Generic Round-Function Recovery for Feistel Networks over Small Domains

F. Betül Durak and Serge Vaudenay

Ecole Polytechnique Fédérale de Lausanne (EPFL)
LASEC - Security and Cryptography Laboratory
Lausanne, Switzerland

Abstract. Feistel Networks (FN) are now massively being used 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 standards FF1 and FF3 from NIST and ANSI standards 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 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. Fig. 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) consist of 16 decimal numbers, of which 6 must be kept confidential. For this reason, these 6 numbers 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 [14]. Herein, we denote by FF3* the repaired scheme.

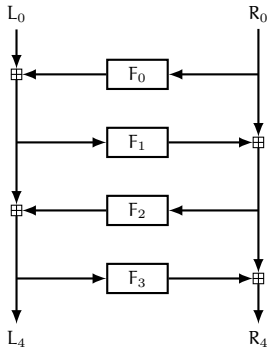


Fig. 1: 4-round Feistel network

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, 15, 17, 20, 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 [19]. In their seminal paper, Luby and Rackoff first showed that a three

[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.

round Feistel construction is a secure pseudorandom permutation from $2n$ bits to $2n$ bits. Moreover, they showed that for more than three rounds FN, all generic CPA 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 $N_l \approx N_r$.

rounds	method	type	requirement	time complexity T	data q	ref
3	yo-yo	known pt		$O(N \ln N)$	$N \ln N$	[14]
4	cycle finding	known pt		$O(N^3)$	$N^{\frac{3}{2}}$	[14]
4	guess and determine	chosen pt		$O\left(N^{\frac{3}{2}}\right)$	$N^{\frac{3}{2}}$	[7]
5	cycle finding	chosen pt		$O\left(N^{\sqrt{N}+3}\right)$	$N^{\frac{3}{2}}$	[14]
5	integral attack	chosen pt	F_1 or F_3 invertible	$O(N^{2.81})$	N^2	[7]
5	yo-yo	full codebook	$\oplus$-Feistel	$O(N^2)$	N^2	[7]
5	guess and determine	full codebook		$O\left(N^{N^{\frac{3}{4}}}\right)$	N^2	[7]
5	SAT solver	full codebook		not specified	N^2	[8]
6	yo-yo	full codebook	$\oplus$-Feistel	$O\left(N^{\frac{1}{2}N}\right)$	N^2	[7]
7	yo-yo	full codebook	$\oplus$-Feistel	$O\left(N^N\right)$	N^2	[7]
r	cycle finding	chosen pt		$O\left(N^{(r-5)N+\sqrt{N}+3}\right)$	$N^{\frac{3}{2}}$	[14]
r	MITM	known pt		$O\left(N^{\lceil \frac{r}{2} \rceil N}\right)$	$r\frac{N}{2}$	Eq. (1), Sec. 2.1
r	MITM*	chosen pt		$N^{\frac{r-3}{2}N(1+o(1))}$	$r\frac{N}{2}$	Eq. (2), Sec. 2.2
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), Sec. 3.1
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), Sec. 3.1
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), Sec. 3.1

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

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 $q = \beta N^{2-\frac{1}{r-2}}$ plaintexts.

We plot on 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 minimum of 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.

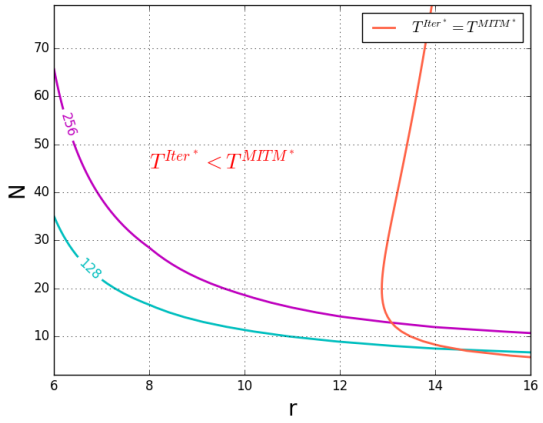


Fig. 2: Parameters (r, N) for $T^{Iter^*} = T^{MITM^*}$ (in red) and parameters to meet a 128-bit and a 256-bit security level.

As an application, we show that the number of rounds in FF3* and FF1 is insufficient to offer a 128-bit security (as they are supposed to) for $N \leqslant 17$

and $N \leqslant 11$, respectively.[2] (E.g., for 6-bit messages of 2-digit messages.) Other applications could be found to reverse engineer an S-box based on FN [8].

2 Preliminaries

In this section we present known techniques to recover the round functions in FN. Note that a round function can be reconstructed up to a constant [13,14]. It means that if $F_0, \ldots, F_{r-1}$ is a solution to map a set of plaintexts to corresponding ciphertexts, then 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 $b_0 + b_2 + b_4 + \cdots = b_1 + b_3 + b_5 + \cdots = 0$, then $F'_0, \ldots, F'_{r-1}$ is a solution for $F'_i(x) = F_i(x - a_i) + b_i$. Therefore, we can fix arbitrarily one point of $F_0, \ldots, F_{r-3}$.

