Attacks Only Get Better:
How to Break FF3 on Large Domains

Viet Tung Hoang[1], David Miller[1], and Ni Trieu[2]

[1] Dept. of Computer Science, Florida State University, USA.
[2] Dept. of Computer Science, Oregon State University, USA.

Abstract. We improve the attack of Durak and Vaudenay (CRYPTO'17) on NIST Format-Preserving Encryption standard FF3, reducing the running time from $O(N^5)$ to $O(N^{17/6})$ for domain $\mathbb{Z}_N \times \mathbb{Z}_N$. Concretely, DV's attack needs about 2^{50} operations to recover encrypted 6-digit PINs, whereas ours only spends about 2^{30} operations. In realizing this goal, we provide a pedagogical example of how to use distinguishing attacks to speed up slide attacks. In addition, we improve the running time of DV's known-plaintext attack on 4-round Feistel of domain $\mathbb{Z}_N \times \mathbb{Z}_N$ from $O(N^3)$ time to just $O(N^{5/3})$ time. We also generalize our attacks to a general domain $\mathbb{Z}_M \times \mathbb{Z}_N$, allowing one to recover encrypted SSNs using about 2^{50} operations. Finally, we provide some proof-of-concept implementations to empirically validate our results.

Keywords: Format-preserving encryption, attacks

1 Introduction

Format-Preserving Encryption (FPE) [6, 12] is a form of deterministic symmetric encryption mechanism that preserves the *format* of plaintexts. For example, encrypting a 16-digit credit-card number under FPE would result in a 16-digit number, and encrypting a valid SSN would produce a ciphertext of nine decimal digits. FPE is widely used in practice by several companies, such as HPE Voltage, Verifone, Protegrity, Ingenico, to encrypt credit-card numbers and protect legacy databases. Recent research [4, 15, 20] however show that existing FPE standards FF1 and FF3 (NIST SP 800-38G, ANSI ASC X9.124) are somewhat vulnerable in small domains. The most damaging attack, due to Durak and Vaudenay (DV) [15], can recover the entire codebook of FF3 using $O(N^5)$ expected time, for domain $\mathbb{Z}_N \times \mathbb{Z}_N$.

Still, the attacks above are feasible only if the domain size is small; their cost becomes prohibitive for moderate and large domains. For example, for domain $\mathbb{Z}_{10}^6$ (namely encrypting 6-digit PINs), DV's attack would use about 2^{50} operations. In this paper, we improve DV's attack to break FF3 on large domains. Our attack can reduce the cost of breaking FF3 on domain $\mathbb{Z}_N \times \mathbb{Z}_N$ to $O(N^{17/6})$

N	**Our queries**	**DV's queries**	**Our rate**	**DV's rate**	**Our time**	**DV's time**
128	16,384	17,388	39%	56.85%	2^{20}	2^{35}
256	52,012	55,176	50%	55.9%	2^{23}	2^{40}
512	165,140	175,164	33%	77.4%	2^{26}	2^{45}

Table 1. Our attack versus DV's. The first column indicates the values of N in the domain $\mathbb{Z}_N \times \mathbb{Z}_N$. The second column and third column show the number of queries in our attack and that of DV respectively; in both attacks, the queries are made over two tweaks. The fourth and fifth columns show our recovery rate and that of DV respectively, and the fifth and sixth columns show our time and DV's time respectively.

expected time, meaning that it will need about 2^{30} operations to break FF3 of the domain $\mathbb{Z}_{10}^6$ above. Achieving this efficiency involves an elegant paradigm of combining distinguishing attacks with slide attacks [10, 11], and improved cryptanalyses of 4-round Feistel. We give rigorous analyses to justify the advantage of our attack; proofs omitted due to lack of space appear in the full version of this paper. We also provide proof-of-concept implementations in Section 5 that empirically confirm our analyses.

We note that our attack essentially performs the same queries as DV's, and thus the two attacks have the same scenario and asymptotic data/space complexity $\Theta(N^{11/6})$ for domain $\mathbb{Z}_N \times \mathbb{Z}_N$. However, DV use more aggressive choices of the parameters, and thus our attack is concretely better in both data and space complexity, albeit at the cost of lower recovery rate. A concrete comparison of the two attacks are given in Table 1. Still, one can improve the recovery rate by relaunching our attack with different tweaks. For example, for domain $\mathbb{Z}_{128} \times \mathbb{Z}_{128}$, if one relaunches our attack another time, the recovery rate would become $1 - (1 - 0.39)^2 \approx 62\%$. See Section 3.3 for further details.

Existing cryptanalysis. Let us begin by reviewing prior attacks on the standards FF1 and FF3. Bellare, Hoang, and Tessaro (BHT) [4] give the first attack on these schemes, showing that one can fully recover a target message using $O(N^6 \log(N))$ pairs of plaintext/ciphertext, on domain $\mathbb{Z}_N \times \mathbb{Z}_N$. Their attack however requires that a designated, partially known message must have the same right half as the target, but it is unclear how one could mount such a correlation in practice. Hoang, Tessaro, and Trieu (HTT) [20] subsequently improve BHT's attack, requiring no correlation between the known messages and the target. Even better, they can reuse the known plaintext/ciphertext pairs to attack multiple targets, thus reduce the *amortized* cost to $O(N^5 \log^2(N))$ pairs per target. Both attacks above apply to a *generic* Feistel-based FPE, meaning that they break both FF1 and FF3, and the only way to thwart them is to increase the round count of the underlying Feistel networks.

