f_3(Y_{i_1}))$. Due to the fact that $Y_{i_3} \neq Y_{i_2}$, this is impossible in the case where both inputs to f_4 are equal. Hence, the inputs have to be different (i.e. $Z_{i_1} \neq Z_{i_2}$) and the collision happens with a probability smaller than 2^{-n}. Finally, we need to consider the condition $L_{i_1} \oplus S_{i_1} = L_{i_2} \oplus S_{i_2} = L_{i_3} \oplus S_{i_3}$. We start by studying the equality

$$f_3(Y_{i_1}) \oplus f_5(U_{i_1}) = f_3(Y_{i_2}) \oplus f_5(U_{i_2}).$$

   If $U_{i_1} \neq U_{i_2}$, this equality happens with a probability 2^{-n}. On the contrary, it means that $Z_{i_2} \neq Z_{i_1}$, i.e. and $U_{i_1} = U_{i_2}$ becomes $Y_{i_1} \oplus Y_{i_2} = f_4(Z_{i_1}) \oplus f_4(Z_{i_2})$, which is also equal to $Y_{i_3} \oplus Y_{i_1}$. This would imply that $Y_{i_2} = Y_{i_3}$, which is impossible. For the equality

$$f_3(Y_{i_1}) \oplus f_5(U_{i_1}) = f_3(Y_{i_3}) \oplus f_5(U_{i_3}),$$

   we need to consider three sub-cases.

i. If U_{i_3} is fresh, it occurs with probability 2^{-n}.
ii. $U_{i_3} = U_{i_1}$: this is equivalent to $Y_{i_3} \oplus f_4(Z_{i_3}) = Y_{i_1} \oplus f_4(Z_{i_1})$, and in that case $L_{i_1} \oplus S_{i_1} = L_{i_3} \oplus S_{i_3}$ is automatically satisfied as $f_3(Y_{i_3}) = f_3(Y_{i_4}) = f_3(Y_{i_1})$. Note that $Z_{i_3} = X_{i_3} \oplus f_3(Y_{i_3}) \neq X_{i_1} \oplus f_3(Y_{i_1}) = Z_{i_1}$ since $f_3(Y_{i_4}) = f_3(Y_{i_3}) = f_3(Y_{i_1})$ and $X_{i_1} \neq X_{i_3}$ (because $i_1 \neq i_3$). Moreover, one has $X_{i_4} \oplus X_{i_2} = f_3(Y_{i_3}) \oplus f_3(Y_{i_2}) = Z_{i_3} \oplus X_{i_3} \oplus Z_{i_2} \oplus X_{i_2}$, which implies that $Z_{i_3} \neq Z_{i_2}$ since $i_3 \neq i_4$. Thus, this case also happens with probability 2^{-n}.
iii. $U_{i_3} = U_{i_2}$: as in the previous case, it happens with a probability smaller than 2^{-n}.

(b) $Y_{i_3} = Y_{i_2}$: this implies that $Y_{i_4} = Y_{i_2}$, which is impossible.

(c) $Y_{i_3} = Y_{i_1}$: this happens with probability 2^{-n}, and also implies that $f_3(Y_{i_4}) = f_3(Y_{i_1})$, which automatically yields $L_{i_4} \oplus S_{i_4} = L_{i_1} \oplus S_{i_1}$. Moreover, since $Y_{i_1} (= Y_{i_4}) \neq Y_{i_2}$, the equality $X_{i_4} \oplus f_3(Y_{i_3}) = X_{i_2} \oplus f_3(Y_{i_2})$ becomes $X_{i_4} \oplus f_3(Y_{i_1}) = X_{i_2} \oplus f_3(Y_{i_2})$ and happens with probability 2^{-n}. Let us now study the equality $Y_{i_4} \oplus f_4(X_{i_4} \oplus f_3(Y_{i_4})) = Y_{i_1} \oplus f_4(X_{i_1} \oplus f_3(Y_{i_1}))$, which is equivalent to $f_4(X_{i_2} \oplus f_3(Y_{i_2})) = f_4(X_{i_1} \oplus f_3(Y_{i_1}))$ (i.e. $f_4(Z_{i_1}) = f_4(Z_{i_2})$). Due to the previous equality, one necessarily has $X_{i_2} \oplus f_3(Y_{i_2}) \neq X_{i_1} \oplus f_3(Y_{i_1})$, since otherwise we would have $X_{i_1} = X_{i_4}$. Hence, the equality also occurs with probability 2^{-n}. Since $Y_{i_1} \neq Y_{i_2}$, the equality $f_4(X_{i_2} \oplus f_3(Y_{i_2})) = f_4(X_{i_1} \oplus f_3(Y_{i_1}))$ gives $Z_{i_1} \neq Z_{i_2}$. Let us now consider the equality $f_3(Y_{i_1}) \oplus f_5(U_{i_1}) = f_3(Y_{i_2}) \oplus f_5(U_{i_2})$. If $U_{i_1} \neq U_{i_2}$, this happens with probability 2^{-n}. On the contrary, it means that $Y_{i_1} \oplus Y_{i_2} = f_4(Z_{i_1}) \oplus f_4(Z_{i_2}) = 0$, which is impossible. For the equality