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

The MITM attack was introduced by Diffie and Hellman [10]. MITM is a generic known-plaintext attack. Briefly, consider an r round encryption $E_1, E_2, \ldots, E_r$ and corresponding $D_1, D_2, \ldots, D_r$ decryption algorithms with keys $K_1, K_2, \ldots, K_r$ of length k. Let $P_1, P_2, \ldots, P_q$ be the plaintexts and $C_1, C_2, \ldots, C_q$ be the corresponding ciphertexts. Let the intermediate values after i^{th} round be $P_1^i, P_2^i, \ldots, P_q^i$ for $1 \leqslant i < r$. The adversary enumerates each possible combination of the keys $K_1, K_2, \ldots, K_u$ for the first $u = \lfloor \frac{r}{2} \rfloor$ rounds and it computes the intermediate values for each plaintexts as $P_1^u, P_2^u, \ldots, P_q^u$ until u^{th} round. 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 $K_r, K_{r-1}, \ldots, K_{u+1}$ 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 $K_1, K_2, \ldots K_r$ 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}.[3]

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 so we can focus on a generic FN with functions specified by tables. This is equivalent to 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**.

[2] It was shown by Durak and Vaudenay [14] that is was not the case for $7 \leqslant N \leqslant 10$ and $N = 7$, respectively. We improve the result with more N and lower complexity. Note that the NIST standard [2] requires $N \geqslant 10$.

[3] 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].

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 [16,17] 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 $M_R^{(0)} = a$ for a packet of N plaintexts. We have $\frac{q}{N}$ packets thus $\frac{q}{N}$ values for a. We set the output of F_0 for one value of a arbitrarily. For all the values of M_L^0, we query $(M_L^0 \| a)$ and obtain N $(C_L \| C_R)$ 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 $(M_L^{(1)} \| M_R^{(1)})$. 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 stored Z values (since they have to be equal). The complexity of this approach consists of enumerating N^{vN} many v-round function 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, which is lower than the current complexity $N^{\frac{q}{N}-1+(u+v)(N-1)}$. 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 which is lower than the MITM itself.

More precisely, the attack works with $q = \frac{rN}{2}$ known plaintext/ciphertext pairs. We enumerate all the possible values for $M_L^{(0)}$ and pick $\frac{q}{N} = \frac{r}{2}$ arbitrary $M_R^{(0)} = a$. Among the $\frac{r}{2}$ possible $M_R^{(0)}$, we can fix one of them arbitrarily, therefore, we can guess only $\frac{r}{2} - 1$ outputs of F_0.

In this attack, we have to guess $N^{\frac{q}{N}-1}$ values for F_0, $N^{u(N-1)}$ values (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 $O\left(N^{\frac{r-1}{2}(N-1)}\right)$ for r is odd. We label the time complexity for described attack as:

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

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

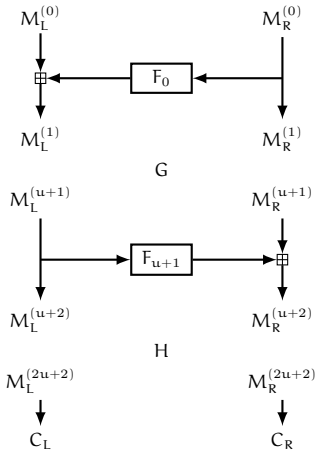


Fig. 3: (2u+2)-round Feistel Network (with u even on the picture)

3 Round-Function-Recovery by Partial Exhaustive Search

We consider exhaustive search algorithms dealing with partial functions. Normally, a function F_i is defined by its set of all possible $(z, F_i(z))$ pairs. We call *partial table* any subset of its table. 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. 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_i denotes the partial table of F_i in T. We say T is homogeneous with density θ if for all i, T_i has density θ. Similarly, a tuple T' is an extension of T if for each i, T'_i is an extension of T_i. 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.

We say that a tuple T *encrypts* a plaintext M into a ciphertext C (or *decrypts* C into M or even that T encrypts the pair (M, C)) if we can evaluate the FN on M with the partial information we have about the round functions and if it gives C. We say that a pair (M, C) is *computable except for* r' *rounds* for a tuple T if the partial functions are enough to encrypt M for up to i rounds and to decrypt C for up to $r - i - r'$ rounds.

We say a tuple T of partial tables is *compatible* with (M, C) if one of the following conditions is satisfied: (i) T encrypts M into C; (ii) (M, C) is computable except for two rounds or more; (iii) (M, C) is computable except for one round and the half of the encryption of M that can be computed after $i + 1$ round matches the respective half of the decryption of y after $r - i - 1$ rounds. Clearly, if T is not compatible with (M, C), then no extension of T can encrypt M into C so we can prune an exhaustive search.

