Generic Round-Function-Recovery Attacks for Feistel Networks over Small Domains

F. Betül Durak(✉) and Serge Vaudenay

Ecole Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland
betul.durak@epfl.ch

Abstract. Feistel Networks (FN) are now being used massively to encrypt credit card numbers through format-preserving encryption. In our work, we focus on FN with two branches, entirely unknown round functions, modular additions (or other group operations), and when the domain size of a branch (called N) is small. We investigate round-function-recovery attacks.

The best known attack so far is an improvement of Meet-In-The-Middle (MITM) attack by Isobe and Shibutani from ASIACRYPT 2013 with optimal data complexity $q = r\frac{N}{2}$ and time complexity $N^{\frac{r-4}{2}N+o(N)}$, where r is the round number in FN. We construct an algorithm with a surprisingly better complexity when r is too low, based on partial exhaustive search. When the data complexity varies from the optimal to the one of a codebook attack $q = N^2$, our time complexity can reach $N^{O\left(N^{1-\frac{1}{r-2}}\right)}$. It crosses the complexity of the improved MITM for $q \sim N\frac{e^3}{r}2^{r-3}$.

We also estimate the lowest secure number of rounds depending on N and the security goal. We show that the format-preserving-encryption schemes FF1 and FF3 standardized by NIST and ANSI cannot offer 128-bit security (as they are supposed to) for $N \leqslant 11$ and $N \leqslant 17$, respectively (the NIST standard only requires $N \geqslant 10$), and we improve the results by Durak and Vaudenay from CRYPTO 2017.

1 Introduction

Feistel Networks (FN) have been used in constructing many block ciphers such as DES [1]. In the classical FN, we construct a permutation from $2n$ bits to $2n$ bits with round functions from n bits to n bits. We call it as balanced Feistel network. Figure 1 represents a 4-round FN with modular addition (modulo the size of the domain for a branch). Other well known types of Feistel networks are unbalanced FN, alternating between contracting and expanding round functions.

Although block ciphers only encrypt blocks of a fixed format (typically: a binary string of length 128), there are many applications requiring to encrypt data of another format (such as a decimal string of a given length) and to have encrypted data in the same format. For example, Credit Card Numbers (CCN)

B. Preneel and F. Vercauteren (Eds.): ACNS 2018, LNCS 10892, pp. 440–458, 2018.
https://doi.org/10.1007/978-3-319-93387-0_23

consist of 16 decimal numbers, whose 6 digits must be kept confidential. For this reason, these 6 digits are typically encrypted in digital transactions using a Format-Preserving Encryption (FPE). Recently, FPE based on FN [5,6,9] have been standardized [2,3]. As an example, the FPE solution of the terminal manufacturer company Verifone encrypts about 30M credit card transactions per day in the United States alone.

In this work, we are specifically interested in FN with two branches (not necessarily balanced) with secret round functions and modular addition operation. Moreover, we are interested in small domain size over larger key space. We investigate the security when the round function is entirely unknown instead of a publicly known round function that mixes the input with a secret key (i.e. round function is $F_i = f_i(k_i, .)$, where k_i is the round key in i^{th} round). We do not assume that round functions are bijective. This applies to FF1 [6] by Bellare et al. and FF3 [9] by Brier et al. which have been standardized by The National Institute of Standards an Technology (NIST) published in March, 2016 [2]. This standard aims at a 128-bit security for any $N \geqslant 10$. FF3 was broken and repaired by Durak and Vaudenay [15]. Herein, we denote by FF3* the repaired scheme.

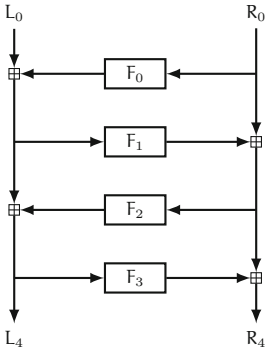


Fig. 1. 4-round Feistel network

Since their invention, Feistel networks and their security analysis have been studied. Many cryptanalysis studies have been done to give key-recovery, message-recovery, round-function-recovery, and differential attacks on different types of Feistel networks [7,12,16,18,21,24]. We summarize the best function recovery attacks in Table 1.[1] The complexities are given in terms of number of encryptions. In Appendix, we present a brief survey of existing attacks. So far, the best generic attack was a variant of Meet-In-The-Middle (MITM) attack.

The most famous security result dates back to late 80's given by Luby-Rackoff [20]. In their seminal paper, Luby and Rackoff first showed that a three round Feistel construction is a secure pseudorandom permutation from $2n$ bits

[1] Table 1 only reports function recovery attacks. It does not include attacks applying with round functions in a small space of N (instead of N^N). It does not include distinguishers such as the ones from Patarin [22] either.

to $2n$ bits. Moreover, they showed that for more than three rounds FN, all generic chosen-plaintext attacks on Feistel schemes require $q = \Omega(2^{\frac{n}{2}})$ queries where n is the input/output size to the round function. Information theoretically, the number q of queries provides $2qn$ bits of information. For r-round FN, we need $rn2^n$ bits of information to recover the round functions (each round function can be represented with a string of size $n2^n$). Therefore, $q = \frac{r}{2}2^n$ is enough to reconstruct the round function, in theory. Patarin [23] further showed that for $q \ll 2^n$, four rounds are secure against known-plaintext attacks (the advantage would be bounded by $\frac{4q}{2^n} + \frac{q^2}{2\cdot 2^n}$ for $q \leqslant \frac{2^n}{67n}$), five rounds are secure against chosen-plaintext attacks (the advantage would be bounded by $\frac{5q}{2^n} + \frac{q^2}{2\cdot 2^n}$ for $q \leqslant \frac{2^n}{67n}$) and six rounds are secure against chosen-plaintext and ciphertext attacks (the advantage would be bounded by $\frac{8q}{2^n} + \frac{q^2}{2\cdot 2^n}$ for $q \leqslant \frac{2^n}{128n}$).

As we will not necessarily assume messages in binary, we use the notation N_l, N_r as the domain size of the round functions. We introduce some known attacks on Feistel networks with our focused properties: two branches with domain size N_l and N_r, with modular addition modulo N_l and N_r, secret random round functions which are balanced ($N = N_l = N_r$) or unbalanced but with $N_l \approx N_r$.

Table 1. Round-function-recovery attacks against generic balanced 2-branch r-round FN with domain branch size N. (All β are different constants such that $\beta < 1$.)

Roundsc	Method	Type	Requirement	Time complexity T	Data q	Ref
3	Yo-yo	Known pt		$O(N \ln N)$	$N \ln N$	[15]
4	Cycle finding	Known pt		$O\left(N^3\right)$	$N^{\frac{3}{2}}$	[15]
4	Guess and determine	Chosen pt		$O\left(N^{\frac{3}{2}}\right)$	$N^{\frac{3}{2}}$	[7]
5	Cycle finding	Chosen pt		$O\left(N^{\sqrt{N}+3}\right)$	$N^{\frac{3}{2}}$	[15]
5	Integral attack	Chosen pt	F_1 or F_3 invertible	$O\left(N^{2.81}\right)$	N^2	[7]
5	Yo-yo	Codebook	$\oplus$-Feistel	$O\left(N^2\right)$	N^2	[7]
5	Guess and determine	Codebook		$O\left(N^{N^{\frac{3}{4}}}\right)$	N^2	[7]
5	SAT solver	Codebook		Not specified	N^2	[8]
6	Yo-yo	Codebook	$\oplus$-Feistel	$O\left(N^{\frac{1}{2}N}\right)$	N^2	[7]
7	Yo-yo	Codebook	$\oplus$-Feistel	$O\left(N^N\right)$	N^2	[7]
r	Cycle finding	Chosen pt		$O\left(N^{(r-5)N+\sqrt{N}+3}\right)$	$N^{\frac{3}{2}}$	[15]
r	MITM	Known pt		$O\left(N^{\lceil \frac{r}{2} \rceil N}\right)$	$r\frac{N}{2}$	Eq. (1)
r	MITM*	Chosen pt		$N^{\frac{r-4}{2}N(1+o(1))}$	$r\frac{N}{2}$	Eq. (2)
r	Iterated partial exhst search	Known pt		$N^{\frac{(r-2)^2}{r-1}N\left(\frac{N}{q}\right)^{\frac{1}{r-2}}(\beta+o(1))}$	$q \leqslant N^2$	Eq. (5)
r	Iterated partial exhst search	Chosen pt		$N^{(r-3)N^{1-\frac{1}{r-2}}(\beta+o(1))}$	$\beta N^{2-\frac{1}{r-2}}$	Eq. (8)
r	Iterated partial exhst search	Chosen pt		$N^{\frac{q}{N}-1+\frac{(r-3)^2}{r-2}N\left(\frac{N}{q}\right)^{\frac{1}{r-3}}(\beta+o(1))}$	$q \leqslant N^2$	Eq. (7)

