IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2019, No. 2, pp. 218–240. DOI:10.13154/tosc.v2019.i2.218-240

Efficient Search for Optimal Diffusion Layers of Generalized Feistel Networks

Patrick Derbez*, Pierre-Alain Fouque†,
Baptiste Lambin‡ and Victor Mollimard

Univ Rennes, The French National Centre for Scientific Research (CNRS), Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), Rennes, France
{baptiste.lambin,patrick.derbez,victor.mollimard}@irisa.fr
pierre-alain.fouque@univ-rennes1.fr

Abstract. The Feistel construction is one of the most studied ways of building block ciphers. Several generalizations were then proposed in the literature, leading to the Generalized Feistel Network, where the round function first applies a classical Feistel operation in parallel on an even number of blocks, and then a permutation is applied to this set of blocks. In 2010 at FSE, Suzaki and Minematsu studied the diffusion of such construction, raising the question of how many rounds are required so that each block of the ciphertext depends on all blocks of the plaintext. They thus gave some optimal permutations, with respect to this diffusion criteria, for a Generalized Feistel Network consisting of 2 to 16 blocks, as well as giving a good candidate for 32 blocks. Later at FSE'19, Cauchois et al. went further and were able to propose optimal even-odd permutations for up to 26 blocks.
In this paper, we complete the literature by building optimal even-odd permutations for 28, 30, 32, 36 blocks which to the best of our knowledge were unknown until now. The main idea behind our constructions and impossibility proof is a new characterization of the total diffusion of a permutation after a given number of rounds. In fact, we propose an efficient algorithm based on this new characterization which constructs all optimal even-odd permutations for the 28, 30, 32, 36 blocks cases and proves a better lower bound for the 34, 38, 40 and 42 blocks cases. In particular, we improve the 32 blocks case by exhibiting optimal even-odd permutations with diffusion round of 9. The existence of such a permutation was an open problem for almost 10 years and the best known permutation in the literature had a diffusion round of 10. Moreover, our characterization can be implemented very efficiently and allows us to easily re-find all optimal even-odd permutations for up to 26 blocks with a basic exhaustive search.

Keywords: Diffusion round · Feistel · Permutations

1 Introduction

The Feistel network is one of the main generic designs for building modern block ciphers. It was initially proposed in the data encryption standard DES [DES77], and is still used in more recent ciphers such as Twofish [SKW+98], Camellia [AIK+00] or SIMON [BSS+13]. The idea behind this construction is to split the plaintext into two halves x_0, x_1, and

*Patrick Derbez was supported by the French Agence Nationale de la Recherche through the CryptAudit project under Contract ANR-17-CE39-0003.
†Pierre-Alain was supported by the French Agence Nationale de la Recherche through the BRUTUS project under Contract ANR-14-CE28-0015.
‡Baptiste Lambin was supported by the Direction Générale de l'Armement (Pôle de Recherche CYBER).

Received: 2019-03-01, Accepted: 2019-05-01, Published: 2019-06-11

build the round function which sends (x_0, x_1) to $(x_1, x_0 \oplus F_i(x_1))$, where F_i is a non-linear function for the i-th round. One of the main advantage of this construction is that F_i does not need to be invertible, and thus it allows to transform a pseudorandom *function* (PRF) into a pseudorandom *permutation* (PRP). Moreover, there are theoretical arguments suggesting that it is a good method to construct block ciphers, as Luby and Rackoff proved in 1988 [LR88] that if each F_i is a pseudorandom function and all three are independent, then 3 rounds of the Feistel construction are enough to get a block cipher which is indistinguishable from a random permutation under the Chosen Plaintext Attack (CPA) model, and 4 rounds with 4 independent functions are enough in the Chosen Ciphertext Attack (CCA) model. This was later improved by Pieprzyk in 1990 [Pie90] : if one takes f as a pseudorandom function, 4 rounds of Feistel with $F_i = f$ for $i = 1, 2, 3$ and $F_4 = f^2$ are sufficient to obtain a block cipher that is indistinguishable from a random permutation in the CPA model. In 1989 at CRYPTO, Zheng et al. [ZMI89] proposed some generalizations of the Feistel construction. Especially, they defined the *Type-2 Feistel*[1] construction, which splits the message into $2k$ blocks and uses a round function of the form

$$(x_0, \ldots, x_{2k-1}) \mapsto (x_{2k-1}, x_0 \oplus F_{i,0}(x_1), x_1, x_2 \oplus F_{i,1}(x_3), x_3, \ldots, x_{2k-2} \oplus F_{i,k-1}(x_{2k-1})),$$

where each $F_{i,j}$ is a pseudorandom function for the i-th round. This is essentially a parallel application of k Feistels followed by a cyclic shift of the blocks. They also showed that when all $F_{i,j}$ are pseudorandom functions, then $2k + 1$ rounds of such a construction provide a block cipher that is indistinguishable from a random permutation. Moreover, the Type-2 construction is inherently easier to compute in parallel, and the corresponding decryption function is basically the same except that the functions $F_{i,j}$ are applied in reverse order, i.e. for r rounds, the first round of decryption uses the functions $F_{r,j}$. Both of these properties make this construction very efficient in practice, both on hardware and software, e.g. TWINE [SMMK12] and Simpira [GM16]. All of these arguments lead to some block ciphers based on this Type-2 Feistel construction, such as HIGHT [HSH+06] and CLEFIA [SSA+07].

At ASIACRYPT'96, Nyberg [Nyb96] studied a variant of the Type-2 Feistel construction using a different permutation than the cyclic shift, called Generalized Feistel Network. Such a construction was used to design block ciphers such as TWINE [SMMK12] and Piccolo [SIH+11]. However, Nyberg only focused on one specific permutation. Suzaki and Minematsu thus studied at FSE'10 [SM10] a more general case where the cyclic shift is replaced by any other permutation of the blocks. Their work was focused on finding permutations with the lowest *diffusion round*. The diffusion round is close to the concept of *diffusion* introduced by Shannon in 1949 [Sha49]. Essentially, a block cipher has *full diffusion* if every bit of the ciphertext depends on every bit of the plaintext. In the context of Generalized Feistel Network (GFN), [SM10] defined the diffusion round as the minimal number of rounds such that every *block* of the ciphertext depends on every *block* of the plaintext. Focusing on blocks instead of bits allows them to get rid of the precise specification of the functions $F_{i,j}$ as well as the exact size of the blocks, thus giving structural results. Especially, they tied the diffusion round of a given GFN to its resistance against Impossible Differential distinguishers [BBS99], proving that if a GFN has a diffusion round of DR, then it needs strictly more than $2DR + 1$ rounds to avoid any Impossible Differential distinguisher. Along with a lower bound on the diffusion round of a GFN of $2k$ blocks, they gave optimal permutations (w.r.t the diffusion round) for $2 \leq 2k \leq 16$. It is worthy to note that such an optimal permutation was then used to design block ciphers such as TWINE [SMMK12]. At FSE'19, Cauchois et al. went further and gave optimal permutations for $18 \leq 2k \leq 26$, as well as good candidates for $2k = 32$ (which was already found in [SM10]), as well as for $2k = 64$ and 128 using a sophisticated technique that they called *Collision-free exhaustive search*. Note that these permutations

[1] Note that some papers use the term Type-2 Generalized Feistel to denote this construction

are even-odd, i.e. the image of an even number is an odd number. On a side note, relaxing the condition that the permutation is the same in each round make the problem easier and in [KPP+17], Kales et al. give such a construction for any number of blocks.

Our contribution. In this paper, we focus on even-odd permutations and we complete the work on the 10-year-old problem (introduced by [SM10]) of finding optimal even-odd permutations for 32 blocks, as well as finding optimal even-odd permutations for 28, 30 and 36 blocks which were not given in the previous literature. To do so, we propose a new characterization of a permutation reaching full diffusion after a given number of rounds. Using this characterization, we are able to create a very efficient algorithm, which on the previously mentioned cases yields all the permutations that achieve full diffusion in 9 rounds. Note that our algorithm essentially uses branch-and-bound techniques, and thus it is hard to evaluate the exact complexity. However, the size of the search space goes from 2^{43} for $2k = 28$ up to 2^{75} for $2k = 42$, but we were able to treat each of these cases in less than one hour for each value of k when using 72 threads. Moreover, this characterization has a very efficient implementation which allowed us to re-find all optimal even-odd permutations for up to 26 blocks with a basic exhaustive search in a few hours, showing that for these cases, there is no need for sophisticated techniques as in [CGT19]. Furthermore, for 34, 38, 40 and 42 blocks, we prove with this method that there is no even-odd permutation with a diffusion round of 9, which is the lower bound on the diffusion round for these sizes given in [SM10]. We were also able to find even-odd permutations with a diffusion round of 10 for $2k = 34$ (which is thus optimal), as well as even-odd permutations with diffusion round 11 for $2k = 38, 40, 42$. Finally, we evaluate the security of our constructed permutations against impossible differentials and differentials (by computing the minimum number of active S-boxes). In particular, for the 32 blocks case, and the impossible differentials, all our permutations have a one-round shorter longest impossible differential distinguisher compared to what was proposed by [CGT19], which brings it down to 17 rounds.