3.1 Iter: Iterative Partial Exhaustive Search

Assume that $\mathtt{q}$ plaintext/ciphertext pairs $(\mathsf{M}_\mathtt{i}, \mathsf{C}_\mathtt{i})$ are known to the adversary. We iteratively construct a pool of tuples which are compatible with all pairs. To construct $\mathtt{Pool}_\mathtt{i}$, we take all possible minimal extensions of tuples from $\mathtt{Pool}_{\mathtt{i}-1}$ which encrypt the $\mathtt{i}$th pair and remain compatible with all others. We proceed as defined by Algorithm 1.

Algorithm 1: 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 elementary tuple T^i which encrypts M_i to C_i do
8  |  |  |  Set T' with T'_j = T_j ∪ T^i_j, j = 0, ..., r-1. (Extend T with T^i.)
9  |  |  |  if T' is a valid extension of T then
10 |  |  |  |  if all (M_{i+1}, C_{i+1}), ..., (M_q, C_q) are compatible with T' then
11 |  |  |  |  |  Add T' in Pool_i.
12 |  |  |  |  end
13 |  |  |  end
14 |  |  end
15 |  end
16 end
17 Output Pool_q.
```

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 homogeneous tuple T_1 of density $\frac{1}{\mathtt{N}}$ which encrypts the pair $(\mathsf{M}_1, \mathsf{C}_1)$. So, we define $\mathtt{Pool}_\mathtt{i}$ as the set of all *minimal extensions* (in the sense that removing any entry in the partial table cancels the property) of T_1 encrypting the pairs $(\mathsf{M}_1, \mathsf{C}_1), \ldots, (\mathsf{M}_\mathtt{i}, \mathsf{C}_\mathtt{i})$ which are compatible with all other pairs.

In practice, instead of enumerating all elementary tuples $\mathsf{T}^\mathtt{i}$ then checking compatibility with T, we can limit ourselves to the enumeration of all compatible elementary tuples. With an appropriate data structure, we can also avoid to retry to encrypt $\mathsf{M}_\mathtt{j}$ or decrypt $\mathsf{C}_\mathtt{j}$ and directly go to the next computable round in every pair if any. This saves the inner loop. For each tuple T in $\mathtt{Pool}_\mathtt{i}$, we maintain a hash table $\mathtt{h}$ in which $\mathtt{h}(\mathtt{u}, \mathtt{x})$ is a list of pairs of the form $(\mathtt{j}, +)$ or $(\mathtt{j}, -)$ with $\mathtt{j} > \mathtt{i}$. If $(\mathtt{j}, +)$ is in $\mathtt{h}(\mathtt{u}, \mathtt{x})$, this means that T encrypts $\mathsf{M}_\mathtt{j}$ up to round $\mathtt{u} - 1$ and that the input to $\mathsf{F}_\mathtt{u}$ (the output of which is unknown) is $\mathtt{x}$. If $(\mathtt{j}, -)$ is in $\mathtt{h}(\mathtt{u}, \mathtt{x})$, this means that T decrypts $\mathsf{M}_\mathtt{j}$ up to round $\mathtt{u} + 1$ and that the input to $\mathsf{F}_\mathtt{u}$ is $\mathtt{x}$. Concretely, this means that $\mathtt{h}(\mathtt{u}, \mathtt{x})$ lists the indices of

(M_j, C_j) pairs who need the value of $F_u(x)$. With these algorithmic tricks, the complexity is expected to be close to the total size of the pools.

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 pairs, restricted to the ones which are compatible with all others. That is, for all set $(t_2, \ldots, t_i)$ of elementary tuples in which t_j encrypts the jth pair, we check if $\{T^1, t_2, \ldots, t_i\}$ are non-conflicting, and if merging them defines partial tables which are compatible with the $q - i$ other pairs. We approximate $|\mathtt{Pool}_i|$ by N^{X-Y} where X is the number of free entries in the partial functions (i.e. the number of defined points throughout all rounds) and Y is the number of free equations over N-values which a tuple must satisfy to be compatible so that 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 T^1 only in $\mathtt{Pool}_1$, we subtract r to X (that is, entries defined in T^1 are fixed so not free) and we subtract 2 to Y (i.e., we consider that the first pair never decimates tuples as it is always compatible by the choice of T^1).

We modeled the structure of $\mathtt{Pool}_i$ as follows. We consider that picking an elementary tuple t_j encrypting the jth plaintext (irrespective of the ciphertext) corresponds to picking one random input to 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. 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, and simply looking at the number of good balls in each of these trials 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 and independent, 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 equations so that the encryption of the first i plaintexts match the corresponding ciphertext. So, $Y_1 = 2(i-1)$ (the first pair 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 trials (out of the last $q - i$ ones) with $r - 1$ good balls, as they encrypt for all but one round so they define a single equation. The variable Y_3 counts twice the number of trials (out of the last $q - i$ ones) with r good balls, as they fully encrypt the pair so they define two equations each. Conditioned to a density of good balls of $\theta_{i,j}$ in round j, we have $E(Y|\theta_{i,.}) = 2(i-1) + (q-i)\sum_{j=1}^{r}(1-\theta_{i,j})\prod_{j' \neq j}\theta_{i,j} + 2(q-i)\prod_j \theta_{i,j}$. 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 $\mathsf{cns} \approx 1$.