Our Contributions. In this work, we propose the best known generic exhaustive search attack on Feistel networks with two branches and random functions with arbitrary number r of rounds. We compare it with MITM. It is better for some parameters. When the data complexity varies in between the optimal (based on information theory) and the one of the codebook attack, our best time complexity goes from $N^{\frac{r-2}{2}N+o(N)}$ (MITM-based, see Eq. (2) for r even) to $N^{O(N^{1-\frac{1}{r-2}})}$ (based on partial exhaustive search, see Eq. (8)), where N is the domain size of the branch. More precisely, the optimal data complexity is $q = r\frac{N}{2}$. MITM works with the optimal data complexity and with time complexity $T^{MITM^*} = N^{\frac{r-2}{2}N+o(N)}$ (see Eq. (2)). Our partial exhaustive search attack can use any data complexity from the optimal to the one of a codebook $q = N^2$, but it is better than MITM for $q > \frac{N \times e^3}{r} 2^{r-3}$. It reaches the time complexity (called T^{Iter^*}) $N^{(r-3)N^{1-\frac{1}{r-2}}(\beta+o(1))}$ for some constant $\beta < 1$ (see Eq. (8)) using $q = \beta N^{2-\frac{1}{r-2}}$ chosen plaintexts.

We plot in Fig. 2 the (r, N) parameters for which we have $T^{Iter^*} = T^{MITM^*}$. As we can see, for any constant N and a low r (including $r = 8$ and $r = 10$ as the NIST standards suggest), $Iter^*$ is the best attack. The same figure includes two curves that correspond to the 128-bit and 256-bit security parameters (r, N). The curves are computed with the minimum between T_{Iter^*} and T^{MITM^*}. It can be read that an intended 128-bit security level in FF3* with $r = 8$, $N \leqslant 17$ and in FF1 with $r = 10$, $N \leqslant 11$ has not been satisfied.[2] E.g., for 6-bit messages and 2-digit messages.[3]

Another application could be to reverse engineer an S-box based on FN [8].

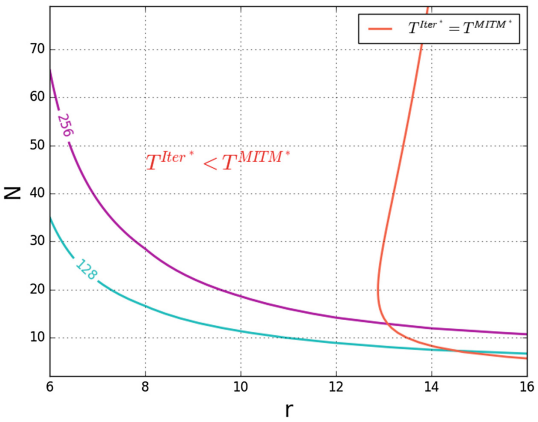


Fig. 2. Parameters (r, N) for $T^{Iter^*} = T^{MITM^*}$ and parameters to meet a 128-bit and a 256-bit security level.

[2] It was shown by Durak and Vaudenay [15] that 128-bit security was not reached by FF3* and FF1 for $7 \leqslant N \leqslant 10$ and $N = 7$, respectively.

[3] Note that the NIST standard [2] requires $N \geqslant 10$.

Structure of Our Paper. In Sect. 2, we review the symmetries in the set of tuples of round functions which define the same FN and we describe the MITM attacks. Our algorithm is described and analyzed in Sect. 3. Section 4 applies our results to format preserving encryption standards. Finally, we conclude.[4]

2 Preliminaries

In this section, we present known techniques to recover the r-tuple of round functions in FN. Note that we actually recover an *equivalent tuple of round functions*. Indeed, we can see that round functions differing by constants can define the actual same cipher [13,15]. Concretely, let $(F_0, \ldots, F_{r-1})$ be a tuple defining a cipher C. For every $a_0, \ldots, a_{r-1}$, $b_0, \ldots b_{r-1}$ such that $a_i = b_{i-1} + b_{i-3} + b_{i-5} + \cdots$ and $b_0 + b_2 + b_4 + \cdots = b_1 + b_3 + b_5 + \cdots = 0$, we can define $(F'_0, \ldots, F'_{r-1})$ by $F'_i(x) = F_i(x - a_i) + b_i$. We obtain a tuple defining the same cipher C. Therefore, we can fix arbitrarily one point of $F_0, \ldots, F_{r-3}$ and we are ensured to find an equivalent tuple of functions including those points.

2.1 Meet-In-The-Middle (MITM) Attack

The MITM attack was introduced by Diffie and Hellman [10]. It is a generic known-plaintext attack. Briefly, consider an r round encryption $E_0, E_1, \ldots, E_{r-1}$ and corresponding $D_0, D_1, \ldots, D_{r-1}$ decryption algorithms. We assume each algorithm uses a k-bit key and we denote the keys by $K_0, K_1, \ldots, K_{r-1}$. Let $M_1, M_2, \ldots, M_q$ be the plaintexts and $C_1, C_2, \ldots, C_q$ be the corresponding ciphertexts. Let the intermediate values entering to round i be $M_1^{(i)}, M_2^{(i)}, \ldots, M_q^{(i)}$ for $1 \leqslant i < r$. The adversary enumerates each possible combination of the keys $K_0, K_1, \ldots, K_{u-1}$ for the first $u = \lfloor \frac{r}{2} \rfloor$ rounds and it computes the intermediate values for each plaintexts as $M_1^{(u)}, M_2^{(u)}, \ldots, M_q^{(u)}$ until round u. Then, these values along with their possible keys are stored in a table (The memory complexity is 2^{uk} messages). Then, the adversary partially decrypts the ciphertext $C_1, C_2, \ldots, C_q$ for each value of the keys $K_{r-1}, K_{r-2}, \ldots, K_u$ backward. Finally, the adversary looks for a match between the partially decrypted values and the rows of the stored table. Each match suggests keys for $K_0, K_1, \ldots K_{r-1}$ and the adversary recovers all the keys. The time complexity of the MITM attack is $2^{(r-u)k}$ and memory complexity is 2^{uk}.[5]

We can apply the MITM attack to the Feistel networks with r rounds and q known plaintext/ciphertext pairs. In our setting, N is quite small, thus we can focus on a generic FN with functions specified by tables. This is equivalent to

[4] The full version of our paper [14] includes appendices with: a description of the message recovery attacks from Bellare et al. [4], the generic round-function-recovery attack from Durak and Vaudenay [13,15], an attack exploiting the bias in the modulo-N reduction inspired by Bleichenbacher (as described by Vaudenay [25]), and the generic round-function-recovery attacks by Biryukov et al. [7].

[5] In order to improve the memory complexity of MITM attack, a new technique called dissection attack has been introduced by Dinur et al. in [11].

using a key of $k = N \log_2 N$ bits. Therefore, the standard MITM attack has a time complexity of $N^{(r-u)N}$ with same memory complexity. We label the time complexity as follows:

$$T^{MITM} = O\left(N^{\lceil \frac{r}{2} \rceil N}\right) \tag{1}$$

with $q = \frac{rN}{2}$ **known plaintexts**. The pseudocode is given in Algorithm 1.