2 Preliminaries

2.1 Generalized Feistel Networks (GFN)

Zheng et al. [ZMI89] introduced Type-2 Feistels as a generalization of the original Feistel construction. Given an even number $2k$ of blocks $(X_0, \ldots, X_{2k-1})$, it first applies the Feistel construction on the pairs of blocks which yields $(X_0 \oplus S_0(X_1),\ X_1, \ldots,\ X_{2k-2} \oplus S_{k-1}(X_{2k-1}), X_{2k-1})$. The blocks are then cyclically right shifted to obtain the result. Later, it was proposed to use another permutation than the cyclic shift in [Nyb96], leading to Generalized Feistel Networks.

Definition 1. Let $2k$ be an even number, n, r be positive integers, and $\{F_{i,j}\}_{i\in\{1,\ldots,r\},j\in\{0,\ldots,k-1\}}$ be a set of cryptographic keyed functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Let π be a permutation over $2k$ elements. A Generalized Feistel Network (GFN) is a block cipher built as $\mathcal{R}_r \circ \cdots \circ \mathcal{R}_1$, where $\mathcal{R}_i$ is the round function

$$\mathcal{R}_i :\ (X_0, \ldots, X_{2k-1}) \to \pi(X_0 \oplus F_{i,0}(X_1),\ X_1, \ldots,\ X_{2k-2} \oplus F_{i,k-1}(X_{2k-1}), X_{2k-1})$$

Note that for this paper, neither the exact definition of the keyed functions $F_{i,j}$ nor their sizes are relevant. We can thus consider all of them as an arbitrary S-box S, leading to the framework depicted in Figure 1[2]. As the only variable parameters are thus k and π, we denote by GFN_π^k a GFN with $2k$ blocks that uses the permutation π.

[2]In practice, one should carefully study the primitive if the same F-function is used, e.g. [GM16]

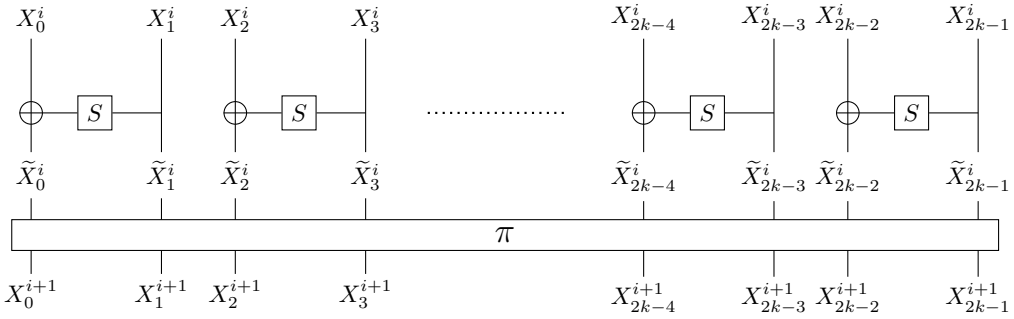

Figure 1: Generalized Feistel Network

2.2 Diffusion Round

We use the notations depicted in Figure 1. The input variables of the i-th round of a GFN are denoted by $(X_0^i, X_1^i, \ldots, X_{2k-1}^i)$. We also denote by $(\widetilde{X}_0^i, \widetilde{X}_1^i, \ldots, \widetilde{X}_{2k-1}^i)$ the variables which are at the input of the permutation π, i.e.

$$(X_0^{i+1}, X_1^{i+1}, \ldots, X_{2k-1}^{i+1}) = \pi(\widetilde{X}_0^i, \widetilde{X}_1^i, \ldots, \widetilde{X}_{2k-1}^i)$$

It is easy to see from Definition 1 that $X_{\pi(0)}^1$ depends on X_0^0 and X_1^0. More generally, any block $\widetilde{X}_j^r$ depends on a certain number of blocks from the round 0, i.e. computing $\widetilde{X}_j^r$ requires some blocks $\{X_{j_0}^0, \ldots, X_{j_l}^0\}$. Note that this does not depend on the size of the functions $F_{i,j}$ in the GFN. As in [SM10], we say in that case that any of these $X_{j_i}^0$ *diffuses* to $\widetilde{X}_j^r$, and we focus our study on the number of rounds needed to reach *full diffusion*.

Definition 2. Let π be a permutation over $2k$ elements. We say that a block X_j^0 fully diffuses after r rounds if for all $i \in \{0, \ldots, 2k-1\}$, X_j^0 diffuses to $\widetilde{X}_i^r$. We say that π reaches full diffusion after r rounds if for all $j \in \{0, \ldots, 2k-1\}$, X_j^0 fully diffuses after r rounds. The smallest r that verifies this property for the block X_i^0 is called the diffusion round of the block X_i^0.

Note that we need to study both the diffusion over the encryption *and* the decryption process. Indeed, there is no guarantee that an encryption function with good diffusion also keeps this property for its inverse. Since we have $(GFN_\pi^k)^{-1} = GFN_{\pi^{-1}}^k$, we need to study both the diffusion of π and π^{-1}. Naturally, we would like both π and π^{-1} to fully diffuse as quickly as possible, which leads to the following definition.

Definition 3. Let π be a permutation over $2k$ elements. Denote by $DR_i(\pi)$ the minimum number of rounds r such that X_i^0 fully diffuses after r rounds in GFN_π^k.

The diffusion round of a permutation π is:

$$DR_{max}(\pi) = \max_{0 \le i \le 2k-1} \left\{ DR_i(\pi), DR_i(\pi^{-1}) \right\} \tag{1}$$

This definition gives the same importance to the total diffusion of both π and π^{-1}. Definition 3 defines a natural partial order on the permutations: a permutation π_1 is better (at diffusing) than a permutation π_2 if $DR_{max}(\pi_1) \le DR_{max}(\pi_2)$. Searching the best permutations (for the diffusion) directly can be difficult. As a result the methodology we adopt in this work is to search for permutations that diffuse totally in the forward direction and then check if their respective inverse also diffuses totally.

2.3 Even-odd Permutations

A naive way to search for optimal permutation would be to simply go through all of them and check the diffusion one permutation by one. However, there are $(2k)!$ permutations, which quickly grows beyond practical means. For example with $2k = 32$, approximately 2^{117} permutations should be checked. To reduce the number of permutations that will be tested, we will restrict ourselves to a specific class of permutations and give an equivalence relation which further reduces the number of permutations to be considered.

In [SM10], Suzaki and Minematsu did an exhaustive search for $1 \leq k \leq 8$, and made the observation that every optimal permutation (for such k) mapped even-number input blocks to odd-number output blocks and vice versa. We call such permutations even-odd. In the rest of this paper, we will use the following notation for even-odd permutations. An even-odd permutation π of size $2k$ will be denoted by the pair of permutations (p, q) of size k verifying $\forall i \in [0, k-1],\ \pi(2i) = 2 \cdot p(i) + 1$ and $\pi(2i+1) = 2 \cdot q(i)$. The search space is now reduced to $(k!)^2$ permutations.

According to this, [SM10] gives the following lower-bound on the diffusion round of even-odd permutations (p, q).

Proposition 1. *Let $\mathcal{F}_i$ be the Fibonacci sequence, i.e. $\mathcal{F}_0 = 0, \mathcal{F}_1 = 1$ and $\mathcal{F}_i = \mathcal{F}_{i-1} + \mathcal{F}_{i-2}, i \geq 2$. Let $\pi = (p, q)$ be an even-odd permutation over $2k$ elements, and i be the smallest integer such that $\mathcal{F}_i \geq k$. Then $DR_{max}(\pi) \geq i + 1$.*

For a given permutation π, if the inequality is tight, we say that π is tight. A proof of this proposition already exists in both [SM10] and [CGT19]. According to our results, we will give another proof of this proposition in Section 3. We will also show in Section 3 that this bound is tight for the cases $2k = 28, 30, 32, 36$ and strict for $2k = 34, 38, 40, 42$.

2.4 Equivalence Classes of Even-odd Permutations

To further reduce the size of the search space, as in [CGT19], we use some equivalence classes, given by the following definition.

Definition 4. Let π and π' be two even-odd permutations over $2k$ elements. We say that π and π' are equivalent if there exists a permutation φ over $2k$ elements such that

$$\pi' = \varphi \circ \pi \circ \varphi^{-1}.$$

From [CGT19], we can then give a set of permutations $\mathbb{P}_k$ such that for any equivalence class, there exists at least one $\pi \in \mathbb{P}_k$ which belongs to this class. This effectively gives us a set of class representatives (in which a few of them are redundant), and this set can be built from the following proposition, proven in [CGT19]. Recall that any permutation can be decomposed into a composition of cycles. We call *cycle structure* the unordered set of the length of these cycles, for example the permutation

$$(0\ 1\ 2\ 3)(4\ 5)(6\ 7)(8)$$

has a cycle structure of $\{4, 2, 2, 1\}$.

Proposition 2. *Let $\mathbb{P}_k$ be a set of even-odd permutations $\pi = (p, q)$ over $2k$ elements constructed as follows. For each possible cycle structure c of a permutation over k elements, pick one permutation p which has a cycle structure equal to c. Then, for every permutation q over k elements, add (p, q) in the set $\mathbb{P}_k$. By doing so, $\mathbb{P}_k$ contains at least one representative of each equivalence class induced by Definition 4. Moreover, $\mathbb{P}_k$ contains exactly $\mathcal{N}_k.k!$ elements, where $\mathcal{N}_k$ is the number of partitions of the integer k.*

This allows us to only consider $\mathcal{N}_k.k!$ permutations instead of $(k!)^2$. This is a significant improvement, as for example with $k = 16$, there are only $231 \times 16! \simeq 2^{52}$ permutations to go through, instead of $(16!)^2 \simeq 2^{88}$. However when k grows, it is still too big a number to try an exhaustive search. As such, we propose in Section 4 an efficient search algorithm to find all optimal even-odd permutations for a given k, without needing to do an exhaustive search.