To estimate θ_i, we look at how it grows compared to θ_{i-1}. During the ith 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 $\frac{1}{N}$).[4] Therefore, on average we have

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

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 $\mathtt{Iter}$, 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 [18]. The way we use this strategy will be discussed in Section 3.5.

Finally, the heuristic complexity is

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

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 $\beta < 1$.

[4] 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.

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 $T^{\mathrm{Iter}^*} < N^{\frac{q}{N}-r+2+\frac{r-4}{N}}$ and that means $T^{\mathrm{Iter}^*} < T^{\mathrm{MITM}^*}$. Therefore, **our Iter* algorithm becomes better than MITM*.** Also, for $N \geqslant \frac{(r-3)^{r-2}}{r-1}$, we have $T^{\mathrm{Iter}^*} < N^{N-r+2}$ 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

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

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 has a right average. 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 $Th \times N^x$, 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 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 1 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 with $|\mathsf{Pool}_3| = N^{2.49}$. 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 predicted. More precisely, each partial function in Pool_3 has on average 2.9 defined entries, which is slightly more than the $N\theta_3 \approx 2.64$ predicted.[5] 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 of 1 (out of N) 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 and the function we analyze is not linear in θ_i.

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.

(Note that the standard requires $N \geqslant 10$ so the first three rows are not relevant in practice.) 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.

[5] This partially explains by the fact that plaintexts are pairwise different.

$\mathsf{r}=5$, $\mathsf{N}=8$, $\mathsf{q}=40$				
#runs	success	$\max\lvert\mathsf{Pool}\rvert$	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	$\max\lvert\mathsf{Pool}\rvert$	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 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 $\max\lvert\mathsf{Pool}\rvert$ column reports $\max_{\mathsf{i}}\mathsf{E}_{\mathsf{runs}}(\log_{\mathsf{N}}\lvert\mathsf{Pool}_{\mathsf{i}}\rvert)$: 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.

$\mathsf{r}=8$ (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	$2^6[\ 2^{2.0}\ ]$		$\mathbf{2^2}[2^{2.0}]$		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	$2^{8192}[\ 2^{10.3}\ ]$		$\mathbf{2^{5202}}[\ 2^{14.3}\ ]$	

Table 3: Complexity T of the chosen-plaintext attacks MITM^* and 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 without using aproximations.

$r = 8$ (FF3*)				$r = 10$ (FF1)			
N	$T^{MITM^*}[q]$	(2)	$T^{Iter^*}[q]$ (8)	N	$T^{MITM^*}[q]$	(2)	$T^{Iter^*}[q]$ (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	$2^{8192}[\ 2^{10.3}\ ]$		$\mathbf{2^{5383}}[\ 2^{14.4}\ ]$

Table 4: Complexity T of the chosen-plaintext attacks MITM* and Iter* with query complexity q for various values of N and $r = 8$ or $r = 10$. Computations for T^{Iter^*} were done using aproximations.

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 the same parameters, we have $T^{DV} = 2^{89}$ with the algorithm from App B.) 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 $T^{Iter^*} = 2^{42}$.