In a different direction, Durak and Vaudenay (DV) [15] give a dedicated attack on FF3, exploiting a bug in its design of round functions. They show that on domain $\mathbb{Z}_N \times \mathbb{Z}_N$, one can recover the entire codebook of FF3 using $O(N^{11/6})$ pairs of chosen plaintext/ciphertext, within $O(N^5)$ expected running time. We

stress that DV's attack does not apply to FF1, and it can be fixed without hurting performance by restricting the tweak space, as DV already suggested.

In response to DV's attack, NIST has temporarily suspended the use of FF3, whereas a draft update of the ANSI ASC X9.124 standard additionally recommends using double encryption on small domains to cope with the other attacks.

A BIRD'S-EYE VIEW ON DV'S ATTACK. We now briefly sketch a blueprint of DV's attack. Recall that in the balanced setting, the encryption scheme of FF3 is simply a tweakable blockcipher $\mathsf{F.E} : \mathsf{F.Keys} \times \mathsf{F.Twk} \times (\mathbb{Z}_N \times \mathbb{Z}_N) \to (\mathbb{Z}_N \times \mathbb{Z}_N)$ that is based on an 8-round balanced Feistel network. Due to a bug in the round functions of FF3, one can find two tweaks T and T^* such that $\mathsf{F.E}(K, T, \cdot)$ is the cascade $g(f(\cdot))$ of two 4-round Feistel networks f and g, whereas $\mathsf{F.E}(K, T^*, \cdot) = f(g(\cdot))$. Then, by mounting a slide attack using $O(N^{11/6})$ encryption queries, we obtain $O(N^2)$ instances, each of $O(N^{5/3})$ pairs of plaintext/ciphertext for f and also $O(N^{5/3})$ pairs for g. However, often just one of those instances provides the correct ciphertexts under f or g; in the remaining instances, the ciphertexts are random strings, independent of the plaintexts. DV resolve this by developing a codebook-recovery attack on 4-round Feistel networks using $O(N^3)$ expected running time. They then try this attack on every instance, using totally $O(N^5)$ expected time.

CONTRIBUTION: ELIMINATING FALSE INSTANCES. To improve the running time of DV's attack, we observe that it is an overkill to use an expensive codebook-recovery attack on false instances. A better solution is to find a cheap test to tell whether an instance is true or false, and then use the codebook-recovery attack on the true instances. A natural choice for such a test is a distinguishing attack on 4-round Feistel. However, the requirement here is a lot more stringent. To eliminate most of the random instances, our distinguishing attack should output 1 with probability about $1/N$ if it is given a false instance. To ensure that we will *not* incorrectly eliminate all true instances, the distinguishing attack should output 1 with high probability, say $1/2$, if it is given a true instance.

Our starting point is Patarin's distinguishing attack on 4-round Feistel [25],[3] which uses $O(\sqrt{N})$ pairs of plaintext/ciphertext. However, using this attack for our purpose runs into two obstacles. First, Patarin's asymptotic analysis is insufficient to pinpoint the hidden constant in the Big-Oh. Next, Patarin's attack fails to meet the requirement above, as given a false instance, the attack outputs 1 with constant probability.

Given the issues above, we instead design a new distinguishing attack, Left-Half Differential (LHD), such that (1) in the ideal world, it returns 1 with probability at most $\frac{1}{\sqrt{N}}$, and (2) in the real world, it returns 1 with probability at least $1 - \frac{1}{8\sqrt{N}} - \frac{10}{N} - \frac{1}{N^{3/4}}$. The LHD attack uses $O(N^{5/6})$ pairs of plaintext/ciphertext, and runs in $O(N^{5/6})$ time. Our analyses are generalized enough to include Patarin's attack as a special case. As a result, we can show that for $N \geq 2^{16}$, if one uses

[3] While Patarin's attack is given for classic Feistel (meaning that $N = 2^n$, and the underlying operator is xor), generalizing it to cover FF3 setting is straightforward.

Type	Power	Source	Data	Time
Known-plaintext	Distinguishing	[24, 1] Here	$O(\sqrt{N})$	$O(\sqrt{N})$
Known-plaintext	Full recovery	[15]	$O(N^{5/3})$	$O(N^3)$
Known-plaintext	Full recovery	Here	$O(N^{5/3})$	$O(N^{5/3})$
Chosen plaintext & ciphertext	Full recovery	[9]	$O(N^{3/2})$	$O(N^{3/2})$

Table 2. A list of attacks on generic 4-round Feistel of domain $\mathbb{Z}_N \times \mathbb{Z}_N$. While the distinguishing attack was discovered by Patarin [24] and independently by Aiello and Venkatesan [1], the analyses in those papers are asymptotic. Our paper gives the first concrete treatment for this attack.

$\lceil 7 \cdot \sqrt{N} \rceil$ pairs of plaintext/ciphertext then Patarin's attack achieves advantage at least $1/2$.