3 Characterization of Full Diffusion

In this section, we will explain our strategy to search for a tight even-odd permutation, that is, a permutation with a diffusion round reaching the Fibonacci bound given in Proposition 1. We will first give an algebraic characterization for a permutation to have full diffusion, then give an algorithm to exploit this characterization and quickly search all such permutations. Note that here we only focus on the diffusion round of the permutation when considering encryption. That is, for a given permutation π, we focus only on $DR(\pi) = \max_{0 \leq i \leq 2k-1} \{DR_i(\pi)\}$. Then, once we found a permutation reaching the Fibonacci bound, we can easily check if π^{-1} also reaches this bound, and if that is the case, we found a tight permutation.

We describe here the main tools we used to design our search algorithm. Note that for two permutations p, q, we denote the composition $p \circ q$ by pq for better reading. We first begin by giving the following proposition.

Proposition 3. *Let $\pi = (p, q)$ be an even-odd permutation over $2k$ elements. Then π achieves full diffusion after r rounds if and only if each block X_j^0 is diffused to at least one block of each pair at the input of the $(r-1)$-th round, i.e. diffused to either $X_{2j'}^{r-1}$ or $X_{2j'+1}^{r-1}$ for each $j' \in \{0, \ldots, k-1\}$.*

Proof. Suppose that a given block X_i^0 has been fully diffused, i.e. to every block $\widetilde{X}_{2j}^r$ and $\widetilde{X}_{2j+1}^r, j \in \{0, \ldots, k-1\}$. Then X_i^0 must have diffused to at least X_{2j+1}^r for every j, as it is the only way to reach $\widetilde{X}_{2j+1}^r$. Thus, X_i^0 must have diffused to $\widetilde{X}_{2j'}^{r-1}$ with $j' = p^{-1}(j)$, which means that it has diffused to either $X_{2j'}^{r-1}$ or $X_{2j'+1}^{r-1}$.

On the other hand, suppose that a given block X_i^0 has diffused to an even block X_{2j}^{r-1}, then X_i^0 will be diffused to only $\widetilde{X}_{2j}^{r-1}$. If X_i^0 has diffused to an odd block X_{2j+1}^{r-1}, it will be diffused to both $\widetilde{X}_{2j}^{r-1}$ and $\widetilde{X}_{2j+1}^{r-1}$. In both cases, it will be diffused to $\widetilde{X}_{2j}^{r-1}$, then to $X_{2j'+1}^r$ with $j' = p(j)$, and finally to both $\widetilde{X}_{2j'}^r$ and $\widetilde{X}_{2j'+1}^r$. Thus, if for all $j \in \{0, \ldots, k-1\}$, X_i^0 is diffused to any block of the j-th pair at the input of the $(r-1)$-th round, it will be diffused to every block $\widetilde{X}_{2p(j)}^r$ and $\widetilde{X}_{2p(j)+1}^r$, and since p is a permutation, this means that we have full diffusion for X_i^0. □

Corollary 1. *Let $\pi = (p, q)$ be an even-odd permutation over $2k$ elements. Then π achieves full diffusion after r rounds if and only if each even block $X_{2j}^0, j \in \{0, \ldots, k-1\}$ diffuses to every even block $\widetilde{X}_{2j'}^{r-1}, j' \in \{0, \ldots, k-1\}$.*

Proof. For the proof of the previous theorem, we can easily see that a block X_j^0 diffuses to either $X_{2j'}^{r-1}$ or $X_{2j'+1}^{r-1}$ if and only if X_j^0 diffuses to $\widetilde{X}_{2j'}^{r-1}$. Moreover, we can easily see that if X_{2j}^0 is fully diffused, so is X_{2j+1}^0. Indeed, X_{2j}^0 being fully diffused is the same as $\widetilde{X}_{2j}^0$ being fully diffused, and X_{2j+1}^0 is always diffused to $\widetilde{X}_{2j}^0$. □

Thus we only need to focus on the diffusion of each block X_{2j}^0 to each block $\widetilde{X}_{2j}^{r-1}$. Now we can take a look a what would happen in an ideal scenario. Assume that we are

studying the diffusion of a block X^0_{2j}. Then X^0_{2j} is diffused to $\widetilde{X}^0_{2j^1_0}$ with $j^1_0 = j$. It is then diffused to both $\widetilde{X}^1_{2j^2_0}$ and $\widetilde{X}^1_{2j^2_0+1}$, with $j^2_0 = p(j^1_0)$. Then again :

- $\widetilde{X}^1_{2j^2_0}$ is diffused to both $\widetilde{X}^2_{2j^3_0}$ and $\widetilde{X}^2_{2j^3_0+1}$, with $j^3_0 = p(j^2_0)$.
- $\widetilde{X}^1_{2j^2_0+1}$ is diffused to $\widetilde{X}^2_{2j^3_1}$ with $j^3_1 = q(j^2_0)$

Assuming an ideal scenario, we would have $j^3_0 \neq j^3_1$, i.e. X^0_{2j} has diffused to two different blocks after 4 rounds (minus the application of π on the fourth round). We can then keep going and get a series of j^i_ℓ which gives us the blocks on which X^0_{2j} has diffused after $i+1$ rounds minus the last application of π, always assuming that we never have $j^i_\ell = j^i_{\ell'}$ for $\ell \neq \ell'$. The propagation for up to 7 rounds is given in Figure 2 at the end of this section.

However, we cannot have $j^i_\ell \neq j^i_{\ell'}$ with $\ell \neq \ell'$ forever. Indeed, since we only have k blocks, we are bound at some point to have $j^i_\ell = j^i_{\ell'}$ and $\ell \neq \ell'$. However, we can easily compute the actual value of each j^i_ℓ. Indeed, if we take for example j^6_6 in Figure 2, then we know that

$$j^6_6 = (qppqp)(j^1_0) = (qppqp)(j).$$

Denote by $\mathbb{J}^i_j$ the set of equations obtained by expressing every j^i_ℓ that way. For example, we would have

$$\begin{aligned}\mathbb{J}^6_j = \{&(ppppp)(j),\\ &(qpppp)(j),\\ &(pqppp)(j),\\ &(ppqpp)(j),\\ &(qpqpp)(j),\\ &(pppqp)(j),\\ &(qppqp)(j),\\ &(pqpqp)(j)\}\end{aligned}$$

According to this, we can give a generic way to compute $\mathbb{J}^i_j$. We start with $\mathbb{J}^1_j = \{j\}$ and $\mathbb{J}^2_j = \{p(j)\}$. To build $\mathbb{J}^i_j$ from $\mathbb{J}^{i-1}_j$, we begin by adding $p(x)$ to $\mathbb{J}^i_j$ for every term x in $\mathbb{J}^{i-1}_j$. Then, for every term x in $\mathbb{J}^{i-1}_j$ such that x can be written as $x = p(y)$ for some $y \in \mathbb{J}^{i-2}_j$, we also add $q(x)$ to $\mathbb{J}^i_j$.

We can justify this construction as follows. Suppose that a given j' belongs to $\mathbb{J}^{i-2}_j$ because X^0_{2j} diffuses to $\widetilde{X}^{i-2}_{2j'}$. Then X^0_{2j} diffuses to both $\widetilde{X}^{i-1}_{2j''}$ and $\widetilde{X}^{i-1}_{2j''+1}$ with $j'' = p(j')$. Thus for the next round, X^0_{2j} will diffuse to both $X^i_{2\widetilde{j}+1}$ and $X^i_{2\widetilde{j'}}$, with $\widetilde{j} = p(j'')$ and $\widetilde{j'} = q(j'')$.

On the other hand, suppose that j' belongs to $\mathbb{J}^{i-2}_j$ because X^0_{2j} diffuses to $\widetilde{X}^{i-2}_{2j'+1}$. In that case, X^0_{2j} will only diffuse to $\widetilde{X}^{i-1}_{2j''}$ with $j'' = q(j')$. For the next round, X^0_{2j} only diffuses to $X^i_{2\widetilde{j}+1}$ with $\widetilde{j} = p(j'')$.

Thus in both cases, we need to have $\widetilde{j} = p(j'')$, but we only require $\widetilde{j'} = q(j'')$ in the first case, which corresponds exactly to the case where the previous term started with a composition by p.

Note that from this construction, we can deduce the following proposition.

Proposition 4. *The size of $\mathbb{J}^i_j$ is exactly $\mathcal{F}_i$ where $\mathcal{F}_i$ is the i-th term of the Fibonacci sequence.*

Proof. We can prove this by induction. Both $\mathbb{J}_j^1$ and $\mathbb{J}_j^2$ are of size 1, which corresponds to $\mathcal{F}_1$ and $\mathcal{F}_2$. We first add an element $p(x)$ to $\mathbb{J}_j^i$ for every $x \in \mathbb{J}_j^{i-1}$, thus $\mathcal{F}_{i-1}$ elements. Then, for every x in $\mathbb{J}_j^{i-1}$ such that $x = p(y)$ with $y \in \mathbb{J}_j^{i-2}$, we add $q(x)$ to $\mathbb{J}_j^i$. However, according to our construction, $\mathbb{J}_j^{i-1}$ contains such an element $x = p(y)$ for *every* term $y \in \mathbb{J}_j^{i-2}$. Thus, there are $\mathcal{F}_{i-2}$ such terms. In the end, $\mathbb{J}_j^i$ contains $\mathcal{F}_{i-1} + \mathcal{F}_{i-2} = \mathcal{F}_i$ elements, which concludes the induction. □

We can now use those sets $\mathbb{J}_j^i$ to fully characterize the fact that a block fully diffuses when using a given permutation.