Algorithm 1. Meet-In-The-Middle Attack (MITM)

```
Collect q plaintext-ciphertext pairs (M_i, C_i), i = 1, ..., q.
foreach K_0, ..., K_{u-1} do
    Compute M_1^(u), ..., M_q^(u) forward from M_1, ..., M_q.
    Store (K_0, ..., K_{u-1}) in h(M_1^(u), ..., M_q^(u)).
end
foreach K_u, ..., K_{r-1} do
    Compute M_1^(u), ..., M_q^(u) backward from C_1, ..., C_q.
    foreach K_0, ..., K_{u-1} in h(M_1^u, ..., M_q^u) do
        Output K_0, ..., K_{r-1}.
    end
end
```

2.2 Improved MITM

In this section, we elaborate and extend the attack mentioned briefly in [11,12] on r-round FN. The same attack appears in [17,18] with $k = \log_2 N$. We are only adapting the algorithm to our settings. We take $u = \lceil \frac{r}{2} \rceil - 1$ and $v = \lfloor \frac{r}{2} \rfloor - 1$ so that $r = u + v + 2$ and $u \approx v$. Consider the FN in Fig. 3 for r even (When r is odd, we can set $u = \lfloor \frac{r}{2} \rfloor - 1$ so that $r - u - 2 = \lceil \frac{r}{2} \rceil - 1$). We can split the $(2u+2)$- round FN in 4 parts: starting with a single round F_0; a u-round Feistel Network called G, the $(u+2)^{th}$ round with function F_{u+1}, and finally another v-round Feistel Network called H.

An intuitive attack works as follows. Fix a value $M_R^{(0)} = a$ and consider all possible $M_L^{(0)}$ so that we obtain N plaintexts. We do it $\frac{q}{N}$ times to obtain q plaintexts. Hence, we have $\frac{q}{N}$ values for a. We set the output of F_0 for one value of a arbitrarily. For all the plaintexts, we query $(M_L^{(0)} \| M_R^{(0)})$ and obtain q $(C_L \| C_R)$ values. We enumerate all the functions of H, and compute $(M_L^{(u+2)} \| M_R^{(u+2)})$ from $(C_L \| C_R)$ by decrypting. We set $Z = M_L^{(u+2)} = M_L^{(u+1)}$ if u is even and set $Z = M_R^{(u+2)} = M_R^{(u+1)}$ if u is odd. We store each Z in a hash table. We then enumerate all the functions of G, and compute $(M_L^{(u+1)} \| M_R^{(u+1)})$ from $(M_L^{(0)} \| M_R^{(0)})$. For each computed values of $M_L^{(u+1)}$ (for u even) or $M_R^{(u+1)}$ (for u odd), we look for a match in the hash table storing Z values (since they have to be equal). The time complexity of this approach consists of enumerating many values and functions with memory complexity $vN \log_2(N)$ to store the hash table.

Enumerating $F_0, (F_1, \ldots, F_u)$ and $F_{u+2}, \ldots, F_{r-1}$ gives $N^{\frac{q}{N}-1+(u+v)(N-1)}$ tuples which are filtered by N^{-q}. We obtain $N^{\frac{q}{N}-1+(u+v)(N-1)-q}$ tuples. Thus, for each filtered tuple, we can deduce input/output values for F_{u+1} and rule out inconsistent tables to isolate the solutions $(F_0, \ldots, F_{r-1})$. This post-filtering has a complexity $N^{\frac{q}{N}-1+(u+v)(N-1)}$. We will see that it is lower than the complexity of the rest of the algorithm. Thus, it disappears in the big-O. The pseudocode is given in Algorithm 2.

Algorithm 2. Improved Meet-In-The-Middle Attack (MITM*)

```
Take a_1, ..., a_{q/N} pairwise different half blocks.
Take M_1, ..., M_q pairwise different such that (M_i)_R ∈ {a_1, ..., a_{q/N}}.
Collect the encryption C_1, ..., C_q of M_1, ..., M_q.
foreach v-round Feistel Network H do
    Compute M_1^{(u+2)}, ..., M_q^{(u+2)} backward from C_1, ..., C_q.
    Set Z_i = M_L^{(u+2)} if u is even and Z_i = M_R^{(u+2)} if u is odd, i = 1, ..., q.
    Store H in h(Z_1, ..., Z_q).
end
Set b_1 arbitrarily.
foreach b_2, ..., b_{q/N} do
    Set F_0(a_i) = b_1, i = 1, ..., q/N.
    foreach u-round Feistel Network G do
        Compute M_1^{(u+1)}, ..., M_q^{(u+1)} forward from M_1, ..., M_q.
        Set Z_i = M_L^{(u+1)} if u is even and Z_i = M_R^{(u+1)} if u is odd, i = 1, ..., q.
        foreach H stored in h(Z_1, ..., Z_q) do
            Deduce input/output values for F_{u+1}.
            if consistent then
                Output (F_0, G, H).
            end
        end
    end
end
```

In this attack, we have to guess $N^{\frac{q}{N}-1}$ values for F_0, $N^{u(N-1)}$ values (we have $N-1$ instead of N because one value per round is free to select) for enumerating $F_1, F_2, \ldots, F_u$ (we guess $N^{\frac{q}{N}-1+u(N-1)}$ values in total). And, we guess $N^{v(N-1)}$ values for enumerating $F_{u+2}, F_{u+3}, \ldots, F_{r-1}$ (we guess $N^{v(N-1)}$ in total). Therefore, the complexity is $O\left(N^{\frac{q}{N}-1+(\frac{r}{2}-1)(N-1)}\right)$ for r is even and $O\left(N^{\frac{q}{N}-1+\left(\frac{r-1}{2}\right)(N-1)}\right)$ for r is odd. We label the time complexity for described attack as:

$$T^{\mathrm{MITM}^*} = O\left(N^{\left(\frac{r}{2}-1\right)N}\right), \quad \text{for } r \text{ even} \tag{2}$$
$$T^{\mathrm{MITM}^*} = O\left(N^{\frac{r-1}{2}N-\frac{1}{2}}\right), \quad \text{for } r \text{ odd}$$

with $q = \frac{rN}{2}$ **chosen plaintexts**.

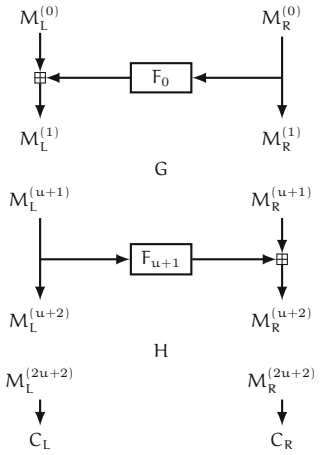


Fig. 3. (2u+2)-round Feistel network (with u even on the picture)

3 Round-Function-Recovery by Partial Exhaustive Search

We consider exhaustive search algorithms dealing with partial functions. Normally, a function F_j is defined by its set of all possible $(z, F_j(z))$ pairs. We call a table as *partial table* if it is a subset of its table. It is a set of pairs such that

$$\forall x, y, z \qquad (x, y) \in F_i \text{ and } (x, z) \in F_i \Longrightarrow y = z.$$

If $(x, y) \in F_i$, we say that $F_i(x)$ is defined and we denote $F_i(x) = y$. The *density* of a partial table is the ratio θ of its cardinality by N. For example, $\theta = \frac{1}{N}$ corresponds to a partial table defined on a single point z and $\theta = 1$ corresponds to the full table. Our aim is to enumerate possible partial tables of increasing density by exhaustive search. So, we will "extend" partial functions. A partial table is an *extension* of another partial table if the former is a superset of the latter.

We deal with partial tables for each round function. We define r-tuples T of partial tables in which T_j denotes the partial table of F_j in T.[6] We say that T is homogeneous with density θ if for all j, T_j has density θ. Similarly, a tuple T' is an extension of T if for each j, T'_j is an extension of T_j. An *elementary tuple* is a homogeneous tuple of density $\frac{1}{N}$. This means that each of its partial functions are defined on a single point.