We may wonder for which N the ciphers offer a 128-bit security. Durak and Vaudenay [14] showed that this is not the case for FF3* with $N \leqslant 10$ and FF1 with $N \leqslant 7$. By doing computations for Iter*, we extend this to show that **FF3* does not offer a 128-bit security for** $N \leqslant 17$, and **FF1 does not offer a 128-bit security for** $N \leqslant 11$.

Genuinely , we can compute in Table 5 the minimum $r_{opt} \geqslant 4$ of the number of rounds for which $\min(T^{MITM^*}, T^{Iter^*}) \geqslant 2^s$ depending on s and N. Again, we computed without using our approximations. For $s = 128$ and $s = 256$, we fetch the following table.[6]

Even by adding a safety margin, this shows that we do not need many rounds to safely encrypt a byte (that is, $N = 2^4$) 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 $r \leqslant 7$ due to the yo-yo attack [7]. For other FN, we recommend never to take $r \leqslant 5$.

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

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

s = 128				s = 256			
N	$\mathsf{r}_{\mathrm{opt}}$	$\mathsf{T}^{\mathrm{MITM}^*}$	$\mathsf{T}^{\mathrm{Iter}^*}$	N	$\mathsf{r}_{\mathrm{opt}}$	$\mathsf{T}^{\mathrm{MITM}^*}$	$\mathsf{T}^{\mathrm{Iter}^*}$
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	$2^{4080.0}$	$\mathbf{2^{370.4}}$

Table 5: Minimal number $\mathsf{r}_{\mathrm{opt}}$ 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 aproximations.

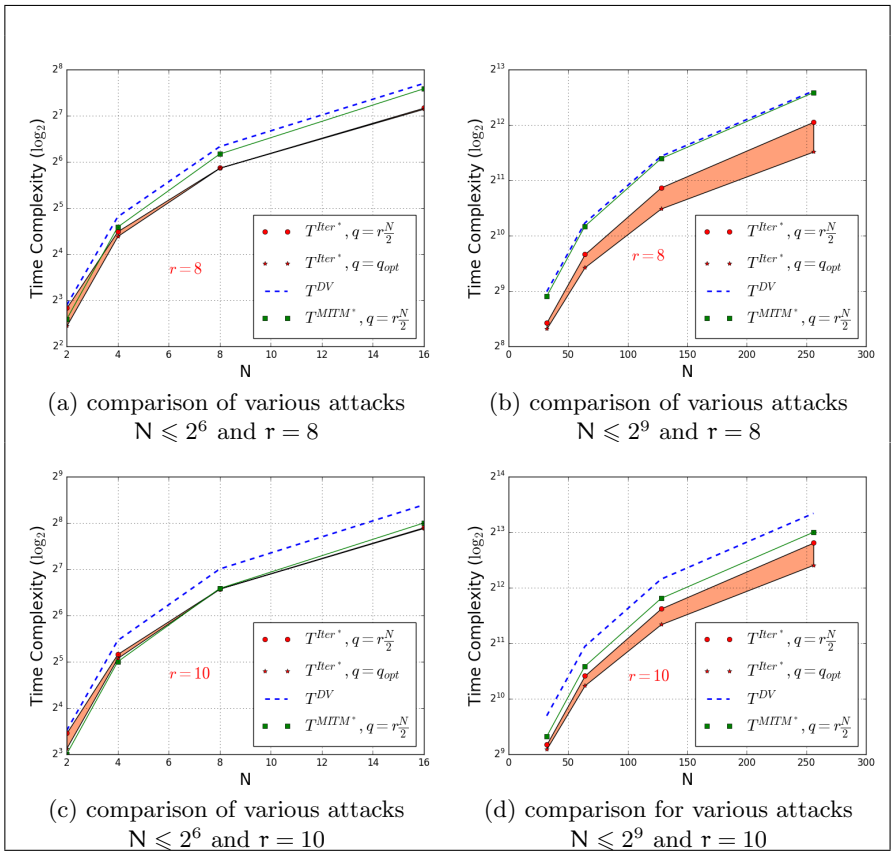
5 Conclusion

Standard Feistel Networks and its variations have created an active research area since its invention and have been used in construction of many cryptographic systems to a wide extent. The security of FN has been studied for so long and many interesting results have been proposed for cryptanalysis purpose. In this work, we consider 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 have 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 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.

References

1. *Data Encryption Standard, National Bureau of Standards, NBS FIPS PUB 46, January 1977.* National Bureau of Standards, U.S. Department of Commerce.
2. *Recommendation for Block Cipher Modes of Operation: Methods for Format Preserving Encryption, NIST Special Publication (SP) 800-38G, March 29, 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.



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

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

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

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

Fig. 4: Time complexity of attacks for different algorithms and various parameters.

4. Mihir Bellare, Viet Tung Hoang, and Stefano Tessaro. Message-recovery attacks on Feistel-based Format-Preserving Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 444–455, New York, NY, USA, 2016. ACM.
5. Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-Preserving Encryption. In Michael J. Jacobson, Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography: 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867, pages 295–312. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
6. Mihir Bellare, Phillip Rogaway, and Terence Spies. The FFX Mode of Operation for Format-Preserving Encryption. draft 1.1. submission to NIST, Feb. 2010. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec.pdf`.

7. Alex Biryukov, Gaëtan Leurent, and Léo Perrin. Cryptanalysis of Feistel Networks with Secret Round Functions. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015: 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566, pages 102–121. Springer International Publishing, 2016.
8. Alex Biryukov and Léo Perrin. On Reverse-Engineering S-boxes with Hidden Design Criteria or Structure. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215, pages 116–140. Springer International Publishing, 2015.
9. Eric Brier, Thomas Peyrin, and Jacques Stern. BPS: A Format-Preserving Encryption Proposal. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf`.
10. W. Diffie and M. E. Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10(6):74–84, June 1977.
11. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems. In *Advances in Cryptology – CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417, pages 719–740, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
12. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. New attacks on Feistel Structures with Improved Memory Complexities. In *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215, pages 433–454, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