Theorem 1. *Let $\mathbb{J}_j^{r-1}$ be the set of equations as defined above. Then for a given permutation $\pi = (p, q)$ over $2k$ blocks, X_{2j}^0 is fully diffused after r rounds if and only if $\mathbb{J}_j^{r-1}$ contains every number in $\{0, \ldots, k-1\}$ at least once.*

Proof. As $\mathbb{J}_j^{r-1} = \{j_0^{r-1}, \ldots, j_{\ell_{r-1}}^{r-1}\}$ is defined, it basically represents that X_{2j}^0 diffuses to every $\widetilde{X}_{2j_\ell^i}^i$. Thus, if $\mathbb{J}_j^i$ contains every number in $\{0, \ldots, k-1\}$ at least once, it exactly means that X_{2j}^0 diffuses to each block $\widetilde{X}_{2j'}^i, j' \in \{0, \ldots, k-1\}$. According to Corollary 1, this means that X_{2j}^0 achieves full diffusion after $i+1$ rounds. □

We can then easily deduce the following corollary.

Corollary 2. *Let $\pi = (p, q)$ be a permutation over $2k$ elements. Then we have $DR(\pi) = i + 1$ if and only if i is the smallest integer such that for every $j \in \{0, \ldots, k-1\}$, $\mathbb{J}_j^i$ contains every number in $\{0, \ldots, k-1\}$ at least once.*

This gives us another proof for the Fibonacci bound given in Proposition 1. Indeed, for $\mathbb{J}_j^i$ to contain every number in $\{0, \ldots, k-1\}$ at least once, $\mathbb{J}_j^i$ must contain at least k terms. Thus, and since the size of $\mathbb{J}_j^i$ does not depend on j, the minimal number of rounds needed to have full diffusion for every block must be such that $\left|\mathbb{J}_j^i\right| = \mathcal{F}_i \geq k$. According to the previous corollary, if i is the smallest integer such that $\mathcal{F}_i \geq k$, this exactly means that $DR_{max}(\pi) \geq i + 1$

Note that from the construction of any $\mathbb{J}_j^i$, each term starts with a composition by p. Since p is a permutation, and we want full diffusion for every blocks, we can remove this first p from every term to get a smaller representation. Essentially, this means that we are considering the diffusion of the block $p^{-1}(j)$, but we will still write $\mathbb{J}_j^i$. As such, $\mathbb{J}_j^6$ for example is thus rewritten as

$$\begin{aligned}\mathbb{J}_j^6 = \{&(p^4)(j),\\ &(qp^3)(j),\\ &(pqp^2)(j),\\ &(p^2qp)(j),\\ &(qpqp)(j),\\ &(p^3q)(j),\\ &(qp^2q)(j),\\ &(pqpq)(j)\}\end{aligned}$$

To illustrate the previous characterization, we introduce what we call the *diffusion table (of rank i)* of an even-odd permutation (p, q) of size $2k$. The columns are indexed by the numbers from 0 to $k-1$ and the row are indexed by the products of p and q used to generate all sets $\mathbb{J}_j^i$. Each cell of the table is the value obtained by applying the permutation indexing the row to the value indexing the column of the cell. For example,

Table 1: Diffusion tables for the cyclical shift (left table) and one optimal permutation proposed by [CGT19] (right table).

x	0	1	2	3	4	5	6	7
p^5	3	4	5	6	7	0	1	2
p^4q	4	5	6	7	0	1	2	3
p^3qp	4	5	6	7	0	1	2	3
p^2qp^2	4	5	6	7	0	1	2	3
pqp^3	4	5	6	7	0	1	2	3
qp^4	4	5	6	7	0	1	2	3
p^2qpq	5	6	7	0	1	2	3	4
pqp^2q	5	6	7	0	1	2	3	4
qp^3q	5	6	7	0	1	2	3	4
$pqpqp$	5	6	7	0	1	2	3	4
qp^2qp	5	6	7	0	1	2	3	4
$qpqp^2$	5	6	7	0	1	2	3	4
$qpqpq$	6	7	0	1	2	3	4	5
diff	4	4	4	4	4	4	4	4

x	0	1	2	3	4	5	6	7
p^5	4	3	5	1	6	7	0	2
p^4q	3	2	1	4	0	6	7	5
p^3qp	2	6	7	5	1	3	4	0
p^2qp^2	6	7	4	0	5	2	3	1
pqp^3	1	4	3	2	0	6	7	5
qp^4	2	5	7	6	3	1	4	0
p^2qpq	7	1	0	6	3	5	2	4
pqp^2q	4	5	2	1	7	0	6	3
qp^3q	5	0	6	2	4	3	1	7
$pqpqp$	5	0	6	3	2	4	1	7
qp^2qp	0	3	1	7	6	5	2	4
$qpqp^2$	3	1	2	4	7	0	5	6
$qpqpq$	1	6	4	3	5	7	0	2
diff	8	8	8	8	8	8	8	8

the cell indexed by p^i and 0 contains $p^i(0)$. This provides a clear visualization of our characterization, as the j-th column is exactly $\mathbb{J}_j^i$.

Thus, we can easily illustrate Corollary 2 by verifying that every column of this table contains every possible values. We thus add one more row at the end of diffusion table called *diff* which contains the number of different values in a column. By construction, this is exactly the number of elements of $\mathbb{J}_j^i$ where j is the index of the column. In tables constructed as described, the full diffusion of a permutation corresponds to a *diff* row containing only the value k.

For example, we give in Table 1 the diffusion tables for the cyclical shift (i.e. $p = (7, 0, 1, 2, 3, 4, 5, 6)$ and $q = (0, 1, 2, 3, 4, 5, 6, 7)$) and one of the optimal permutation proposed by [CGT19] (i.e. $p = (6, 3, 7, 1, 0, 2, 4, 5)$ and $q = (3, 5, 1, 6, 4, 0, 2, 7)$) for $k = 8$ and $i = 7$, thus the optimal permutation clearly have a diffusion round of 8.

Finally, we can reformulate the problem of finding optimal even-odd permutations with these tables. Indeed, it corresponds to finding the minimal i and even-odd permutations of size $2k$ such that their diffusion table have their *diff* row containing only k.

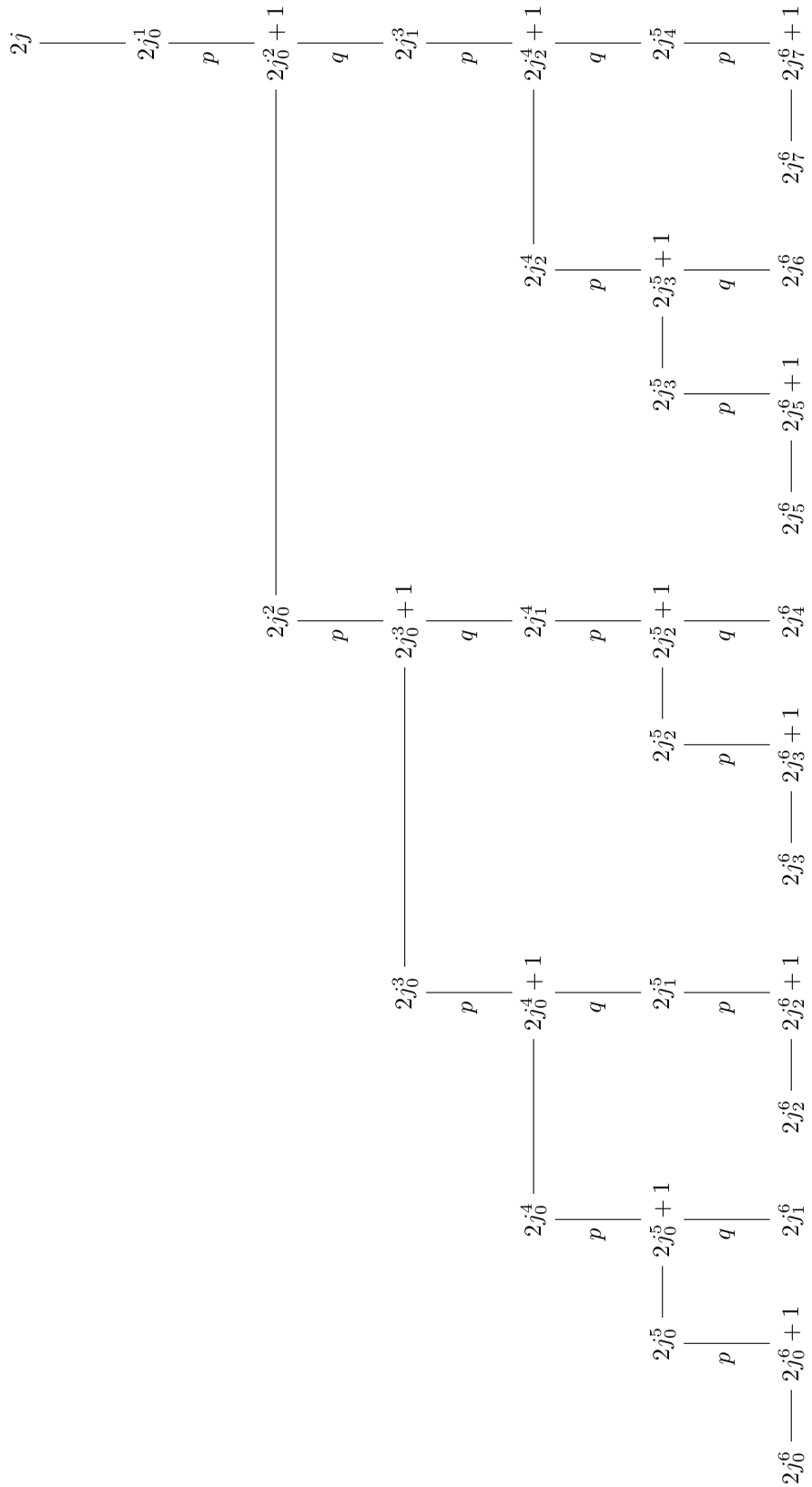

Figure 2: Propagation tree for 7 rounds (minus the last application of π)