[6] We denote an r-tuple with capital letter T. Each tuple T consists of r tables, i.e. $T = \{T_0, \ldots T_{r-1}\}$. When we have multiple r-tuples, we denote different tuples indexed with a superscript $T^1, T^2, \ldots$.

Again, our aim is to start with an elementary tuple and to list all extensions, as long as they are compatible (as defined below) with the collected pairs of plaintexts and ciphertexts (M, C). We say that a tuple T *encrypts* a plaintext M_i into a ciphertext C_i (or *decrypts* C_i into M_i or even that T encrypts the pair $(\mathsf{M}_i, \mathsf{C}_i)$) if we can evaluate the FN on M_i with the partial information we have about the round functions and if it gives C_i. We say that a pair $(\mathsf{M}_i, \mathsf{C}_i)$ is *computable except for r' rounds* for a tuple T if there exists a round number j such that the partial functions are enough to encrypt M_i for up to j rounds and to decrypt C_i for up to $r - j - r'$ rounds.

We want to define what it means for a tuple to be compatible with (M, C). Roughly, it is compatible if for each i, there exists an extension encrypting M_i into C_i. (However, it does not mean there exists an extension encrypting each M_i to C_i.) More precisely, we say that a tuple T of partial tables is *compatible* with (M, C) if for each i, at least one of the following conditions is satisfied:

(i) T encrypts M_i into C_i
(in this case, there is no need to extend T);
(ii) $(\mathsf{M}_i, \mathsf{C}_i)$ is computable except for two rounds or more
(indeed, if two rounds are undetermined, we know that we can extend T to encrypt M_i to C_i);
(iii) $(\mathsf{M}_i, \mathsf{C}_i)$ is computable except for one round (numbered s below) and their is a match in the value skipping the missing round: more precisely, their exists $s \in \{0, \ldots, r-1\}$ and one (x, y) pair such that if $\mathsf{T}'_s = \mathsf{T}_s \cup \{(x, y)\}$, the tuple $\mathsf{T}' = (\mathsf{T}_0, \ldots, \mathsf{T}_{s-1}, \mathsf{T}'_s, \mathsf{T}_{s+1}, \ldots, \mathsf{T}_{r-1})$ encrypts M_i to C_i
(indeed, we know we can extend the missing round with $\mathsf{T}_s(x) = y$).

Clearly, if no condition is satisfied for i, then no extension of T can encrypt M_i into C_i, so we can prune an exhaustive search.

3.1 Iter: Iterative Partial Exhaustive Search

Assume that q plaintext/ciphertext pairs $(\mathsf{M}_i, \mathsf{C}_i)$ are known to the adversary. Due to the symmetries in the set of tuples which are compatible with the codebook, we can focus on the tuples which are extensions of an arbitrarily fixed elementary tuple T^1 which encrypts the pair $(\mathsf{M}_1, \mathsf{C}_1)$. So, we define Pool_i as the set of all extensions T of T^1 encrypting the pairs $(\mathsf{M}_1, \mathsf{C}_1), \ldots, (\mathsf{M}_i, \mathsf{C}_i)$, which are compatible with all other pairs, and which are *minimal* (in the sense that removing any entry in the partial tables of T makes at least one $(\mathsf{M}_j, \mathsf{C}_j)$ pair not computable, for $1 \leqslant j \leqslant i$).

We iteratively construct Pool_i. For that, we take all possible minimal extensions of tuples from Pool_{i-1} which encrypt the i^{th} pair and remain compatible with all others. We proceed as defined by Algorithm 3.

With an appropriate data structure, we can avoid to retry to encrypt M_j or decrypt C_j and directly go to the next computable round (if any) in every pair. For each tuple T in Pool_i, we maintain a hash table h in which $h(u, x)$ is a list of pairs of the form $(j, +)$ or $(j, -)$ with $j > i$. If $(j, +)$ is in $h(u, x)$, this means that

Algorithm 3. Iterative partial exhaustive search round-function-recovery attack

```
1  Collect q plaintext-ciphertext pairs (M_i, C_i), i = 1, ..., q.
2  Get an arbitrary elementary tuple T^1 which encrypts M_1 to C_1.
3  Initialize Pool_1 = {T^1}.
4  foreach i = 2, ..., q do
5  |  Initialize Pool_i to empty.
6  |  foreach T ∈ Pool_{i-1} do
7  |  |  foreach minimal extension T' of T encrypting M_i to C_i do
8  |  |  |  if all (M_{i+1}, C_{i+1}), ..., (M_q, C_q) are compatible with T' then
9  |  |  |  |  Add T' in Pool_i.
10 |  |  |  end
11 |  |  end
12 |  end
13 end
14 Output Pool_q.
```

T encrypts M_j up to round $u-1$ and that the input to F_u (the output of which is unknown) is x. If $(j,-)$ is in $h(u,x)$, this means that T decrypts M_j up to round $u+1$ and that the input to F_u is x. Concretely, this means that $h(u,x)$ lists the indices of (M_j, C_j) pairs who need the value of $F_u(x)$ to encrypt/decrypt one more round. With this type algorithmic trick, we save the inner loop and the complexity is expected to be close to the total size of the pools: $\sum_{i=1}^{q} |\mathtt{Pool}_i|$.

3.2 A Heuristic Complexity Analysis of Iter

We heuristically estimate $|\mathtt{Pool}_i|$. First, we recall that $\mathtt{Pool}_i$ is the subset of all minimal extensions of the elementary tuple T^1 which encrypt the first i plaintext/ciphertext pairs, restricted to the ones which are compatible with all others.

We approximate $|\mathtt{Pool}_i|$ by N^{X-Y} where X is the number of entries in the partial tables (i.e. the number of defined points throughout all rounds) and Y is the number of independent equations modulo N which a tuple must satisfy to be compatible. So, N^{-Y} is the probability for a tuple to satisfy the conditions in $\mathtt{Pool}_i$. In other words, the N^X possible tuples are decimated by a factor N^Y. To treat the fact that we start with only T^1 in $\mathtt{Pool}_1$, we decrease X by r (it means that entries defined in T^1 do not have to be enumerated as they are fixed) and we decrease Y by 2 (i.e., we consider that the (M_1, C_1) pair never decimates tuples as it is always compatible by the choice of T^1).

Although it would be inefficient to proceed this way, we could see $\mathtt{Pool}_i$ as follows. For all sets $(T^2, \ldots, T^i)$ of elementary tuples in which T^j encrypts the j^{th} pair, we check if $\{T^1, T^2, \ldots, T^i\}$ are non-conflicting, and check if merging them defines partial tables which are compatible with the $q-i$ other pairs. We consider that picking an elementary tuple T^j encrypting the jth plaintext (irrespective of the ciphertext) corresponds to picking one random input in each of the r round functions. We call this a trial. An input to one round function corresponds to

a ball with a number from 0 to $N-1$. A round function is a bag of N balls. So, we have r bags of balls and a trial consists of picking one ball in each bag. Balls are replaced in their respective bags after picking them. Each T^j makes one trial. Consequently, we have i trials. The balls which are picked during these i trials are called *good balls*. Then, checking compatibility with the remaining $q-i$ pairs corresponds to making $q-i$ additional trials. In those additional trials, we simply look at the number of good balls to see how many rounds can be processed for encryption/decryption.

We estimate the random variable X as the total number of good balls (to which we subtract the r balls corresponding to the trial of T^1). Conditioned to a density of good balls of $\theta_{i,j}$ in round j, we have $E(X|\theta_{i,.}) = \sum_{j=1}^{r} \theta_{i,j} N - r$. All $\theta_{i,j}$ are random, independent, and with expected value θ_i. So, $E(X) = r\theta_i N - r$.