In our test, we run LHD *twice*, first on the plaintext/ciphertext pairs of f, and then on those of g. Thus given a false instance, the chance that we fail to eliminate it is at most $\frac{1}{N}$, whereas given a true instance, the chance that we accept it is at least $\left(1 - \frac{1}{8\sqrt{N}} - \frac{10}{N} - \frac{1}{N^{3/4}}\right)^2$. Even better, our experiments indicate that in practice our test is nearly perfect, meaning that empirically, we never miss a true instance, and eliminate almost all false instances.

We note that while the idea of using distinguishing attacks to eliminate false instances in slide attacks was already known in the literature [2], to the best of our knowledge, nobody has ever explored this direction. Our analyses of FF3 thus provide a pedagogical example of this paradigm.

Contribution: A better attack on 4-round Feistel. Thanks to the LHD tests above, we are now left with $O(N)$ false instances and a few true instances. If one uses DV's codebook recovery attack on 4-round Feistel, one would end up with $O(N^4)$ expected time, which is still very expensive. The core part of DV's attack needs to find all directed 3-cycles of zero weight in a (random) directed graph $\mathcal{G} = (V, E)$. DV's approach is to enumerate all directed 3-cycles via some sparse matrix multiplications, and then pick those of zero weight, spending $O(|V| \cdot |E|)$ time. We instead give an elementary algorithm that uses $O(|V| + |E|)$ expected time. In addition, DV's attack relies on a conjecture of Feistel networks. They however can only empirically verify this conjecture for $N \in \{2, 2^2, \ldots, 2^9\}$. In this work, we resolve this conjecture, solving an open problem posed by DV.

Our algorithm above leads to the best known-plaintext attack to 4-round Feistel in the literature, using $O(N^{5/3})$ data and time complexity. A prior work by Biryukov, Leurent, and Perrin [9] is slightly better, recovering the codebook within $O(N^{3/2})$ data and time, but this attack requires chosen plaintexts and ciphertexts. A comparison of the attacks on 4-round Feistel is listed in Table 2.

Other contributions. We also generalize our FF3 attack to unbalanced settings, for a general domain $\mathbb{Z}_M \times \mathbb{Z}_N$, with $M \geq N \geq 64$, so that we can recover,

say encrypted SSNs. The asymmetry $M \geq N$ however requires some care in the extension of the attack on 4-round Feistel. In particular, due to the symmetry of 4-round Feistel, given plaintext/ciphertext pairs $(M_1, C_1), \ldots, (M_p, C_p)$, one can view $M_1, \ldots, M_p$ as the "ciphertexts" of $C_1, \ldots, C_p$ under an inverse 4-round Feistel, leading to a dual attack. The two attacks yield no difference in the balanced setting $M = N$, but if $M \gg N$, we find that the dual attack provides a superior recovery rate. We also introduce some tricks that substantially improve both the data complexity and the recovery rate.

On the other hand, there are often some gaps between the choices of the parameters according to DV's analyses, and what their experiments suggest. Even worse, the performance of their attacks is highly sensitive: in some experiments, if they triple the number of plaintext/ciphertext pairs, ironically, the recovery rate drops from 77% to 0%. DV thus have to calibrate concrete choices of the parameters via extensive experiments. In contrast, we choose to err on the conservative side in our analyses, and our estimates are consistent with the experiments. We also add some fail-safe to avoid the performance degradation when the number of plaintext/ciphertext pairs increases.

Limitation of our attack on FF3. Our attack exploits the same bug of FF3 as DV's attack, and thus it can be thwarted without hurting performance by restricting the tweak space, as DV suggested. In addition, both of our attack and DV's requires that the adversary can *adaptively* make chosen plaintexts on $\Theta(N^2)$ queries for domain $\mathbb{Z}_N \times \mathbb{Z}_N$, but it is unclear how to mount this kind of attack, especially with that many queries, in practice.

Additional related work. There have been two separate lines of building FPE schemes. On the theoretical side, we have provably secure constructions that are based on card shuffling, such as Swap-or-Not [18], Mix-and-Cut [26], or Sometimes-Recurse [22] that are too slow for performance-hungry applications. On the practical side, in addition to FF1/FF3, there are other industry proposals, such as FNR from Cisco [14], or DTP from Protegrity [21], that have no theoretical justification. Hoang, Tessaro, and Trieu [20] however show that FNR is somewhat vulnerable in tiny domains, and DTP is completely broken even in large domains.

In a different direction, Bellare and Hoang [3] study the security of DFF, an FPE scheme currently proposed to NIST for standardization [28], and show that for appropriately large domains, DFF provides a way to localize and limit the damage from key exposure. However, as DFF is based on a 10-round Feistel network, it is still subject to prior attacks on generic Feistel-based FPE [5, 20] on tiny domains.

Very recently, Durak and Vaudenay [16] give some theoretical codebook-recovery attacks on generic balanced r-round Feistel, for $r \geq 5$. They conclude that on domain $\mathbb{Z}_N \times \mathbb{Z}_N$, FF1 cannot provide 128-bit security for $N \leq 11$, and FF3 for $N \leq 17$.

2 Preliminaries

NOTATION. If y is a string then let $|y|$ denote its length and let $y[i]$ denote its i-th bit for $1 \le i \le |y|$. We write $y[i:j]$ to denote the substring of y, from the ith bit to the j-th bit, inclusive. If X is a finite set, we let $x \xleftarrow{\$} X$ denote picking an element of X uniformly at random and assigning it to x. We use the code based game playing framework of [7]. In particular, by $\Pr[\mathrm{G}]$ we denote the probability that the execution of game G returns true.