4 Searching for an Optimal Permutation over 9 Rounds

4.1 Efficient Search Algorithm

First, we can see that our characterization can be very efficiently implemented, as testing if $\pi = (p, q)$ has full diffusion mostly requires only a few table lookups. An example of an implementation for this test for 9 rounds is given in Appendix B, and its efficiency allowed us to recover all optimal even-odd permutations for $k \leq 13$ with a basic exhaustive search. Especially, for $k = 13$, we were able to go through all $\mathcal{N}_{13}.13! \simeq 2^{39}$ permutations and check them in about 410 minutes on a single core. While these optimal permutations were already known, it shows that the sophisticated techniques introduced in [CGT19] were not necessary for these cases.

However for $k \geq 14$, it becomes too expensive to make this exhaustive search. We thus focus on finding optimal even-odd permutations for $14 \leq k \leq 21$, hence such permutations would have a diffusion round of 9. Given a cycle structure for p, we can easily find a permutation p with such structure and thus we need to search q such that $\pi = (p, q)$ needs 9 rounds to reach full diffusion, i.e., such that each $\mathbb{J}_j^8$ contains all numbers from 0 to $k-1$.

Note that we cannot exploit $\mathbb{J}_j^8$ directly. Indeed, one might want to guess parts of q and check if $\mathbb{J}_j^8$ does not contains too many duplicates. However, to fully compute $\mathbb{J}_j^8$, we need to guess q in its entirety, which makes this strategy too expensive. We thus describe an efficient way to exploit this characterization to find optimal even-odd permutations.

First for a given j, if we take a look at $\mathbb{J}_j^6$, we can see that we need to make only 7 guesses over the images of q to fully compute $\mathbb{J}_j^6$. Indeed, we need to know

$$q(j), (qp)(j), (qp^2)(j), (qp^3)(j), (qpq)(j), (qp^2q)(j) \text{ and } (qpqp)(j).$$

Let $\mathbb{X}_j^6$ and $\mathbb{Y}_j^6$ be two subsets of $\mathbb{J}_j^6$, such that $\mathbb{X}_j^6 \cup \mathbb{Y}_j^6 = \mathbb{J}_j^6$, with

$$\mathbb{X}_j^6 = \{p^4(j), (pqp^2)(j), (p^2qp)(j), (p^3q)(j), (pqpq)(j)\}$$

$$\text{and } \mathbb{Y}_j^6 = \{(qp^3)(j), (qpqp)(j), (qp^2q)(j)\}.$$

According to the construction of $\mathbb{J}_j^8$, we can actually write

$$\mathbb{J}_j^8 = p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \cup (pq)(\mathbb{X}_j^6) \cup (qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6).$$

Assume that we made the 7 guesses mentioned above. In that case, we know the exact values in both $\mathbb{X}_j^6$ and $\mathbb{Y}_j^6$. Moreover, since p is known, we know exactly the values in $p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$. Finally, since we guessed 7 images of q, there might be some values in $(pq)(\mathbb{X}_j^6)$ and $(qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$ that are known.

Hence, we create three sets $\mathbb{K}_j, \widetilde{\mathbb{X}}_j^6$ and $\widetilde{\mathbb{Y}}_j^6$:

- $\mathbb{K}_j$ is the set of all known values of $\mathbb{J}_j^8$. Thus $p^2(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6) \subset \mathbb{K}_j$ and there might be a few elements from $(pq)(\mathbb{X}_j^6)$ and $(qp)(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$ in $\mathbb{K}_j$ too.
- $\widetilde{\mathbb{X}}_j^6$ is the subset of $\mathbb{X}_j^6$ such that for any $x \in \widetilde{\mathbb{X}}_j^6$, the value of $q(x)$ yet remains to be determined.
- In the same way, $\widetilde{\mathbb{Y}}_j^6$ is the subset of $p(\mathbb{X}_j^6 \cup \mathbb{Y}_j^6)$ such that for any $x \in \widetilde{\mathbb{Y}}_j^6$, the value of $q(x)$ is not determined.

For j to be fully diffused, we thus have the constraint

$$C_j : \left| \mathbb{K}_j \cup q(\widetilde{\mathbb{Y}}_j^6) \cup (pq)(\widetilde{\mathbb{X}}_j^6) \right| \geq k.$$

We then check if this constraint is valid, i.e. if there exist some guesses for the remaining images of q such that C_j holds, and this is described in the next section.

Now if we take a look at $\mathbb{J}^6_{j'}$ where $j' = p(j)$, we can see that we only need 3 more guesses to compute it, instead of 7 as before. Indeed, we already guessed

$$\begin{aligned}(qp)(j) &= q(j')\\ (qp^2)(j) &= (qp)(j')\\ (qp^3)(j) &= (qp^2)(j')\\ (qpqp)(j) &= (qpq)(j')\end{aligned}$$

and thus it only remains to guess

$$\begin{aligned}(qp^4)(j) &= (qp^3)(j')\\ (qp^2qp)(j) &= (qp^2q)(j')\\ (qpqp^2)(j) &= (qpqp)(j').\end{aligned}$$

By doing these guesses, we can build the sets $\mathbb{K}_{j'}, \mathbb{X}^6_{j'}$ and $\mathbb{Y}^6_{j'}$ as before, and thus get another constraint that needs to be checked

$$C_{j'} : \left|\mathbb{K}_{j'} \cup q(\widetilde{\mathbb{Y}}^6_{j'}) \cup (pq)(\widetilde{\mathbb{X}}^6_{j'})\right| \geq k.$$

However by making those three new guesses, we might be able to compute new values in $\widetilde{\mathbb{X}}^6_j$ and $\widetilde{\mathbb{Y}}^6_j$. We thus need to update the constraint C_j according to these guesses, and then check again if C_j is valid.

This can be repeated until we have fully guessed q, in which case we have a solution, or show that no matter which guesses we made there is no solution which satisfies all constraints. This is the core of our algorithm, which is described from a high-level point of view in Algorithm 1.

Note however that the actual algorithm is a bit more sophisticated. Indeed, it might occur at some point that $p(j)$ was already processed, i.e. $C_{p(j)}$ is already a constraint we have. When this happens, we need to choose another starting block j, and re-apply the algorithm, while still keeping all previously computed constraints. In practice, we found that the most efficient strategy is to use an element from the shortest cycle of p as the first starting block. Then, if we need to choose another starting block, we pick an element in the next shortest cycle of p and so on. Moreover, when making some guesses for the images of q, it might happen that we already made this guess. This is not a problem, as this guess basically becomes free and does not add any more cost. Finally, except for the first seven guesses, we update and check all constraints after *each* guess.

4.2 Checking the Constraints

We first give a naive way to check if a constraint is valid. We are given three sets $\mathbb{K}, \mathbb{X}$ and $\mathbb{Y}$, resulting in the constraint

$$C : |\mathbb{K} \cup q(\mathbb{Y}) \cup (pq)(\mathbb{X})| \geq k.$$

We know the full permutation p, and for any $x \in \mathbb{X} \cup \mathbb{Y}$, $q(x)$ is still unknown. Let $\mathbb{A}$ denote the set of values a for which we still do not know the preimage of a through q, i.e. for any $a \in \mathbb{A}$, we do not know which x results in $q(x) = a$. Considering the guesses we already made on q, we always know this set $\mathbb{A}$, and thus have the following two relations $(pq)(\mathbb{X}) \subset p(\mathbb{A})$ and $q(\mathbb{Y}) \subset \mathbb{A}$. According to this, we can write

$$|\mathbb{K} \cup q(\mathbb{Y}) \cup (pq)(\mathbb{X})| \leq |\mathbb{K} \cup \mathbb{A} \cup p(\mathbb{A})|\,.$$

Algorithm 1 Searching for optimal even-odd permutations over 9 rounds

1.1

```
1: function nextGuess(p, q, j, ℂ)                    ℂ is the list of known constraints
2:     if q is fully determined then
3:         Print p, q
4:     else
5:         while all guesses are not made do
6:             Guess (qp^3)(j), (qp^2q)(j) and (qpqp)(j)
7:             Update every constraints in ℂ according to those guesses
8:             Deduce the new constraint C_j
9:             ℂ' ← ℂ ∪ {C_j}
10:            if ∃ invalid constraint in ℂ' then
11:                Make a new guess
12:            else
13:                nextGuess(p, q, p(j), ℂ')
14:            end if
15:        end while
16:    end if
17: end function

18: p ← chosen permutation with a given structure cycle
19: j ← an element from the smallest cycle of p
20: while all guesses are not made do
21:     Guess q(j), (qp)(j), (qp^2)(j), (qp^3)(j), (qpq)(j), (qp^2q)(j) and (qpqp)(j)
22:     Deduce the constraint C_j
23:     if C_j is a valid constraint then
24:         ℂ ← {C_j}
25:         nextGuess(p, q, p(j), ℂ)
26:     end if
27: end while
```

Hence if $|\mathbb{K} \cup \mathbb{A} \cup p(\mathbb{A})| < k$, we know that the constraint C cannot be valid. However, we can actually go further and get more precise information by doing the following.