The random variable Y is set to $Y = Y_1 + Y_2 + Y_3$. The variable Y_1 counts the number of modulo N equations so that the encryption of the first i plaintexts match the corresponding ciphertext. So, $Y_1 = 2(i-1)$ (the first pair (M_1, C_1) is satisfied by default, and each of the $i-1$ other ones define two equations due to the two halves of the ciphertexts). The variable Y_2 counts the number of equations coming from pairs encrypted for all but one round. So, Y_2 counts the number of trials (out of the last $q-i$ ones) picking exactly $r-1$ good balls, as they encrypt for all but one round so they define a single equation. The variable Y_3 counts the number of equations coming from pairs in $(M_{i+1}, C_{i+1}), \ldots, (M_q, C_q)$ which are fully encrypted. So, Y_3 is twice the number of trials (out of the last $q-i$ ones) with r good balls, as they fully encrypt their corresponding pair and thus define two equations each. Conditioned to a density of good balls of $\theta_{i,j}$ in round j, we have

$$E(Y|\theta_{i,.}) = \underbrace{2(i-1)}_{Y_1} + \underbrace{(q-i)\sum_{j=1}^{r}(1-\theta_{i,j})\prod_{j'\neq j}\theta_{i,j}}_{Y_2} + \underbrace{2(q-i)\prod_{j}\theta_{i,j}}_{Y_3}.$$

All $\theta_{i,j}$ are random and independent, with expected value θ_i. Thus,

$$E(Y) = 2(i-1) + r\theta_i^{r-1}(1-\theta_i)(q-i) + 2\theta_i^r(q-i).$$

We obtain $|\mathsf{Pool}_i| \approx \mathsf{cns} \times N^{E(X-Y)}$ where cns is adjusted such that $|\mathsf{Pool}_1| = 1$. Hence,

$$|\mathsf{Pool}_i| \approx \mathsf{cns} \times N^{r\theta_i N - r - 2(i-1) - r\theta_i^{r-1}(1-\theta_i)(q-i) - 2\theta_i^r(q-i)} \tag{3}$$

with $\mathsf{cns} \approx 1$ such that $|\mathsf{Pool}_1| = 1$.

To estimate θ_i, we look at how it grows compared to θ_{i-1}. During the i^{th} trial, with probability θ_{i-1} a picked ball is already good (so the density remains the same), and with probability $1-\theta_{i-1}$, picking a ball defines an additional good one (so the density increases by $\frac{1}{N}$).[7] Therefore, on average we have

[7] It would increase with a probability a bit larger than $1-\theta_{i-1}$, namely $\frac{N^2(1-\theta_{i-1})}{N^2-(i-1)}$ if the messages are not independent but conditioned to being pairwise different.

$$\theta_i = \theta_{i-1} + \frac{1}{N} \times (1 - \theta_{i-1}).$$

As $\theta_1 = \frac{1}{N}$, we deduce $\theta_i = 1 - \left(1 - \frac{1}{N}\right)^i$.

Assuming that the above model fits well with Iter, the expected value of $\log|\mathsf{Pool}_i|$ should match Eq. (3). However, Eq. (3) cannot represent well the expected value of $|\mathsf{Pool}_i|$ as exponential with bigger exponents will have a huge impact on the average. This motivates an abort strategy when the pool becomes too big. The abort strategy has known and influenced many works [19]. The way we use this strategy will be discussed in Sect. 3.5.

Finally, the heuristic complexity is computed by

$$T^{\mathrm{Iter}} = \sum_{i=1}^{N} N^{r\theta_i N - 2i - r\theta_i^{r-1}(1-\theta_i)(q-i) - 2\theta_i^r(q-i) - r + 2}. \tag{4}$$

3.3 Approximation of the Complexity

For $i \ll N$, we can write $\theta_i = \frac{i}{N}$. By neglecting θ_i^r against θ_i^{r-1}, the complexity is approximated by the maximum of $N^{r\theta N - 2N\theta - r\theta^{r-1}q - r + 2}$. We can easily show that the maximum is reached by $\theta = \theta_c$ with

$$\theta_c = \left(\frac{r-2}{r(r-1)}\right)^{\frac{1}{r-2}} \left(\frac{N}{q}\right)^{\frac{1}{r-2}}.$$

We obtain the complexity

$$T^{\mathrm{Iter}} \approx N^{\frac{(r-2)^2}{r-1}\left(\frac{r-2}{r(r-1)}\right)^{\frac{1}{r-2}} N\left(\frac{N}{q}\right)^{\frac{1}{r-2}} - r + 2} \tag{5}$$

with q **known plaintexts**. We will see later that (5) approximates well (4).

The best complexity is reached with **the full codebook** $q = N^2$ with

$$T^{\mathrm{Iter}} \approx N^{\frac{(r-2)^2}{r-1}\left(\frac{r-2}{r(r-1)}\right)^{\frac{1}{r-2}} N^{1-\frac{1}{r-2}} - r + 2} \tag{6}$$

which is $T^{\mathrm{Iter}} = N^{\frac{(r-2)^2}{r-1}(\beta + o(1))N^{1-\frac{1}{r-2}}}$ for some $\beta < 1$.

3.4 Iter*: A Chosen Plaintext Extension to Iter

Finally, if q is not too close to N^2, a chosen plaintext attack variant consists of fixing the right half of the plaintext as much as possible, then guessing F_0 on these points and run the known-plaintext attack on $r-1$ rounds to obtain

$$T^{\mathrm{Iter}^*} = N^{\frac{q}{N}-1} T_{r-1}^{\mathrm{Iter}} \approx N^{\frac{q}{N} - 1 + \frac{(r-3)^2}{r-2}\left(\frac{r-3}{(r-1)(r-2)}\right)^{\frac{1}{r-3}} N\left(\frac{N}{q}\right)^{\frac{1}{r-3}} - r + 3} \tag{7}$$

with q **chosen plaintexts** such that $q \leqslant N^2$.

Discussion. For $N^2 > q > N\frac{r-3}{(r-1)(r-2)}\left(2\frac{(r-3)^2}{(r-2)(r-4)}\right)^{r-3} \sim \frac{Ne^3}{r}2^{r-3}$, we have $T^{\text{Iter}^*} < N^{\frac{q}{N}-r+2+\frac{r-4}{N}}$ and that means $T^{\text{Iter}^*} < T^{\text{MITM}^*}$. Therefore, **our** Iter* **algorithm becomes better than** MITM*. Also, for $N \geqslant \frac{(r-3)^{r-2}}{r-1}$, we have $T^{\text{Iter}^*} < N^{N-r+2}$ so Iter* **is faster than exhaustive search on a single round function**.

Optimization with Larger q. We easily obtain that T^{Iter^*} in (7) is optimal with

$$T^{\text{Iter}^*} = N^{\frac{q}{N}-1}T_{r-1}^{\text{Iter}} \approx N^{(r-3)N^{1-\frac{1}{r-2}}\left(\frac{1}{r-1}\right)^{\frac{1}{r-2}}-r+2} \tag{8}$$

for

$$q = \frac{r-3}{r-2}N^{2-\frac{1}{r-2}}\left(\frac{1}{r-1}\right)^{\frac{1}{r-2}}.$$

chosen plaintexts.

3.5 Variants of Iter and Iter*

Optimized Algorithm. We can speed up the algorithm by adding more points in the tuples as soon as we can compute them. Concretely, if one plaintext/ciphertext pair can be "computed" except in one or two rounds, we can deduce the values in the missing rounds and define them in the tuple. Adding x points reduce the number of iterations to define the next pool by N^x.

Abort Strategy. Our complexity is not an average complexity but its logarithm is an average logarithmic complexity. To avoid having a too high average complexity, we may change the algorithm to make it abort if the pool exceeds a threshold to be defined. For instance, if our theoretical formula predicts a complexity Th, to make sure that the worst case complexity does not exceed $\mathsf{Th} \times N^x$, we set this to the threshold value. This will affect the success probability, which is 100% without the abort strategy, but may be lower for any real number x.