FPE. An FPE scheme F is a pair of deterministic algorithms $(\mathsf{F.E}, \mathsf{F.D})$, where $\mathsf{F.E} : \mathsf{F.Keys} \times \mathsf{F.Twk} \times \mathsf{F.Dom} \to \mathsf{F.Dom}$ is the encryption algorithm, $\mathsf{F.D} : \mathsf{F.Keys} \times \mathsf{F.Twk} \times \mathsf{F.Dom} \to \mathsf{F.Dom}$ the decryption algorithm, $\mathsf{F.Keys}$ the key space, $\mathsf{F.Twk}$ the tweak space, and $\mathsf{F.Dom}$ the domain. For every key $K \in \mathsf{F.Keys}$ and tweak $T \in \mathsf{T}$, the map $\mathsf{F.E}(K, T, \cdot)$ is a permutation over $\mathsf{F.Dom}$, and $\mathsf{F.D}(K, T, \cdot)$ reverses $\mathsf{F.E}(K, T, \cdot)$.

FEISTEL-BASED FPES. Most existing FPE schemes, including FF3, are based on Feistel networks. Following BHT [5], we specify Feistel-based FPE in a general, parameterized way. This allows us to refer to both schemes of ideal round functions for the analysis, and schemes of some concrete round functions for realizing the standards.

We associate to parameters $r, M, N, \boxplus, \mathrm{PL}$ an FPE scheme $\mathsf{F} = \mathbf{Feistel}[r, M, N, \boxplus, \mathrm{PL}]$. Here $r \ge 2$ is an integer, the number of rounds, and $\boxplus$ is an operation for which $(\mathbb{Z}_M, \boxplus)$ and $(\mathbb{Z}_N, \boxplus)$ are Abelian groups. We let $\boxminus$ denote the inverse operator of $\boxplus$, meaning that $(X \boxplus Y) \boxminus Y = X$ for every X and Y. Integers $M, N \ge 1$ define the domain of F as $\mathsf{F.Dom} = \mathbb{Z}_M \times \mathbb{Z}_N$. The parameter $\mathrm{PL} = (\mathcal{T}, \mathcal{K}, F_1, \ldots, F_r)$ specifies the set $\mathcal{T}$ of tweaks and a set $\mathcal{K}$ of keys, meaning $\mathsf{F.Twk} = \mathcal{T}$ and $\mathsf{F.Keys} = \mathcal{K}$, and the round functions $F_1, \ldots, F_r$ such that $F_i : \mathcal{K} \times \mathcal{T} \times \mathbb{Z}_N \to \mathbb{Z}_M$ if i is odd, and $F_i : \mathcal{K} \times \mathcal{T} \times \mathbb{Z}_M \to \mathbb{Z}_N$ if i is even. The code of $\mathsf{F.E}$ and $\mathsf{F.D}$ is shown in Fig. 1.

Classic Feistel schemes correspond to the boolean case, where $M = 2^m$ and $N = 2^n$ are powers of two, and $\boxplus$ is the bitwise xor operator $\oplus$. The scheme is balanced if $M = N$ and unbalanced otherwise. For $X = (L, R) \in \mathbb{Z}_M \times \mathbb{Z}_N$, we call L and R the *left segment* and *right segment* of X, respectively. We write $\mathsf{LH}(X)$ and $\mathsf{RH}(X)$ to refer to the left and right segments of X respectively. For simplicity, we assume that 0 is the zero element of the groups $(\mathbb{Z}_M, \boxplus)$ and $(\mathbb{Z}_N, \boxplus)$.

FEISTEL-BASED BLOCKCIPHERS. If the tweak space $\mathcal{T}$ is a singleton set then FPE degenerates into a blockcipher (of a general domain). For such a blockcipher F, we write $\mathsf{F.E}(K, M)$ and $\mathsf{F.D}(K, C)$ instead of $\mathsf{F.E}(K, T, M)$ and $\mathsf{F.D}(K, T, C)$ respectively.

In our analysis of Feistel-based blockciphers, the round functions are modeled as truly random. We write $\mathbf{Feistel}[r, M, N, \boxplus]$ to denote $\mathbf{Feistel}[r, M, N, \boxplus, \mathrm{PL}]$, for the ideal choice of $\mathrm{PL} = (\mathcal{T}, \mathcal{K}, F_1, \ldots, F_r)$ in which (i) $\mathcal{T} = \{\varepsilon\}$ where ε is the empty string, and (ii) $\mathcal{K}$ is the set $\mathbf{RF}(r, M, N)$ of all tuples of functions $(G_1, \ldots, G_r)$ such that $G_i : \mathbb{Z}_N \to \mathbb{Z}_M$ if i is odd, and $G_i : \mathbb{Z}_M \to \mathbb{Z}_N$ if i is

$\underline{\mathsf{F.E}(K,T,X)}$
$(L,R) \leftarrow X$
For $i = 1$ to r do
  If $(i \bmod 2 = 1)$ then $L \leftarrow L \boxplus F_i(K,T,R)$
  Else $R \leftarrow R \boxplus F_i(K,T,L)$
Return (L,R)