We can formulate our problem in the following generic way. We are given three sets $\mathbb{K}, \mathbb{A}$, and $\mathbb{B}$ $(= p(\mathbb{A}))$, and we search for two sets $\widetilde{\mathbb{A}} \subset \mathbb{A}$ and $\widetilde{\mathbb{B}} \subset \mathbb{B}$ such that $|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}|$ is maximal, with $\widetilde{\mathbb{A}} = q(\mathbb{Y})$ and $\widetilde{\mathbb{B}} = (pq)(\mathbb{X})$. Note that, since p and q are permutations, we have $|\widetilde{\mathbb{A}}| = |\mathbb{X}|$ and $|\widetilde{\mathbb{B}}| = |\mathbb{Y}|$. Hence our idea is to determine whether there is at least one such pair $(\widetilde{\mathbb{A}}, \widetilde{\mathbb{B}})$ satisfying $|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}| \geq k$. Indeed if no such pair exists then constraint C does not hold. Note that if $\mathbb{X} \cap \mathbb{Y} \neq \emptyset$ then it is possible for such pair to exist while C does not hold. However we found this filter powerful enough for our need.

We can partition $\mathbb{K} \cup \mathbb{A} \cup \mathbb{B}$ into the following eight disjoint sets:

$$\begin{array}{ll} \mathbb{S}_0 = \mathbb{K} \cap \mathbb{A} \cap \mathbb{B} & \mathbb{S}_1 = \mathbb{K}^c \cap \mathbb{A} \cap \mathbb{B} \\ \mathbb{S}_2 = \mathbb{K} \cap \mathbb{A}^c \cap \mathbb{B} & \mathbb{S}_3 = \mathbb{K} \cap \mathbb{A} \cap \mathbb{B}^c \\ \mathbb{S}_4 = \mathbb{K}^c \cap \mathbb{A}^c \cap \mathbb{B} & \mathbb{S}_5 = \mathbb{K}^c \cap \mathbb{A} \cap \mathbb{B}^c \\ \mathbb{S}_6 = \mathbb{K} \cap \mathbb{A}^c \cap \mathbb{B}^c & \mathbb{S}_7 = \mathbb{K}^c \cap \mathbb{A}^c \cap \mathbb{B}^c \end{array}$$

Let k_A (resp. k_B) denote the cardinality of $\widetilde{\mathbb{A}}$ (resp. $\widetilde{\mathbb{B}}$), and k_A^i, k_B^i be such that

$$k_A^i = |\widetilde{\mathbb{A}} \cap \mathbb{S}_i| \leq \min(|\mathbb{S}_i|, k_A), \quad k_B^i = |\widetilde{\mathbb{B}} \cap \mathbb{S}_i| \leq \min(|\mathbb{S}_i|, k_B).$$

Since all $\mathbb{S}_i$ are disjoint, $\widetilde{\mathbb{A}} \subset \mathbb{A}$ and $\widetilde{\mathbb{B}} \subset \mathbb{B}$, notice that we have

$$\begin{aligned} k_A^2 = k_A^4 = k_A^6 = k_A^7 = 0 \text{ and } k_A = k_A^0 + k_A^1 + k_A^3 + k_A^5 \\ k_B^3 = k_B^5 = k_B^6 = k_B^7 = 0 \text{ and } k_B = k_B^0 + k_B^1 + k_B^2 + k_B^4. \end{aligned}$$

By selecting the two sets $\widetilde{\mathbb{A}} \cap \mathbb{S}_1$ and $\widetilde{\mathbb{B}} \cap \mathbb{S}_1$ as disjoint as possible we have:

$$\begin{aligned} |\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}| = |K| + k_A + k_B - k_A^0 - k_B^0 - k_B^2 - k_A^3 \\ - \max(k_A^1 + k_B^1 - |\mathbb{S}_1|, 0) \end{aligned}$$

Indeed, first we have at most $|K| + k_A + k_B$ elements in $\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}$. However among all those elements, some might be the same, which explains the remaining terms :

- Elements of $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$ included in $\mathbb{S}_0$, $\mathbb{S}_2$ or $\mathbb{S}_3$ are duplicates since they all belong to $\mathbb{K}$.
- We need to take k_A^1 (resp. k_B^1) elements from $\mathbb{A}$ (resp. from $\mathbb{B}$), where all those elements belongs to $\mathbb{S}_1$. We thus have two cases. If $k_A^1 + k_B^1 \leq |\mathbb{S}_1|$, we can freely choose all those elements without having duplicates between $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$. Indeed for example, if we have $k_A^1 = k_B^1 = 1$ and $\mathbb{S}_1 = \{0, 1, 2\}$, then we can put 0 in $\widetilde{\mathbb{A}}$ and 1 in $\widetilde{\mathbb{B}}$, thus resulting in no duplicates between $\widetilde{\mathbb{A}}$ and $\widetilde{\mathbb{B}}$. However if we have $k_A^1 + k_B^1 > |\mathbb{S}_1|$, then no matter what, we will have duplicates. Thus in the best case, we have $\max(k_A^1 + k_B^1 - |\mathbb{S}_1|, 0)$ duplicates that we need to count out.

Hence, maximizing $|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}|$ is straightforward as there is one specific order in which to find the values of k_A^i and k_B^i that always maximize the size of the union. We only give the way to optimally build $\widetilde{\mathbb{A}}$ since it is fully similar for $\widetilde{\mathbb{B}}$:

- First, using elements from $\mathbb{S}_5$ to build $\widetilde{\mathbb{A}}$ does not add any duplicate, thus we first pull elements from $\mathbb{S}_5$ and $k_A^5 = \min(k_A, |\mathbb{S}_5|)$.
- As mentioned above, using one element from $\mathbb{S}_1$ adds either zero or one duplicate, thus we then pull elements from $\mathbb{S}_1$ and $k_A^1 = \min(k_A - k_A^5, |\mathbb{S}_1|)$.

Table 2: Results for optimal permutations with $DR_{max}(\pi) = 9$

k	Time	Structure of p	Structure of q	Number of solutions
14	180 min	(6, 6, 1, 1)	(6, 6, 2)	144
		(6, 6, 2)	(6, 6, 1, 1)	144
		(6, 3, 2, 2, 1)	(6, 3, 2, 2, 1)	144
		(12, 2)	(12, 1, 1)	24
		(12, 1, 1)	(12, 2)	24
15	480 min	(10, 2, 2, 1)	(10, 2, 2, 1)	160
16	1023 min	(6, 6, 3, 1)	(6, 6, 3, 1)	432
		(6, 6, 2, 2)	(6, 3, 3, 2, 1, 1)	288
		(6, 3, 3, 2, 1, 1)	(6, 6, 2, 2)	216
17	1700 min	-	-	0
18	2213 min	(8, 8, 1, 1)	(8, 8, 2)	256
		(8, 8, 2)	(8, 8, 1, 1)	256
19	1913 min	-	-	0
20	1116 min	-	-	0
21	400 min	-	-	0

- Finally, elements from either $\mathbb{S}_0$ and $\mathbb{S}_3$ necessarily add duplicates, so we freely choose any $k_A^0 \leq |\mathbb{S}_0|$ and $k_A^3 \leq |\mathbb{S}_3|$ such that $k_A^0 + k_A^3 = k_A - k_A^5 - k_A^1$.

Finally, computing the maximal value for $|\mathbb{K} \cup \widetilde{\mathbb{A}} \cup \widetilde{\mathbb{B}}|$ only requires to compute $|\mathbb{S}_1|$, $|\mathbb{S}_4|$ and $|\mathbb{S}_5|$ and we then check how it compares to k.

4.3 Results

We ran our algorithm for every k such that we need at least 9 rounds to have full diffusion, according to Proposition 1. This corresponds to $14 \leq k \leq 21$, and we were able to find *all* optimal even-odd permutations for $k \in \{14, 15, 16, 18\}$. For $k \in \{17, 19, 20, 21\}$, our algorithm allowed us to prove that there is no even-odd permutation leading to a full diffusion after 9 rounds. Since 9 rounds correspond to the Fibonacci bound, we know that for these cases, we need at least 10 rounds to have full diffusion, and we give later in this section an optimal solution for $k = 17$ reaching full diffusion in 10 rounds, as well as good permutations for $k = 19, 20, 21$ with a diffusion round of 11. We can thus give the following theorem to summarize our results.