Other Improvements. We believe that we could improve our algorithms in many ways. For instance, we could take the (M_i, C_i) pairs in an optimized order so that we do not have too many new values appearing in the first and last round functions. This would decrease the number of tuples to consider.

3.6 Experimental Results

We implemented Algorithm 3 with known plaintext, $r = 5$, $N = 8$, $q = 40$. Our algorithm always ended with a pool limited to a correct set of full tables.

With these parameters, Eq. (3) estimates Pool_3 to be the largest, and estimates $|\mathsf{Pool}_3| = N^{2.49}$. We checked over 100 executions, that $\log_N |\mathsf{Pool}_3|$ has an average of 4.37 and a standard deviation of 0.60. This is a bad news as it is quite larger than what is predicted. More precisely, each partial function in Pool_3 has

on average 2.9 defined entries, which is slightly more than the $N\theta_3 \approx 2.64$ which is predicted.[8] But adjusting θ_3 to $\frac{2.9}{N}$ in Eq. (3) gives $N^{3.04}$, which is not enough to explain the high $|\mathsf{Pool}_3|$ which is observed. So, our model for the random variable X may be correct but Y may be overestimated: Iter decimates less than expected. Although we thought Pool_3 would be the largest from our theory, the largest observed pool during our experiment were Pool_4 with logarithmic size with average 5.28. This indicates that our model for Iter is not accurate.

All these problems find several explanations. First of all, our parameter N is so small that a tiny variation of number of defined entries (supposed to be $\theta_i N$) in each round has a dramatic impact on the number of tuples. Second, our approach takes the θ_i as uniform in all rounds and runs although there are variations. Some rounds have more than $\theta_i N$ entries and some others have less. The function we analyze is not linear in θ_i. It is exponential. So, any round with more than $\theta_i N$ defined entries increase the complexity quite a lot.

The good news is that using our optimized variant reduced the gap substantially. The largest Pool becomes $\max_i \log_N(|\mathsf{Pool}_i|) = 3.46$. Using the abort strategy with $x = 1$ gives a success rate of 42% and $\max_i \log_N(|\mathsf{Pool}_i|) = 3.08$. So, we believe that **our anticipated complexities are achievable with a good success probability**. However, finding a good model for decimation and for the improved algorithm remains an open question.

We summarize our experiments in the Table 2. For the $\max|\mathsf{Pool}|$ column is the average (logarithmically) of the largest observed pool. The logarithm is the maximum over each iteration of the average over the runs of the logarithm of the pool size. The computed average only includes successful runs, as unsuccessful ones are all on the abort threshold.

4 Applications

In the standards, the supported domain size of messages in FF1 and FF3* is greater than 100 (i.e. $N^2 \geqslant 100$). For FF1 and FF3*, the best attack is roughly Iter^* for very low N, then MITM^* for larger N. More precisely, we achieve the results shown in Table 3.[9]

For a better precision, we did the computation without approximations, i.e. by using Eq. (4) instead of Eq. (5) in Eq. (7). In any case, we have checked that the figures with approximation do not differ much. They are reported in the Table 4.

As an example, for FF3* with $N = 2^3$ (i.e., messages have 6 bits), MITM^* uses $q = 2^5$ pairs (half of the codebook) and search on three points for F_0, the entire (but one point) F_1 and F_2, one bit of F_3 in the encryption direction, and the entire (but one point) F_7 and F_6 and one bit of F_5 in the decryption direction. This is $N^{3+2(N-1)} \times 2^{N-1} = 2^{58}$. With Iter^*, we also use $q = 2^5$ and the pool reaches its critical density for $\theta_c \approx \frac{4.4}{N}$. The complexity is $T^{\mathrm{Iter}^*} = 2^{42}$.

[8] This is partially explained by the fact that plaintexts are pairwise different.

[9] Note that the standard requires $N \geqslant 10$. Hence, the first three rows are not relevant in practice.

Table 2. Experimental results with parameters $\mathsf{r} = 5$, $\mathsf{N} = 8$, and $\mathsf{q} = 40$ and with parameters $\mathsf{r} = 5$, $\mathsf{N} = 10$, and $\mathsf{q} = 40$. The $\max|\mathsf{Pool}|$ column reports $\max_{\mathsf{i}} \mathsf{E}_{\mathsf{runs}}(\log_{\mathsf{N}} |\mathsf{Pool}_{\mathsf{i}}|)$: the average (logarithmically) of the largest observed pool. It is compared with Th which is derived as the largest theoretical pool size by our theory. The column opt shows whether we used the optimization trick. The abort column indicates when we used the abort strategy, and with which bound.

$\mathsf{r} = 5$, $\mathsf{N} = 8$, $\mathsf{q} = 40$				
#runs	success	$\max\|\mathsf{Pool}\|$	opt	abort
100	100%	$\mathsf{Th} \times \mathsf{N}^{2.79}$	no	no
10 000	0%		no	Th
1 000	0%		no	$\mathsf{Th} \times \mathsf{N}$
1 000	3%	$\mathsf{Th} \times \mathsf{N}^{1.76}$	no	$\mathsf{Th} \times \mathsf{N}^2$
100	100%	$\mathsf{Th} \times \mathsf{N}^{0.93}$	yes	no
10 000	1%	$\mathsf{Th} \times \mathsf{N}^{-0.29}$	yes	Th
100	42%	$\mathsf{Th} \times \mathsf{N}^{0.59}$	yes	$\mathsf{Th} \times \mathsf{N}$
100	99%	$\mathsf{Th} \times \mathsf{N}^{0.90}$	yes	$\mathsf{Th} \times \mathsf{N}^2$

$\mathsf{r} = 5$, $\mathsf{N} = 10$, $\mathsf{q} = 40$				
#runs	success	$\max\|\mathsf{Pool}\|$	opt	abort
10 000	0%		no	Th
1 000	0%		no	$\mathsf{Th} \times \mathsf{N}$
100	0%		no	$\mathsf{Th} \times \mathsf{N}^2$
14	100%	$\mathsf{Th} \times \mathsf{N}^{1.40}$	yes	no
10 000	1%	$\mathsf{Th} \times \mathsf{N}^{-0.31}$	yes	Th
100	19%	$\mathsf{Th} \times \mathsf{N}^{0.60}$	yes	$\mathsf{Th} \times \mathsf{N}$
19	68%	$\mathsf{Th} \times \mathsf{N}^{1.25}$	yes	$\mathsf{Th} \times \mathsf{N}^2$

Table 3. Time complexity of the chosen-plaintext attacks MITM^* ($\mathsf{T}^{\mathsf{MITM}^*}$) and Iter^* ($\mathsf{T}^{\mathsf{Iter}^*}$) with query complexity q for various values of N and $\mathsf{r} = 8$ or $\mathsf{r} = 10$. Computations for $\mathsf{T}^{\mathsf{Iter}^*}$ were done without using approximations.