$\underline{\mathsf{F.D}(K,T,Y)}$
$(L,R) \leftarrow Y$
For $i = r$ to 1 do
  If $i \bmod 2 = 1$ then $L \leftarrow L \boxminus F_i(K,T,R)$
  Else $R \leftarrow R \boxminus F_i(K,T,L)$
Return (L,R)

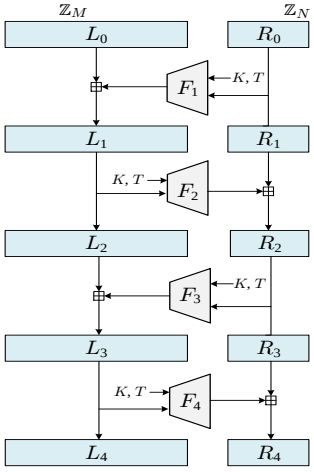

Fig. 1. Left: The code for the encryption and decryption algorithms of $\mathsf{F} = \mathbf{Feistel}[r, M, N, \boxplus, \mathrm{PL}]$ where $\mathrm{PL} = (\mathcal{T}, \mathcal{K}, F_1, \ldots, F_r)$. **Right:** An illustration of encryption with $r = 4$ rounds.

even, and (iii) for $1 \le i \le r$, the function $F_i(K, \cdot)$ is defined as $G_i(\cdot)$, where $(G_1, \ldots, G_r) \leftarrow K$.

3 Breaking FF3

In this section, we describe a chosen-plaintext codebook-recovery attack on FF3 that we call *Slide-then-Differential* (SD) attack.[4] This is based on Triangle-Finding (TF) attack, a known-plaintext codebook-recovery attack on 4-round Feistel that we will present in the next section. The running time of TF is $O(M^{5/3})$, and it actually recovers the round functions of the Feistel network, using

$$p = \max\Big\{ \lfloor 2^{1/3} M^{2/3} N \rfloor, \lceil M(\ln(M) + 5) \rceil \Big\} \tag{1}$$

known plaintext/ciphertext pairs. We note that TF is used in a modular way; one does not need to know its technical details to understand SD.

The FF3 scheme. FF3 is a Feistel-based FPE scheme $\mathsf{F} = \mathbf{Feistel}[8, M, N, \boxplus, \mathrm{PL}]$ of 8 rounds, where M and N are integers such that $M \ge N \ge 2$ and $MN \ge 100$.[5] The parameter PL specifies tweak space $\mathsf{F.Twk} = \{0,1\}^{2\tau}$, and

[4] While the notion of chosen-plaintext codebook-recovery attacks on blockciphers is folklore, one has to exercise some care in carrying this notion to FPE, because FPE domains can be tiny. In the full version we give a formal definition of chosen-plaintext codebook-recovery attacks on FPE.

[5] In NIST specification, the $\boxplus$ operation is the modular addition in $\mathbb{Z}_N$ and $\mathbb{Z}_M$, but here we will consider a generic group operator. Moreover, FF3 uses near-balanced

two keyed hash functions $H_1 : \mathsf{F.Keys} \times \{0,1\}^\tau \times \mathbb{Z}_N \to \mathbb{Z}_M$, and $H_2 : \mathsf{F.Keys} \times \{0,1\}^\tau \times Z_M \to \mathbb{Z}_N$. For each $i \leq 8$, if i is odd then the round function F_i is constructed via $F_i(K,T,X) = H_1(K, T[1:\tau] \oplus [i-1]_\tau, X)$, otherwise if i is even then $F_i(K,T,X) = H_2(K, T[\tau+1:2\tau] \oplus [i-1]_\tau, X)$, where $[j]_\tau$ is a τ-bit encoding of the integer j and $\oplus$ is the bitwise xor operator.

In analysis, the hash functions H_1 and H_2 are modeled as truly random. Formally, let $\mathcal{K}$ be the set $\mathbf{RF}(\tau, M, N)$ of all pairs of functions (G_1, G_2) such that $G_1 : \{0,1\}^{2\tau} \times \mathbb{Z}_N \to \mathbb{Z}_M$, and $G_2 : \{0,1\}^{2\tau} \times \mathbb{Z}_M \to \mathbb{Z}_N$. Then for each $j \leq 2$, define $H_j(K,\cdot,\cdot) = G_j(\cdot,\cdot)$, where $(G_1, G_2) \leftarrow K$, and we write $\mathbf{FF3}[M, N, \tau, \boxplus]$ to denote this ideal version of FF3.

In our attack to FF3, we will consider $M \geq N \geq 64$ and $MN \geq 2p$, where p is specified as in Equation (1). While there are indeed applications of smaller values of M and N, they are already susceptible to prior attacks [15, 4, 20] whose running time is practical in those tiny domains. In addition, to simplify our asymptotic analysis, we will assume that $N = \Omega(\sqrt{M})$, which applies to the setting of the FF3 scheme, since FF3 uses near-balanced Feistel. Thus $p \in O(M^{2/3}N)$.