13. F. Betül Durak and Serge Vaudenay. Breaking the FF3 Format-Preserving Encryption. Proceedings of ESC 2017: `https://www.cryptolux.org/mediawiki-esc2017/images/8/83/Proceedings_esc2017.pdf`.
14. F. Betül Durak and Serge Vaudenay. Breaking the FF3 Format-Preserving Encryption Standard Over Small Domains. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II*, volume 10402, pages 679–707. Springer International Publishing, Cham, 2017.
15. Viet Tung Hoang and Phillip Rogaway. On Generalized Feistel networks. In *Advances in Cryptology – CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223, pages 613–630, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
16. Takanori Isobe and Kyoji Shibutani. All Subkeys Recovery Attack on Block Ciphers: Extending Meet-in-the-Middle Approach. In *Selected Areas in Cryptography: 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707, pages 202–221, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
17. Takanori Isobe and Kyoji Shibutani. Generic Key Recovery Attack on Feistel Scheme. In *Advances in Cryptology - ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269, pages 464–485, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
18. Jiqiang Lu, Jongsung Kim, Nathan Keller, and Orr Dunkelman. Improving the Efficiency of Impossible Differential Cryptanalysis of Reduced Camellia and MISTY1.

In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, pages 370–386, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
19. Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, April 1988.
20. Valérie Nachef, Emmanuel Volte, and Jacques Patarin. Differential Attacks on Generalized Feistel Schemes. In *Cryptology and Network Security: 12th International Conference, CANS 2013, Paraty, Brazil, November 20-22. 2013. Proceedings*, volume 8257, pages 1–19. Springer International Publishing, 2013.
21. Jacques Patarin. New Results on Pseudorandom Permutation Generators Based on the DES Scheme. In *Advances in Cryptology — CRYPTO '91: Proceedings*, volume 576, pages 301–312, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.
22. Jacques Patarin. Generic attacks on Feistel schemes. `http://eprint.iacr.org/2008/036`, 2008.
23. Jacques Patarin. Security of Balanced and Unbalanced Feistel Schemes with Non-Linear Equalities. `http://eprint.iacr.org/2010/293`, 2010.
24. Jacques Patarin, Valérie Nachef, and Côme Berbain. Generic Attacks on Unbalanced Feistel Schemes with Contracting Functions. In *Advances in Cryptology – ASIACRYPT 2006: 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006. Proceedings*, volume 4284, pages 396–411, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
25. Serge Vaudenay. The security of DSA and ECDSA. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003: 6th International Workshop on Practice and Theory in Public Key Cryptography Miami, FL, USA, January 6–8, 2003 Proceedings*, volume 2567, pages 309–323, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

A Message Recovery Attack [4]

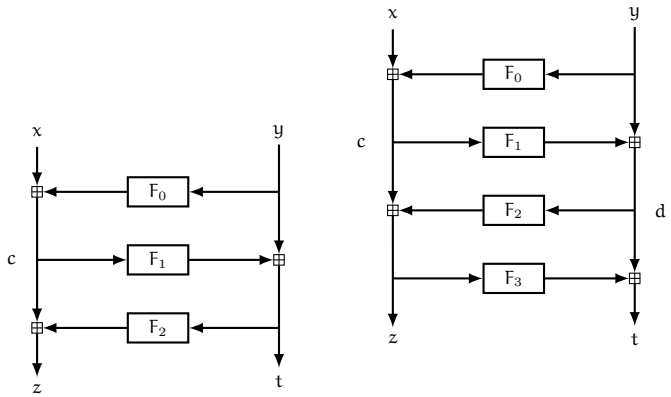
In their recent work [4] on FFX FPE schemes, Bellare et. al. consider an FN scheme with round functions built as tweakable block ciphers. They gave a message recovery attack with data complexity larger than the domain size by using small number of messages per tweak. Basically, their attack is a differential attack that exploits the bias introduced on the left/right part of the input in Feistel networks. The idea of the bias they exploit was discovered by Patarin [21]. Namely, consider two messages $M^{(0)} = (M_L^{(0)}, M_R^{(0)})$ and $M'^{(0)} = (M_L'^{(0)}, M_R^{(0)})$ as an input with same $M_R^{(0)}$ to the FN with modular addition under the same tweak. Let $M_L^{(i)}$ (resp. $M_L'^{(i)}$) be the output of left part of FN in i^{th} round. Then, we can show that $M_L^{(i)} - M_L'^{(i)}$ is most likely to be $M_L^{(0)} - M_L'^{(0)}$.

In [4], more specifically, the authors consider two messages M and M' encrypted under FN where they share the same right that is known to the adversary. In the attack, the adversary obtains the encryption of M and M' under q tweaks, the entire message M' and the shared half of the messages. At the end, the adversary outputs the unknown half of the message M with probability close to 1 by using a bias. The bias simply works as the following: Consider q pairs (M, C_i) and (M', C_i') for each tweak, if we apply modular subtraction to the left part of the C_i and C_i' and modular addition to known left part under each