$\mathsf{r} = 8$ (FF3*)				$\mathsf{r} = 10$ (FF1)			
N	$\mathsf{T}^{\mathsf{MITM}^*}[\mathsf{q}]$	(2)	$\mathsf{T}^{\mathsf{Iter}^*}[\mathsf{q}]$	(8)	N	$\mathsf{T}^{\mathsf{MITM}^*}[\mathsf{q}]$	(2)	$\mathsf{T}^{\mathsf{Iter}^*}[\mathsf{q}]$	(8)
2^1	$2^6[\ 2^{2.0}\ ]$		$\mathbf{2^2}[2^{2.0}]$		2^1	$2^8[\ 2^{2.0}\ ]$		$\mathbf{2^3}[\ 2^{2.0}\ ]$	
2^2	$2^{24}[\ 2^{4.0}\ ]$		$\mathbf{2^{13}}[2^{4.0}]$		2^2	$2^{32}[\ 2^{4.0}\ ]$		$\mathbf{2^{21}}[\ 2^{4.0}\ ]$	
2^3	$2^{72}[\ 2^{5.0}\ ]$		$\mathbf{2^{42}}[2^{5.0}]$		2^3	$2^{96}[\ 2^{5.3}\ ]$		$\mathbf{2^{72}}[\ 2^{5.3}\ ]$	
2^4	$2^{192}[\ 2^{6.0}\ ]$		$\mathbf{2^{116}}[2^{6.6}]$		2^4	$2^{256}[\ 2^{6.3}\ ]$		$\mathbf{2^{199}}[\ 2^{6.8}\ ]$	
2^5	$2^{480}[\ 2^{7.0}\ ]$		$\mathbf{2^{279}}[2^{8.3}]$		2^5	$2^{640}[\ 2^{7.3}\ ]$		$\mathbf{2^{487}}[\ 2^{8.6}\ ]$	
2^6	$2^{1152}[\ 2^{8.0}\ ]$		$\mathbf{2^{627}}[2^{10.1}]$		2^6	$2^{1536}[\ 2^{8.3}\ ]$		$\mathbf{2^{1115}}[\ 2^{10.5}\ ]$	
2^7	$2^{2688}[\ 2^{9.0}\ ]$		$\mathbf{2^{1343}}[2^{12.0}]$		2^7	$2^{3584}[\ 2^{9.3}\ ]$		$\mathbf{2^{2445}}[\ 2^{12.4}\ ]$	
2^8	$2^{6144}[\ 2^{10.0}\ ]$		$\mathbf{2^{2788}}[2^{13.8}]$		2^8	$2^{8192}[\ 2^{10.3}\ ]$		$\mathbf{2^{5202}}[\ 2^{14.3}\ ]$	

We may wonder for which N the ciphers offer a 128-bit security. Durak and Vaudenay [15] showed that this is not the case for FF3* with $\mathsf{N} \leqslant 10$ and FF1 with $\mathsf{N} \leqslant 7$. By doing computations for Iter^*, we extend this to show that **FF3***

Table 4. Time complexity of the chosen-plaintext attacks MITM* ($\mathsf{T}^{\mathrm{MITM}^*}$) and Iter* ($\mathsf{T}^{\mathrm{Iter}^*}$) with query complexity q for various values of N and $\mathsf{r} = 8$ or $\mathsf{r} = 10$. Computations for $\mathsf{T}^{\mathrm{Iter}^*}$ were done using approximations.

$\mathsf{r} = 8$ (FF3*)				$\mathsf{r} = 10$ (FF1)			
N	$\mathsf{T}^{\mathrm{MITM}^*}[\mathsf{q}]$ (2)		$\mathsf{T}^{\mathrm{Iter}^*}[\mathsf{q}]$ (8)	N	$\mathsf{T}^{\mathrm{MITM}^*}[\mathsf{q}]$ (2)		$\mathsf{T}^{\mathrm{Iter}^*}[\mathsf{q}]$ (8)
2^1	$2^6[$	$2^{2.0}$]	$\mathbf{2^1}[2^{2.0}]$	2^1	$2^8[$	$2^{2.0}$]	$\mathbf{2^2}[$ $2^{2.0}$]
2^2	$2^{24}[$	$2^{4.0}$]	$\mathbf{2^{13}}[2^{4.0}]$	2^2	$2^{32}[$	$2^{4.0}$]	$\mathbf{2^{21}}[$ $2^{4.0}$]
2^3	$2^{72}[$	$2^{5.0}$]	$\mathbf{2^{44}}[2^{5.0}]$	2^3	$2^{96}[$	$2^{5.3}$]	$\mathbf{2^{75}}[$ $2^{5.3}$]
2^4	$2^{192}[$	$2^{6.0}$]	$\mathbf{2^{122}}[2^{6.6}]$	2^4	$2^{256}[$	$2^{6.3}$]	$\mathbf{2^{209}}[$ $2^{6.9}$]
2^5	$2^{480}[$	$2^{7.0}$]	$\mathbf{2^{295}}[2^{8.4}]$	2^5	$2^{640}[$	$2^{7.3}$]	$\mathbf{2^{512}}[$ $2^{8.8}$]
2^6	$2^{1152}[$	$2^{8.0}$]	$\mathbf{2^{658}}[2^{10.3}]$	2^6	$2^{1536}[$	$2^{8.3}$]	$\mathbf{2^{1166}}[$ $2^{10.7}$]
2^7	$2^{2688}[$	$2^{9.0}$]	$\mathbf{2^{1401}}[2^{12.1}]$	2^7	$2^{3584}[$	$2^{9.3}$]	$\mathbf{2^{2543}}[$ $2^{12.5}$]
2^8	$2^{6144}[$	$2^{10.0}$]	$\mathbf{2^{2890}}[2^{13.9}]$	2^8	$2^{8192}[$	$2^{10.3}$]	$\mathbf{2^{5383}}[$ $2^{14.4}$]

does not offer a 128-bit security for $\mathsf{N} \leqslant 17$, and FF1 does not offer a 128-bit security for $\mathsf{N} \leqslant 11$.

Genuinely, we can compute in Table 5 the minimum $\mathsf{r}_{\mathtt{opt}} \geqslant 4$ of the number of rounds for which $\min(\mathsf{T}^{\mathrm{MITM}^*}, \mathsf{T}^{\mathrm{Iter}^*}) \geqslant 2^{\mathsf{s}}$ depending on s and N. Again, we computed without using our approximations. For $\mathsf{s} = 128$ and $\mathsf{s} = 256$, we fetch the following table.[10]

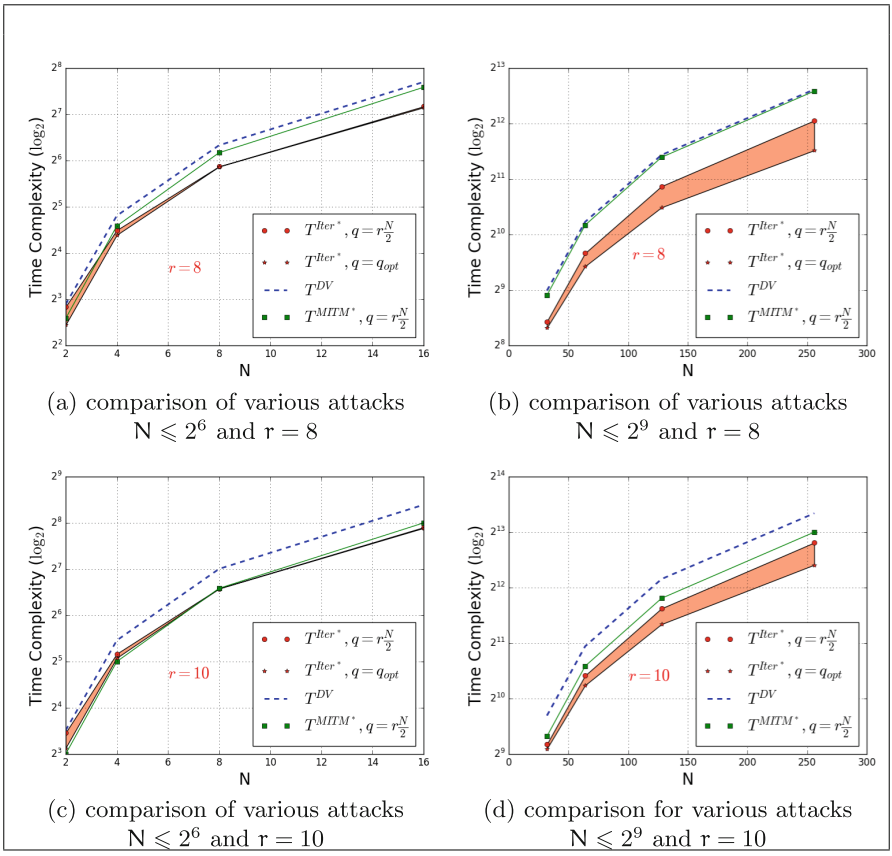
Table 5. Minimal number $\mathsf{r}_{\mathtt{opt}}$ of rounds for various N in order to have complexities at least 2^{128} or 2^{256}. Computations for $\mathsf{T}^{\mathrm{Iter}^*}$ were done without using approximations.