3.1 DV's Blueprint for Breaking FF3

Let $\mathsf{F} = \mathbf{FF3}[M, N, \tau, \boxplus]$. Let K and T be a key and tweak for F, respectively. Recall that $\mathsf{F.E}(K, T, \cdot)$ is an 8-round Feistel network. View $\mathsf{F.E}(K, T, \cdot)$ as the cascade of two 4-round Feistel networks f and g, meaning $\mathsf{F.E}(K, T, X) = g(f(X))$ for every $X \in \mathbb{Z}_M \times \mathbb{Z}_N$. DV [15] observe that $\mathsf{F.E}(K, T', \cdot)$ is the cascade of g and f—note that the ordering of f and g is now reversed—where $T' = T \oplus ([4]_\tau \,\|\, [4]_\tau)$. See Fig. 2 for an illustration.

A sketch of DV's attack. From the observation above, one can launch a chosen-plaintext codebook-recovery attack on F as follows; this can also be viewed as a slide attack [10, 11]. Let p be as specified in Equation (1) and let $s = \left\lfloor \sqrt{MN/2p} \right\rfloor \geq 1$.[6] Sample s elements uniformly and independently from $\mathbb{Z}_M \times \mathbb{Z}_N$, and let S be the set of these elements. Repeat this process, and let S^* be the resulting set. Recall that the adversary is given an encryption oracle Enc in this attack. Now, for each $U_0 \in S$, we iterate $U_i \leftarrow \textsc{Enc}(T, U_{i-1})$, for $i = 1, \ldots, 2p$, forming a $\mathtt{U}$-*chain* $U_0 \to U_1 \to \cdots \to U_{2p}$. For each $V_0 \in S^*$, let $V_i \leftarrow \textsc{Enc}(T', V_{i-1})$ for $i = 1, \ldots, 2p$, forming a $\mathtt{V}$-*chain* $V_0 \to V_1 \to \cdots \to V_{2p}$.

Consider a $\mathtt{U}$-chain and a $\mathtt{V}$-chain such that each chain has at least p distinct elements.[7] If there is some index $i < p$ such that $V_0 = f(U_i)$ then the pair (U_i, V_0) is called a *slid pair*, and $V_k = f(U_{i+k})$ and $U_{i+k+1} = g(V_k)$ for every

Feistel, and thus the values of M and N are very close: if one wants to encrypt m characters in radix d, then $M = d^{\lceil m/2 \rceil}$ and $N = d^{\lfloor m/2 \rfloor}$.

[6] DV actually use different concrete choices of p and s to aggressively improve the recovery rate.

[7] To test if, say a $\mathtt{U}$-chain $(U_0, \ldots, U_{2p})$ contains at least p distinct elements, we only need to check if $U_0 \notin \{U_1, \ldots, U_{p-1}\}$, since $|\{U_0, \ldots, U_{2p}\}| < p$ if and only if U_0 is within a cycle of length $k < p$ in the functional graph of the permutation $f(g(\cdot))$.

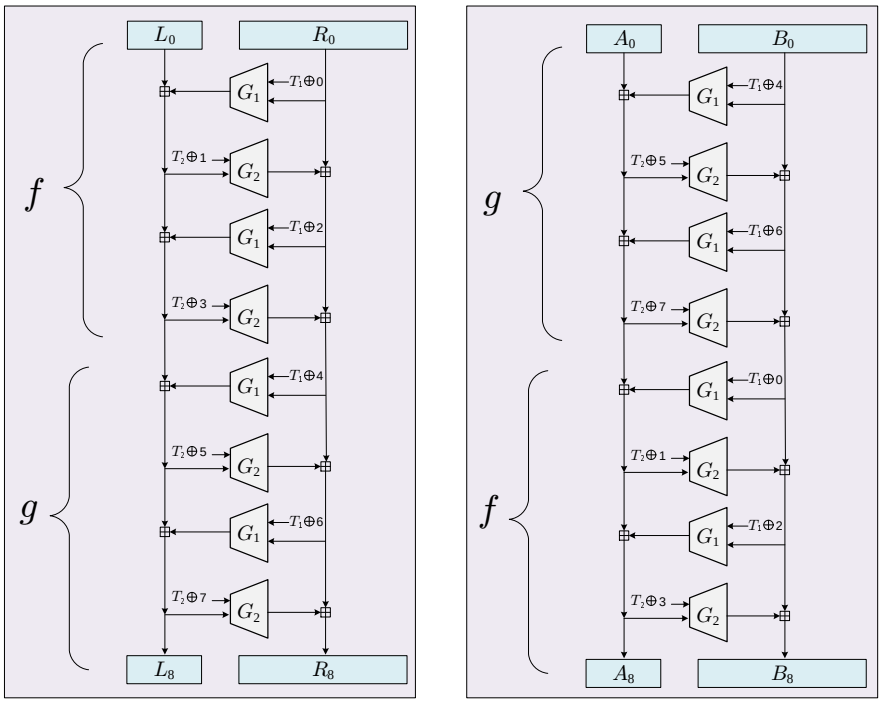

Fig. 2. Left: Encryption $\mathsf{F.E}(K, T, \cdot)$ as a cascade of 4-round Feistel networks f and g. **Right:** Slided encryption $\mathsf{F.E}(K, T', \cdot)$ as a cascade of g and f, with $T' = T \oplus ([4]_\tau \,\|\, [4]_\tau)$. Here T_1 and T_2 are the left half and right half of the tweak T, respectively. For simplicity, in the picture, instead of writing, say $T_1 \oplus [0]_\tau$, we simply write $T_1 \oplus 0$.