$$f_3(Y_{i_1}) \oplus f_5(U_{i_1}) = f_3(Y_{i_3}) \oplus f_5(U_{i_3}),$$

we need to consider three sub-cases.
i. If U_{i_3} is fresh, it occurs with probability 2^{-n}.
ii. $U_{i_3} = U_{i_1}$: this is equivalent to $f_4(Z_{i_3}) = f_4(Z_{i_1})$, and in that case $L_{i_1} \oplus S_{i_1} = L_{i_3} \oplus S_{i_3}$ is automatically satisfied. Note that $Z_{i_3} = X_{i_3} \oplus f_3(Y_{i_3}) \neq X_{i_1} \oplus f_3(Y_{i_1}) = Z_{i_1}$ since $f_3(Y_{i_4}) = f_3(Y_{i_3}) = f_3(Y_{i_1})$ and $X_{i_1} \neq X_{i_3}$ (because $i_1 \neq i_3$). Moreover, one has $X_{i_4} \oplus X_{i_2} = f_3(Y_{i_3}) \oplus f_3(Y_{i_2}) = Z_{i_3} \oplus X_{i_3} \oplus Z_{i_2} \oplus X_{i_2}$, which implies that $Z_{i_3} \neq Z_{i_2}$ since $i_3 \neq i_4$. Thus, this case also happens with probability 2^{-n}.
iii. $U_{i_3} = U_{i_2}$: as in the previous case, it happens with a probability smaller than 2^{-n}.

2. One has $i_3 = i_1$. Then, the equality $Y_{i_4} = Y_{i_3}$ happens with probability 2^{-n}. However, in this case, $f_3(Y_{i_4}) = f_3(Y_{i_1})$ is automatic. As in the previous case, the equality $X_{i_4} \oplus f_3(Y_{i_4}) = X_{i_2} \oplus f_3(Y_{i_2})$ becomes $X_{i_4} \oplus f_3(Y_{i_1}) = X_{i_2} \oplus f_3(Y_{i_2})$, with $Y_{i_1} \neq Y_{i_2}$ due to the fact that $Y_{i_1} = Y_{i_3} = Y_{i_4} \neq Y_{i_2}$. Hence, this equality occurs with a probability 2^{-n}. Let us now study the equality $Y_{i_4} \oplus f_4(X_{i_4} \oplus f_3(Y_{i_4})) = Y_{i_1} \oplus f_4(X_{i_1} \oplus f_3(Y_{i_1}))$, which is equivalent to $f_4(X_{i_2} \oplus f_3(Y_{i_2})) = f_4(X_{i_1} \oplus f_3(Y_{i_1}))$ (i.e. $f_4(Z_{i_1}) = f_4(Z_{i_2})$). Due to the fact that $X_{i_4} \oplus X_{i_2} = f_3(Y_{i_1}) \oplus f_3(Y_{i_2})$, both inputs to f_4 have to be different

(i.e. $Z_{i_1} \neq Z_{i_2}$) and the collision happens with a probability smaller than 2^{-n}. Finally, we need to consider the condition $L_{i_1} \oplus S_{i_1} = L_{i_2} \oplus S_{i_2}$. Since $f_4(Z_{i_1}) = f_4(Z_{i_2})$ and $Y_{i_1} \neq Y_{i_2}$, one has $U_{i_1} \neq U_{i_2}$, which means that this equation happens with probability 2^{-n}.

Final bound. Overall, the number of possible choices for the indices of the first case is at most $2^{2n} \times 2^{3n}$: there are 2^n choices for i_1, and then 2^n choices for the three remaining indices since the R value has to be constant. Hence, the probability of the first case adds a query to a free line is at most $6/2^n$, which is also the average number of such a tuple of indices. Similarly, for the second case, there are at most $2^{2n} \times 2^{2n}$ possible choices for the indices, which means that the average number of such triples of queries is at most 1. Markov's inequality yields

$$\Pr[N \geq 4n] \leq \frac{1 + \frac{6}{2^n}}{4n}.$$

Remark 8. Although the current bound on ℓ in the Feistel case is sufficient to prove that the attack works, we point out that the actual bound should be much sharper. Using a careful analysis of N, it should be possible to prove that $\Pr[N \geq 4n] \leq \widetilde{\mathcal{O}}\left(\frac{1}{2^n}\right)$. We support this conjecture by providing computer simulations in Table 1. As expected, the number of occurrences of the three-point structure from case 2 never grows beyond $2n$. In fact, for $n = 7$ we have an average of 0.9638 and a standard deviation of 1.0334. For $n = 8$ we have an average of 1.0034 and a standard deviation of 1.0226.

Number of solutions	n=7	n=8
0	3 982	3 743
1	3 521	3 594
2	1 660	1832
3	616	608
4	157	174
5	54	39
6	7	9
7	1	1
8	2	0

Table 1: Number of occurrences of the 3-point structure from case 2, for $n = 7, 8$, and 10 000 trials.