Theorem 2. *To build a Generalized Feistel Network GFN_π^k with full diffusion where π is an even-odd permutation, we have :*

- *For $k = 14, 15, 16$ and 18, the optimal number of rounds for full diffusion is 9.*
- *For $k = 17$, the optimal number of rounds for full diffusion is 10.*
- *For $k = 19, 20$ and 21, the optimal number of rounds for full diffusion is at least 10 and at most 11.*

We give in Table 2 an overview of our results. The first column gives the total time needed for our algorithm to either exhaust all optimal even-odd permutations, or prove that no such permutation exists. Note that this is the total CPU time, i.e. when using a single CPU, however our algorithm is highly parallelizable and thus the real time can

Table 3: Optimal equivalence classes with $k = 16$

(p, q)
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 14, 12, 15)$ $q = (2, 6, 12, 10, 1, 13, 4, 15, 7, 9, 14, 5, 8, 3, 11, 0)$
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 14, 12, 15)$ $q = (5, 6, 13, 10, 4, 12, 9, 15, 2, 1, 14, 7, 11, 3, 8, 0)$
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 12, 15, 14)$ $q = (2, 1, 12, 8, 7, 14, 5, 4, 13, 11, 10, 15, 9, 3, 6, 0)$
$p = (1, 2, 3, 4, 5, 0, 7, 8, 6, 10, 11, 9, 13, 12, 14, 15)$ $q = (7, 6, 14, 9, 11, 15, 1, 12, 2, 13, 5, 4, 10, 8, 3, 0)$

be drastically reduced.[3] This shows that our algorithm is extremely efficient, as it can quickly solve the case $k = 16$ for which [CGT19] were not able to give an optimal solution. The second (resp. third) column gives the possible cycle structures of p (resp. q) in an optimal permutation, and the last column gives the number of solutions which have this structure. We can notice that not only the number of solutions is quite low, but also that the number of possible cycle structures is also quite limited. Moreover, we *always* have a fixed point in either p or q.

The most important result in this table is that there are actually even-odd permutations which have full diffusion after 9 rounds for $k = 16$, while both [SM10] and [CGT19] could only find a permutation with full diffusion after 10 rounds, leaving open the question of whether the theoretical bound of 9 rounds (from Proposition 1) could be reached. Our results shows that it is indeed possible, and thus this proves that our permutations are optimal when considering even-odd permutations. We will see in the next section that we can further regroup these permutations into more precise equivalence classes, leading for the case $k = 16$ to four equivalence classes, given in Table 3.

4.4 Security Analysis

Recall that our search space $\mathbb{P}_k$ defined in Proposition 2 contains *at least* one representative for each class. Hence, among all the permutations we found, some of them might actually be in the same equivalence class. We can thus go further and regroup all representatives that belong to the same class using the following proposition.

Proposition 5. *Let $\pi = (p, q)$ be a permutation over $2k$ elements. Then for any permutation r such that $r \circ p \circ r^{-1} = p$, (p, q) and $(p, r \circ q \circ r^{-1})$ are equivalent.*

Proof. Let $\pi = (p, q)$ and $\pi' = (p, r \circ q \circ r^{-1})$ where r is a permutation such that $r \circ p \circ r^{-1} = p$. Recall that we have $\pi(2i) = 2p(i) + 1$ and $\pi(2i + 1) = 2q(i)$, for all $i \in \{0, \ldots, k - 1\}$. Now let φ be the permutation over $2k$ elements defined as

$$\varphi(2i) = 2r(i),\ \varphi(2i + 1) = 2r(i) + 1,\ \forall i \in \{0, \ldots, k - 1\}.$$

Then we have $\pi' = \varphi \circ \pi \circ \varphi^{-1}$. Indeed, if we look at the image of an even number $2i$, we

[3] Less than one hour for each k using 72 threads.

have

$$\begin{aligned}\varphi \circ \pi \circ \varphi^{-1}(2i) &= \varphi \circ \pi(2r^{-1}(i)) \\ &= \varphi(2(p \circ r^{-1})(i) + 1) \\ &= 2(r \circ p \circ r^{-1})(i) + 1 \\ &= 2p(i) + 1 = \pi'(2i).\end{aligned}$$

In the same way, the image of an odd number $2i + 1$ is

$$\begin{aligned}\varphi \circ \pi \circ \varphi^{-1}(2i + 1) &= \varphi \circ \pi(2r^{-1}(i) + 1) \\ &= \varphi(2(q \circ r^{-1})(i)) \\ &= 2(r \circ q \circ r^{-1})(i) \\ &= \pi'(2i + 1)\end{aligned}$$

We thus have $\pi' = \varphi \circ \pi \circ \varphi^{-1}$. Hence, π and π' are conjugate and thus equivalent, according to Definition 4. □

This leads us to the equivalence classes given in Table 4 to 7 in Appendix A for $k = 14, 15, 18$. The column (p, q) gives both permutations p and q. The column Imp. Diff. gives the number of rounds for the longest Impossible Differential distinguisher. Note that this is only considering *structural* Impossible Differentials, where we do not specify neither the size of the blocks nor the definition of the S-boxes, such that contradictions are obtained on blocks rather than bits. The columns $S_N^{s,\delta}$ give the minimal number of rounds to get at least N active S-boxes, where each S-box is of size s and the highest differential probability is $2^{-\delta}$. We chose to only consider three cases : $S_N^{4,2}$, $S_N^{8,6}$ and $S_N^{8,7}$. The first case represents the best case for 4-bit S-boxes. Indeed, we know that there is no APN bijective S-boxes of size 4 (which would lead to a highest differential probability of 2^{-3}). As such, the best case is when the highest differential probability is 2^{-2}. It is still unknown whether 8-bit APN bijective S-boxes exist, so we consider both cases. If such an APN 8-bit S-box exists, the column $S_N^{8,7}$ is relevant, otherwise it would be $S_N^{8,6}$ (for example the AES S-box). The last thing is that N depends on the size of the key (as well as δ). Indeed, if we have a key of size λ, then we want N to verify $2^{-\delta N} < 2^{-\lambda}$, i.e. $N > \frac{\lambda}{\delta}$. As the evaluation of the minimal number of rounds to get at least N S-boxes can be quite expensive, we limited ourselves to $\lambda = 2ks$, where k follows the notation in this paper, i.e. we have $2k$ blocks of s bits and the key is of the same size as the state. Finally, the last column N_{20} shows the minimal number of active S-boxes for 20 rounds, as both [SM10] and [CGT19] also gave this metric for the permutations they found. It is worth mentioning that while our permutations are optimal (w.r.t the diffusion round), for the case $k = 16$, they have a minimal number of active S-boxes over 20 rounds which is lower than for the permutations given in [CGT19] in the same case, where those permutations have a diffusion round of 10 and at least 70 actives S-boxes over 20 rounds. However for all our permutations, the longest Impossible Differentials distinguisher we can build is over 17 rounds, which is at least one round lower than for the permutations with $k = 16$ given in [CGT19].

Note that we were still able to find the following optimal even-odd permutation for $k = 17$, which thus has a diffusion round of 10 :

$$\begin{aligned}p &= (7, 1, 4, 13, 8, 16, 2, 3, 12, 5, 0, 9, 15, 14, 10, 11, 6) \\ q &= (8, 0, 9, 10, 3, 2, 16, 6, 14, 11, 7, 4, 1, 12, 5, 15, 13)\end{aligned}$$

For this permutation, the longest Impossible Differential distinguisher is over 19 rounds, and $S_{69}^{4,2}, S_{46}^{8,6}, S_{39}^{8,7}, N_{20}$ are respectively 20, 16, 15 and 20. For $k = 19, 20, 21$, we easily found permutations reaching full diffusion after 11 rounds with a random search, leaving open the question to find one permutation with a diffusion round of 10. We give an example for these cases below

$$k = 19 : \begin{array}{l} p = (18, 3, 5, 9, 13, 15, 10, 16, 11, 8, 6, 1, 0, 2, 14, 7, 17, 12, 4) \\ q = (9, 14, 2, 6, 3, 8, 16, 4, 0, 13, 18, 15, 5, 11, 7, 17, 12, 1, 10) \end{array}$$

$$k = 20 : \begin{array}{l} p = (14, 5, 15, 1, 17, 3, 11, 8, 4, 0, 6, 13, 19, 10, 2, 9, 18, 12, 16, 7) \\ q = (1, 17, 5, 18, 12, 2, 0, 16, 13, 6, 3, 10, 14, 8, 11, 19, 9, 15, 7, 4) \end{array}$$

$$k = 21 : \begin{array}{l} p = (19, 10, 7, 17, 2, 16, 20, 9, 6, 0, 3, 12, 18, 1, 4, 11, 15, 13, 14, 8, 5) \\ q = (20, 12, 0, 8, 7, 1, 4, 2, 10, 13, 5, 6, 11, 14, 19, 15, 9, 16, 3, 17, 18) \end{array}$$

5 Conclusion

We solved a 10-year-old problem which was to find an optimal (w.r.t diffusion round) even-odd permutation for a Generalized Feistel Network with 32 blocks. More specifically, we showed that there exist permutations which have a diffusion round of 9, while the best permutation found before had a diffusion round of 10. To do so, we give a precise characterization for the permutation to have full diffusion after a given number of rounds. This characterization allowed us to get a very efficient exhaustive search for $k \leq 13$. Even if optimal permutations were already known for these sizes, this shows that our characterization is powerful, thus we have no need to use the elaborated techniques from [CGT19] to treat all these cases. We then exploit this characterization to design a very efficient algorithm that allows us to exhibit *all* optimal even-odd permutations for 32 blocks, as well as for 28, 30 and 36 blocks, which also have an optimal diffusion round of 9 and were not given in the previous literature. For 34, 38, 40 and 42 blocks, our algorithm also allows us to prove that there is no even-odd permutation with a diffusion round of 9 (which is the lower bound), which is again a new result. However for these cases, we were able to give better optimality bounds when considering even-odd permutations, namely for $2k = 34$ the optimal number of rounds for full diffusion is exactly 10 rounds and for $2k = 38, 40, 42$, at most 11 rounds. We also give some security evaluation for Impossible Differentials and Differentials (through the minimum number of active S-boxes). Especially for Impossible Differentials, for the 32 blocks case, all our permutations have their longest impossible differential distinguishers over 17 rounds, which is at least one round lower than every permutation given in [CGT19] for this case.