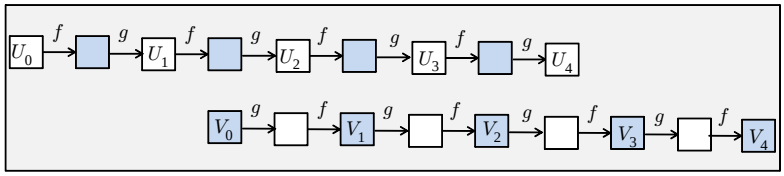

Fig. 3. Illustration of the slide attack. Here (U_1, V_0) is a slid pair.

$0 \le k < p$. Likewise, if there is some index $j < p$ such that $U_0 = g(V_j)$ then the pair (V_j, U_0) is also called a slid pair, and $U_k = g(V_{j+k})$ and $V_{j+k+1} = f(U_k)$ for every $0 \le k < p$. See Fig. 3 for an illustration.

Suppose that somehow we manage to find a slid pair. Then we get p input/output pairs for f, and can run TF to recover the codebook of f. Likewise, we can also recover the codebook of g. By composing the codebook of f and g, we finally recover the codebook of F on tweak T. We can also compose g and f to recover the codebook of F on tweak T'.

Procedure $\mathsf{SD}^{\textsc{Enc}}()$

Pick arbitrary $T \in \{0,1\}^{2\tau}$; $T' \leftarrow T \oplus ([4]_\tau \,\|\, [4]_\tau)$
$UCh \leftarrow \mathsf{MakeChain}^{\textsc{Enc}}(T)$; $VCh \leftarrow \mathsf{MakeChain}^{\textsc{Enc}}(T')$
For $\boldsymbol{U} \in UCh, \boldsymbol{V} \in VCh$ do
  $\boldsymbol{C} \leftarrow\!\!\$\ \mathsf{Slide}(\boldsymbol{U}, \boldsymbol{V})$; If $\boldsymbol{C} \neq \perp$ then return $(T, \boldsymbol{C})$
  $\boldsymbol{C}' \leftarrow\!\!\$\ \mathsf{Slide}(\boldsymbol{V}, \boldsymbol{U})$; If $\boldsymbol{C}' \neq \perp$ then return $(T', \boldsymbol{C}')$

Procedure $\mathsf{MakeChain}^{\textsc{Enc}}(T)$

$p \leftarrow \max\left\{\lfloor 2^{1/3} M^{2/3} N \rfloor, \lceil M(\ln(M)+5) \rceil\right\}$
$s \leftarrow \lfloor \sqrt{MN/2p} \rfloor$; $S, UCh \leftarrow \emptyset$
For $i = 1$ to s do $U \leftarrow\!\!\$\ \mathbb{Z}_M \times \mathbb{Z}_N$; $S \leftarrow S \cup \{U\}$
For $U_0 \in S$ do
  For $i = 1$ to $2p$ do $U_i \leftarrow \textsc{Enc}(T, U_{i-1})$
  If $U_0 \notin \{U_1, \ldots, U_{p-1}\}$ then $UCh \leftarrow UCh \cup \{(U_0, \ldots, U_{2p})\}$
Return UCh

Fig. 4. The blueprint of DV's attack, which is also the main procedure of the SD attack. Procedure $\mathsf{Slide}(\boldsymbol{U}, \boldsymbol{V})$ takes as input two chains $\boldsymbol{U} = (U_0, \ldots, U_{2p})$ and $\boldsymbol{V} = (V_0, \ldots, V_{2p})$, tries to find a slid pair (U_i, V_0), and then uses TF to recover the codebook. Numbers M, N, τ are global parameters.

The code of the blueprint of DV's attack is given in Fig. 4, which is also the main procedure of our SD attack. The two attacks however differ in how they implement procedure Slide for finding a slid pair, among $2s^2p \approx MN$ candidates. DV simply try every possible candidate, by running (a slow version of) the TF algorithm to recover the codebook of f and g. As we will show below, there are often very few slid pairs, and thus DV's attack essentially has to run TF for about $\Theta(MN)$ times, which is very expensive. The key idea in our Slide-then-Differential, which we will elaborate in Section 3.2, is to use some differential analysis to quickly eliminate false candidates.

The number of slid pairs. Clearly, the attack above only works if there exists at least one slid pair. Let P be the random variable for the number of slid pairs. DV use a heuristic[8] to estimate that $\Pr[P \geq 1] \approx 1 - e^{-2s^2p/MN} \approx 1 - 1/e$, under the model that the cascade of f and g is an ideal permutation. We instead give a rigorous lower bound of $\Pr[P \geq 1]$ for a *generic* value p in Lemma 1 in the same model; the proof is in the full version. For $s = 1$ (equivalently, $M < 1024$), we can compute the exact probability $\Pr[P \geq 1]$, but we stress that this result only holds in the model above. The experiments in Section 5 show that empirically, the event $P \geq 1$ happens with higher probability.

[8] While DV only consider balanced Feistel networks, their heuristic can be easily generalized to the general case. For completeness, in the proof of Lemma 1, we also describe this heuristic argument.