tweak. The attack observes a value more likely than the others, and outputs this value as the unknown half of the message. The data complexity of the attack for an r round FN is $q = 24(\log(N) + 4)N^{(r-3)}$ where N is the input size of the right branch in FN. The time complexity is linear in q.

B Generic Round-Function-Recovery Attack [13, 14]

In their recent work, Durak and Vaudenay gave a generic round-function-recovery attack to the FN for 3, 4 and more rounds. The attack for 3-round and 4-round FN are known-plaintext attacks while the attack for 5-round FN is a chosen-plaintext attack. For the description of attacks, consider the Fig. 5.



(a) 3-round Feistel network (b) 4-round Feistel network

Fig. 5: 3-round and 4-round Feistel Schemes

Let's start with the attack for 2-round FN, which is relatively easier. For 2-round FN, consider input $x\|y$ with output $z\|t$. The first round function is easy to compute since we know x and z, $F_0(y) = x - z$. We can deduce the second round function by simply computing the $F_1(z) = y - t$. With N known plaintexts/ciphertext pair, we recover the entire table of F_0 and F_1.

The 3-round attack by Durak and Vaudenay [13, 14] is a known-plaintext attack. The idea is to fix arbitrarily the output of the round function F_0 for some inputs (refer to Fig. 5 (a)). This gives one point of the table of F_0. If we do so, we can apply 2-round attack. Then, we can make the pool of known points for F_0, F_1, F_2 increase in a yo-yo game. The time and data complexity is N to recover a good fraction of the tables and $N \ln N$ to recover them totally, with good probability.

In 4-round known-plaintext attack, the authors are interested in a special structure in a set of pairs of plaintext-ciphertext pairs with specific properties

(refer to Fig. 5 (b)). Given a pair of plaintext-ciphertext pairs, these properties are the equality on the intermediate values of a Feistel Network which are unknown as well as the known outputs. In order to distinguish the pairs of plaintext-ciphertext with these properties, the attack constructs a weighted directed graph. In this graph, a vertex is a pair of distinct pairs $(xyzt, x'y'z't')$ with $t - y = t' - y'$ and $z = z'$ values. There are vertices which are called "good" when the vertices have $c = c'$. Using good vertices, the attack deduces the equation $x + F_0(y) = x' + F_0(y')$. One problem in this approach is to tell good and non-good vertices apart. For this, the attack uses a cycle finding algorithm to identify these good vertices in the graph. The data complexity of the attack is $N^{\frac{3}{2}}$ and time complexity is $O\left(N^3\right)$.

5-round recovery is straightforward expansion of 4-round attack as a chosen-plaintext attack. In the first round of 5-round FN, for the chosen plaintexts, we start with guessing the outputs of the first round function F_0. This gives us a partial table of F_0. If we run the 4-round attack given above, for the rest of the rounds in 5-round FN, we can look for the consistency of this partial table with the 4-round attack results. The time complexity is, therefore, $T^{DV} = O\left(N^{\sqrt{N}+3}\right)$ for $r = 5$. We extend this to more rounds by guessing the round functions on the $r - 5$ rounds and the attack runs in time

$$T^{DV} = O\left(N^{(r-5)N+\sqrt{N}+3}\right)$$

Durak and Vaudenay [13, 14] also give a non-generic attack on FF3 (which has 8 rounds). Due to the bad tweak management in FF3, the authors could reduce the attack on 8-round FF3 construction to a 4-round FN attack and apply their generic attack. They also fix FF3 into the FF3* scheme.

C Biased Outputs Due to the Modular Addition

Consider an FN with round functions producing an ℓ-bit output number which is reduced modulo N. For an output $x \in \{0, 1, 2, \ldots, 2^\ell - 1\}$ of a round function, we have $y = x \mod N$. Let $2^\ell = uN + v$ with $v < N$. Now, the probability that $y = t$ for $t \in \{0, \ldots, v - 1\}$ is $\Pr(y = t) = \frac{u+1}{N}$, whereas the probability that $y = t$ is a bit smaller for $t \in \{v, \ldots, N - 1\}$, i.e. $\Pr(y = t) = \frac{u}{N}$. The bias comes from the fact that the round function typically outputs a value between 0 to $2^l - 1$ ($l = 128$) and then it reduced to modulo N. Therefore, the output of a round function is biased to some values. We can exploit this to design an attack to FN as in the Bleichenbacher attack against DSA [7].