References

[AIK+00] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. 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*, pages 39–56, 2000.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 12–23, 1999.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.

[CGT19] Victor Cauchois, Clément Gomez, and Gaël Thomas. General Diffusion Analysis: How to Find Optimal Permutations for Generalized Type-II Feistel Schemes. *IACR Trans. Symmetric Cryptol.*, 2019(1), 2019.

[DES77] DES. Data Encryption Standard. *FIPS PUB 46, Federal information processing standards publication 46*, 1977.

[GM16] Shay Gueron and Nicky Mouha. Simpira v2: A family of efficient permutations using the AES round function. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 95–125, 2016.

[HSH+06] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pages 46–59, 2006.

[KPP+17] Daniel Kales, Léo Perrin, Angela Promitzer, Sebastian Ramacher, and Christian Rechberger. Improvements to the linear layer of lowmc: A faster picnic. *IACR Cryptology ePrint Archive*, 2017:1148, 2017.

[LR88] Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, 1988.

[Nyb96] Kaisa Nyberg. Generalized Feistel Networks. In *Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings*, pages 91–104, 1996.

[Pie90] Josef Pieprzyk. How to Construct Pseudorandom Permutations from Single Pseudorandom Functions. In *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, pages 140–150, 1990.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[SIH+11] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 342–357, 2011.

[SKW+98] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. *NIST AES Proposal*, 15:23, 1998.

[SM10] Tomoyasu Suzaki and Kazuhiko Minematsu. Improving the generalized Feistel. In *International Workshop on Fast Software Encryption*, pages 19–39. Springer, 2010.

[SMMK12] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE : A Lightweight Block Cipher for Multiple Platforms. In *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, pages 339–354, 2012.

[SSA+07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, pages 181–195, 2007.

[ZMI89] Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 461–480, 1989.

A Results for Optimal Permutations

Table 4: Security evaluation for the best equivalence classes with $k = 14$

(p, q)	Imp. Diff	$S_{57}^{4,2}$	$S_{38}^{8,6}$	$S_{33}^{8,7}$	N_{20}
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 12, 13)$ $q = (10, 7, 13, 11, 9, 8, 4, 1, 12, 5, 3, 2, 6, 0)$	17	19	14	13	66
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 12, 13)$ $q = (8, 6, 13, 10, 7, 9, 1, 12, 5, 2, 4, 3, 11, 0)$	17	19	14	13	66
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 12)$ $q = (9, 1, 13, 5, 2, 10, 3, 7, 12, 11, 8, 4, 6, 0)$	17	19	14	13	66
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 12)$ $q = (4, 1, 13, 5, 10, 9, 2, 11, 8, 12, 6, 3, 7, 0)$	17	19	14	13	66
$p = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 0, 13, 12)$ $q = (3, 5, 2, 13, 0, 10, 9, 11, 8, 12, 6, 4, 7, 1)$	17	22	14	13	52
$p = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 0, 13, 12)$ $q = (3, 1, 13, 11, 8, 10, 9, 7, 12, 5, 2, 4, 6, 0)$	17	22	14	13	52
$p = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 0, 12, 13)$ $q = (3, 11, 8, 13, 6, 10, 9, 5, 2, 12, 0, 4, 7, 1)$	17	22	14	13	52
$p = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 0, 12, 13)$ $q = (3, 7, 13, 5, 2, 10, 9, 1, 12, 11, 8, 4, 6, 0)$	17	22	14	13	52
$p = (1, 2, 3, 4, 5, 0, 7, 8, 6, 10, 9, 12, 11, 13)$ $q = (4, 9, 6, 11, 13, 12, 10, 2, 8, 1, 5, 3, 7, 0)$	17	23	19	18	46

Table 5: Security evaluation for the best equivalence classes with $k = 15$

(p, q)	Imp. Diff	$S_{61}^{4,2}$	$S_{41}^{8,6}$	$S_{35}^{8,7}$	N_{20}
$p = (1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 11, 10, 13, 12, 14)$ $q = (12, 5, 10, 3, 11, 1, 13, 9, 14, 7, 4, 6, 2, 8, 0)$	17	20	16	14	61
$p = (1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 11, 10, 13, 12, 14)$ $q = (13, 9, 10, 7, 11, 5, 12, 3, 14, 1, 4, 6, 8, 2, 0)$	17	42	28	24	30

Table 6: Security evaluation for the best equivalence classes with $k = 16$

(p, q)	Imp. Diff	$S_{65}^{4,2}$	$S_{43}^{8,6}$	$S_{37}^{8,7}$	N_{20}
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 14, 12, 15)$ $q = (2, 6, 12, 10, 1, 13, 4, 15, 7, 9, 14, 5, 8, 3, 11, 0)$	17	33	22	19	40
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 14, 12, 15)$ $q = (5, 6, 13, 10, 4, 12, 9, 15, 2, 1, 14, 7, 11, 3, 8, 0)$	17	33	22	19	40
$p = (1, 2, 3, 4, 5, 0, 7, 8, 9, 10, 11, 6, 13, 12, 15, 14)$ $q = (2, 1, 12, 8, 7, 14, 5, 4, 13, 11, 10, 15, 9, 3, 6, 0)$	17	50	33	29	26
$p = (1, 2, 3, 4, 5, 0, 7, 8, 6, 10, 11, 9, 13, 12, 14, 15)$ $q = (7, 6, 14, 9, 11, 15, 1, 12, 2, 13, 5, 4, 10, 8, 3, 0)$	17	50	33	29	26

Table 7: Security evaluation for the best equivalence classes with $k = 18$

(p, q)	Imp. Diff	$S_{73}^{4,2}$	$S_{49}^{8,6}$	$S_{42}^{8,7}$	N_{20}
$p = (1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, 12, 13, 14, 15, 8, 16, 17)$ $q = (10, 9, 14, 12, 15, 11, 13, 17, 2, 1, 6, 4, 7, 3, 5, 16, 8, 0)$	17	31	22	19	44
$p = (1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, 12, 13, 14, 15, 8, 16, 17)$ $q = (14, 8, 12, 15, 13, 10, 9, 17, 7, 6, 16, 3, 5, 1, 4, 2, 11, 0)$	17	56	38	32	26
$p = (1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, 12, 13, 14, 15, 8, 17, 16)$ $q = (2, 1, 6, 12, 15, 3, 13, 16, 10, 9, 14, 4, 7, 11, 5, 17, 8, 0)$	17	31	22	19	44
$p = (1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, 12, 13, 14, 15, 8, 17, 16)$ $q = (11, 5, 9, 12, 2, 7, 6, 16, 3, 13, 1, 4, 10, 15, 14, 17, 8, 0)$	17	56	38	32	26

B Efficient Implementation to Test 9 Round Full Diffusion

We give an example of a C++ implementation of the characterization for a permutation to have full diffusion over 9 rounds. This function takes `powerp` and `q` as parameters, which are respectively, the precomputed values of each power of p, i.e. `powerp[i][j] =` $p^i(j)$, and the permutation q.

```
bool checkDiffusion(vector<vector<unsigned int>> const & powerp,
                    vector<unsigned int> const & q){

        auto const & p6 = powerp[6];
        auto const & p5 = powerp[5];
        auto const & p4 = powerp[4];
        auto const & p3 = powerp[3];
        auto const & p2 = powerp[2];
        auto const & p = powerp[1];

        unsigned int sizeperm = q.size();
        for(unsigned int x = 0; x < sizeperm; x++){
                unsigned int qx = q[x];
                unsigned int qpq = q[p[qx]];
                unsigned int qp = q[p[x]];
                unsigned int qp2 = q[p2[x]];
                unsigned int qp3 = q[p3[x]];
                unsigned int qp2q = q[p2[qx]]];
                unsigned int qpqp = q[p[qp]];

                unsigned int indicator = (1 <<p6[x]);
                indicator |= (1 << p5[qx]);
                indicator |= (1 << p4[qp]);
                indicator |= (1 << p3[qp2]);
                indicator |= (1 << p2[qp3]);
                indicator |= (1 << p[q[p4[x]]]);
                indicator |= (1 << q[p5[x]]);
                indicator |= (1 << p3[qpq]);
                indicator |= (1 << p2[qp2q]);
                indicator |= (1 << p2[qpqp]);
                indicator |= (1 << p[q[p3[qx]]]);
                indicator |= (1 << p[q[p2[qp]]]);
                indicator |= (1 << p[q[p[qp2]]]);
                indicator |= (1 << q[p4[qx]]);
                indicator |= (1 << q[p3[qp]]);
                indicator |= (1 << q[p2[qp2]]);
                indicator |= (1 << q[p[qp3]]);
                indicator |= (1 << p[q[p[qpq]]]);
                indicator |= (1 << q[p2[qpq]]);
                indicator |= (1 << q[p[qp2q]]);
                indicator |= (1 << q[p[qpqp]]);
                if(__builtin_popcount(indicator) != sizeperm)
                        return false;
        }
        return true;
}
```