$\mathsf{s} = 128$				$\mathsf{s} = 256$			
N	$\mathsf{r}_{\mathtt{opt}}$	$\mathsf{T}^{\mathrm{MITM}^*}$	$\mathsf{T}^{\mathrm{Iter}^*}$	N	$\mathsf{r}_{\mathtt{opt}}$	$\mathsf{T}^{\mathrm{MITM}^*}$	$\mathsf{T}^{\mathrm{Iter}^*}$
2^1	260	$2^{258.0}$	$\mathbf{2^{128.5}}$	2^1	516	$2^{514.0}$	$\mathbf{2^{256.5}}$
2^2	40	$2^{152.0}$	$\mathbf{2^{129.3}}$	2^2	77	$\mathbf{2^{228.0}}$	$2^{257.6}$
2^3	14	$2^{144.0}$	$\mathbf{2^{136.5}}$	2^3	24	$\mathbf{2^{264.0}}$	$2^{272.2}$
2^4	9	$2^{240.0}$	$\mathbf{2^{155.8}}$	2^4	12	$2^{320.0}$	$\mathbf{2^{289.1}}$
2^5	7	$2^{465.0}$	$\mathbf{2^{187.9}}$	2^5	8	$2^{480.0}$	$\mathbf{2^{279.3}}$
2^6	6	$2^{768.0}$	$\mathbf{2^{236.2}}$	2^6	7	$2^{1134.0}$	$\mathbf{2^{415.8}}$
2^7	5	$2^{1778.0}$	$\mathbf{2^{195.4}}$	2^7	6	$2^{1792.0}$	$\mathbf{2^{485.0}}$
2^8	5	$2^{4080.0}$	$\mathbf{2^{370.4}}$	2^8	5	$2^{4080.0}$	$\mathbf{2^{370.4}}$

Even by adding a safety margin, this shows that we do not need many rounds to safely encrypt a byte (that is, $\mathsf{N} = 2^4$) with respect to our best attacks. However, with low r, we should care about other attacks as in Table 1. Indeed, for $\oplus$-FN, we recommend never to take $\mathsf{r} \leqslant 7$ due to the yo-yo attack [7]. For other FN, we recommend never to take $\mathsf{r} \leqslant 5$.

[10] In this table, we computed the value of q suggested by our formulas but rounded in the $\left[\frac{\mathsf{r}\mathsf{N}}{2}, \mathsf{N}^2\right]$ interval.

In Fig. 4, we plot complexities for $\mathtt{r} = 8$ or $\mathtt{r} = 10$ and various ranges of $\mathtt{N}$. The regions for $\mathsf{T}^{\mathtt{Iter}^*}$ we plot have a minimum for the optimal $\mathtt{q}$ and a maximum for $\mathtt{r} = \frac{\mathtt{rN}}{2}$. The region corresponds to all complexities for $\mathtt{q} \in [\frac{\mathtt{rN}}{2}, \mathtt{N}^2]$.



(a) comparison of various attacks $\mathtt{N} \leqslant 2^6$ and $\mathtt{r} = 8$

(b) comparison of various attacks $\mathtt{N} \leqslant 2^9$ and $\mathtt{r} = 8$

(c) comparison of various attacks $\mathtt{N} \leqslant 2^6$ and $\mathtt{r} = 10$

(d) comparison for various attacks $\mathtt{N} \leqslant 2^9$ and $\mathtt{r} = 10$

Fig. 4. Time complexity of attacks against generic 8-round and 10-round FN for $\mathtt{Iter}^*$ with $\mathtt{q}$ minimal or $\mathtt{q}$ making the complexity optimal, for DV [15], and MITM^*.

5 Conclusion

Standard Feistel Networks and its variations have created an active research area since their invention and have been used in constructions of many cryptographic systems to a wide extent. The security of FN has been studied for many decades resulting in many interesting results for cryptanalysis purpose. In this work, we analyze the security of a very specific type of FN with two branches, secure random round functions, and modular addition to analyze its security. Additionally, we consider small domains. The best attack was believed to be MITM. However,

we show that partial exhaustive search can be better. Concretely, we show that the number of rounds recommended by NIST is insufficient in FF1 and FF3* for very small N.

This specific FN with the described properties has been used to build Format-Preserving Encryption and perhaps will inspire many other constructions. However, the security of FN with various properties is not clear (regardless of the significant security analyses mentioned in the introduction) and has to be investigated more. Our work shows only that a caution should be taken in order to meet the desired security level in the systems.

We proposed a new algorithm based on partial exhaustive search. We observed a gap between our heuristic complexity and experiments and suggested possible explanations. However, the problem to reduce this gap is left as an open problem.

References

1. Data Encryption Standard, National Bureau of Standards, NBS FIPS PUB 46, January 1977. National Bureau of Standards. U.S, Department of Commerce (1977)
2. Recommendation for Block Cipher Modes of Operation: Methods for Format Preserving Encryption, NIST Special Publication (SP) 800-38G, 29 March 2016. National Institute of Standards and Technology
3. Retail Financial Services - Requirements for Protection of Sensitive Payment Card Data - Part 1: Using Encryption Method. American National Standards Institute (2016)
4. Bellare, M., Hoang, V.T., Tessaro, S.: Message-recovery attacks on Feistel-based format-preserving encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 444–455. ACM, New York (2016)
5. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-preserving encryption. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 295–312. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05445-7_19
6. Bellare, M., Rogaway, P., Spies, T.: The FFX Mode of Operation for Format-Preserving Encryption. draft 1.1. Submission to NIST, February 2010. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec.pdf
7. Biryukov, A., Leurent, G., Perrin, L.: Cryptanalysis of feistel networks with secret round functions. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 102–121. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31301-6_6
8. Biryukov, A., Perrin, L.: On reverse-engineering S-boxes with hidden design criteria or structure. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 116–140. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_6
9. Brier, E., Peyrin, T., Stern, J.: BPS: A Format-Preserving Encryption Proposal. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf
10. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6), 74–84 (1977)

11. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 719–740. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_42
12. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: New attacks on feistel structures with improved memory complexities. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 433–454. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_21
13. Durak, F.B., Vaudenay, S.: Breaking the FF3 format-preserving encryption. In: Proceedings of ESC 2017. https://www.cryptolux.org/mediawiki-esc2017/images/8/83/Proceedings_esc2017.pdf
14. Durak, F.B., Vaudenay, S.: Generic Round-Function-Recovery attacks for Feistel Networks over Small Domains. https://eprint.iacr.org/2018/108.pdf
15. Durak, F.B., Vaudenay, S.: Breaking the FF3 format-preserving encryption standard over small domains. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 679–707. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_23
16. Hoang, V.T., Rogaway, P.: On generalized Feistel networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 613–630. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_33
17. Isobe, T., Shibutani, K.: All subkeys recovery attack on block ciphers: extending meet-in-the-middle approach. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 202–221. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_14
18. Isobe, T., Shibutani, K.: Generic key recovery attack on Feistel scheme. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 464–485. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_24
19. Lu, J., Kim, J., Keller, N., Dunkelman, O.: Improving the efficiency of impossible differential cryptanalysis of reduced Camellia and MISTY1. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 370–386. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79263-5_24
20. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Comput. **17**(2), 373–386 (1988)
21. Nachef, V., Volte, E., Patarin, J.: Differential attacks on generalized Feistel schemes. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 1–19. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02937-5_1
22. Patarin, J.: Generic attacks on Feistel schemes (2008). http://eprint.iacr.org/2008/036
23. Patarin, J.: Security of Balanced and Unbalanced Feistel Schemes with Non-linear Equalities (2010). http://eprint.iacr.org/2010/293
24. Patarin, J., Nachef, V., Berbain, C.: Generic attacks on unbalanced Feistel schemes with contracting functions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 396–411. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_26
25. Vaudenay, S.: The security of DSA and ECDSA. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 309–323. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36288-6_23