Assume that we first let the round function output a uniformly distributed ℓ-bit value, and then the value is reduced modulo N with $N < 2^\ell$. We consider x and y as random variables. With the characteristic function of random variable y distributed among the circle in the N^{th} roots of unity, we define the bias of y as $\mathtt{bias}(y) = \mathbb{E}[e^{\frac{2i\pi y}{N}}] = \mathbb{E}[e^{\frac{2i\pi x}{N}}]$.

[7] It is unpublished but cited by Vaudenay [25]

$$\mathtt{bias}(y) = \frac{1}{2^\ell}\sum_{x=0}^{2^\ell-1} e^{\frac{2i\pi x}{N}} = \frac{1}{2^\ell} e^{\frac{2i\pi(2^\ell-1)}{N}} \frac{\sin\left(\frac{\pi 2^\ell}{N}\right)}{\sin\left(\frac{\pi}{N}\right)}$$

which comes from the sum of the geometric sequence that is $\sum_{j=0}^{2^\ell-1} t^j = \frac{1-t^{2^\ell}}{1-t}$.

When we have r-round FN, we have $\lfloor \frac{r}{2} \rfloor$ random y in total for a right branch. Assume that all the random variables are independent. For a branch (left or right), we have a sum of biased random variables which are defined as $\mathtt{bias}(\sum y) = \mathtt{bias}(y)^{\lfloor \frac{r}{2} \rfloor}$.

In the worst case, we have $2^\ell \mod N = \frac{N}{2}$ (to have $\sin(\frac{\pi 2^\ell}{N}) = 1$ in the numerator of $\mathtt{bias}(y)$), so $|\mathtt{bias}(\sum y)| = \left|\frac{1}{2^\ell \sin\left(\frac{\pi}{N}\right)}\right|^{\lfloor \frac{r}{2} \rfloor}$. Then, we have $|\mathtt{bias}(\sum y)| \sim \left|\frac{\pi 2^\ell}{N}\right|^{-\lfloor \frac{r}{2} \rfloor}$ for large N.

With the bias that we have computed, we can make (in the worst case) a distinguisher to decide if a variable has a bias. The complexity to decide if a variable has bias τ or 0 with good advantage is $O(q)$ with $q = \frac{1}{\tau^2}$. Therefore, the time complexity for the distinguisher is $O\left(\left(\frac{\pi 2^\ell}{N}\right)^{2\lfloor \frac{r}{2} \rfloor}\right)$. (Note that this can only make sense if $q \leqslant N^2$.)

In FF1 and FF3*, we use $N < 2^{\ell-32}$ and $r \geqslant 8$, so q is too large.

D Generic Round-Function-Recovery Attack with Guess and Determine Method [7]

In [7], chosen-plaintext and ciphertext attacks are given for 4 and 5-round FN with modular addition. Their attack is based on a distinguisher for a 3-round FN introduced by Luby and Rackoff in [19]. For this distinguisher, refer to the Fig. 5 (a). Let the adversary have access to both encryption and decryption oracle. The adversary selects a δ, it queries the encryption oracle with arbitrary $(x\|y)$ and $(x+\delta\|y)$, and obtains $(z\|t)$ and $(z'\|t')$ respectively. Then, the adversary queries $(z+\delta\|t)$ for decryption and obtains $(x''\|y'')$. The distinguisher checks if $t-y'' = t'-y$ to distinguish 3-round Feistel Network from a random permutation.

In the 4-round attack given by Biryukov et. al. (refer to the Fig. 5 (b)), consider a type of plaintext/ciphertext of the form $(xyzt)$, $(x''y''(z+\delta)t'')$, and $((x+\delta)yz't')$ with corresponding d, d, d' values in plaintexts/ciphertexts respectively. More precisely, the attacker starts with one arbitrarily fixed value of F_3 and one guessed value for F_3 (meaning that it iterates N time what follows). He sets z and $z+\delta$ such that their image by F_3 is known. For each d, it sets t and t''. With the queries to the decryption oracle for $(z\|t)$ and $(z+\delta\|t'')$, the attacker obtains $(x\|y)$ and $(x''\|y'')$. With a query to the encryption oracle on $(x+\delta\|y)$, it gets $(z'\|t')$. With all the obtained values, the attacker can use the 3-round property of the above defined distinguisher and can find an $F_3(z') = t'-d+y''-y$ output of F_3 for a new value. The adversary iterates

on d to get new outputs of F_3 until it finds no conflict. On average, the conflicts occur after $\mathsf{O}(\sqrt{\mathsf{N}})$ trials for d. The time complexity of this attack is $\mathsf{O}\left(\mathsf{N}^{\frac{3}{2}}\right)$ with $\mathsf{N}^{\frac{3}{2}}$ data complexity.

The 5-round attack in Biryukov et. al. is extended form of 3-round distinguisher and 4-round attack with more guesses. Its time complexity is $\mathsf{O}\left(\mathsf{N}^{\mathsf{N}^{\frac{3}{4}}}\right)$ with N^2 data complexity.