Lemma 1. *Let $M \geq N \geq 8$ and let $\mathsf{F} = \mathbf{FF3}[M, N, \tau, \boxplus]$. Let $p \geq 1$ be an integer such that $p \leq MN/2$ and let $s = \lfloor \sqrt{MN/2p} \rfloor$. Let f and g be as above, and let π be the cascade of f and g. Let P be the random variable for the number of slid pairs. Let $\delta = \frac{2p}{MN} - \frac{(2.5p^2 - 1.5p)}{(MN)^2}$. We will model π as an ideal random permutation on $\mathbb{Z}_M \times \mathbb{Z}_N$.*

(a) If $s = 1$ then $\Pr[P \geq 1] = \delta \approx \frac{3}{8}$.
(b) If $s \geq 2$ then $\Pr[P \geq 1] \geq \frac{s^2\delta}{2} \approx \frac{1}{2}$.

Above, we show that it is quite likely that there are one or more slid pairs. However, often there will be very few of them. Lemma 2 below bounds the expected number of slid pairs for a *generic* value of p; the proof is in the full version. Combining this with Markov's inequality, one can show that with probability at least 0.8, there are at most 5 slid pairs.

Lemma 2. *Let $M \geq N \geq 64$ and let $\mathsf{F} = \mathbf{FF3}[M, N, \tau, \boxplus]$. Let $p \geq 1$ be an integer such that $p \leq MN/2$ and let $s = \lfloor \sqrt{MN/2p} \rfloor$. Let f and g be as above, and let π be the cascade of f and g. Let P be the random variable for the number of slid pairs. If we model π as an ideal random permutation on $\mathbb{Z}_M \times \mathbb{Z}_N$ then $\mathbf{E}[P] \leq \frac{2s^2p}{MN} \leq 1$.*

3.2 Distinguishing Slid-pair Candidates

As shown above, we often have very few slid pairs, among $2s^2p \approx MN$ candidates, and using TF to find the actual slid pairs is an overkill. Note that each candidate gives us p plaintext/ciphertext pairs for f. If a candidate is indeed a slid pair, then the ciphertexts for f are indeed the images of the corresponding plaintexts under f, otherwise we can view them as produced from an ideal permutation on $\mathbb{Z}_M \times \mathbb{Z}_N$. The analogous claim also holds for g. A natural solution is to find a quick distinguishing attack for 4-round Feistel, so that we can tell the true candidates from the false ones.

Our distinguishing attack on 4-round Feistel. Below, we will give a distinguishing attack Left-Half Differential (LHD) of 4-round Feistel such that (1) in the ideal world, it returns 1 with probability at most $\frac{N^{5/6}}{M^{4/3}}$, and (2) in the real world, it returns 1 with probability at least $1 - \frac{\sqrt{N}}{8M} - \frac{9.7}{M} - \frac{0.88N^{3/4}}{M^{3/2}}$. For each slid-pair candidate, we run LHD on the plaintext/ciphertext pairs of f, and also on those of g. We will accept the candidate if LHD returns 1 in both cases. Then, for each false candidate, the chance that we incorrectly accept it is at most $N^{5/3}/M^{8/3}$. Since we have at most MN false candidates, on average we will have at most

$$MN \cdot \frac{N^{5/3}}{M^{8/3}} = \frac{N^{8/3}}{M^{5/3}} \leq N$$

false candidates that survived our test. In addition, for each true candidate, the chance that we incorrectly reject it is at most

$$1 - \left(1 - \frac{\sqrt{N}}{8M} - \frac{9.7}{M} - \frac{0.88N^{3/4}}{M^{3/2}}\right)^2 \leq 0.37$$

for $M, N \geq 64$. We note that our bounds are very conservative, since we obtain them via Chebyshev's inequality, which is loose. In fact, our empirical results, presented in Section 5, significantly outperform the theoretical estimates. In particular, on average just one (possibly false) candidate survives our test, and we never incorrectly reject a true candidate.

Proceeding into details, the LHD algorithm is based on the following Lemma 3, which is a generalization of a result by Patarin for balanced, boolean Feistel [24]. A more general version of Lemma 3 appears in [5] for a Feistel network of an even number of rounds, but this result only provides a (very tight) approximation of the bound, instead of an exact one. The proof is in the full version

Lemma 3. *Let $M, N \geq 8$ be integers and $\overline{\mathsf{F}} = \mathbf{Feistel}[4, M, N, \boxplus]$. Let X and X' be two distinct messages in $\mathbb{Z}_M \times \mathbb{Z}_N$ such that $\mathsf{RH}(X) = \mathsf{RH}(X')$. Let C and C' be the ciphertexts of X and X' under $\overline{\mathsf{F}}$ with a uniformly random key. Then*

$$\Pr[\mathsf{LH}(C) \boxminus \mathsf{LH}(C') = \mathsf{LH}(X) \boxminus \mathsf{LH}(X')] = \frac{M+N-1}{MN} \quad .$$

Lemma 3 above shows that if we encrypt two messages X and X' of the same right segment under a 4-round Feistel network, then there will be some bias in the distribution of the ciphertexts C and C': (1) the chance that $\mathsf{LH}(C) \boxminus \mathsf{LH}(C') = \mathsf{LH}(X) \boxminus \mathsf{LH}(X')$ is $\frac{M+N-1}{MN}$, (2) had we instead sampled C and C' uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$, this probability would have been just $\frac{N}{MN-1}$. Our distinguishing attack LHD will amplify this bias, by using several messages of the same right segments.

Random variables $X_1, \ldots, X_m \in \mathbb{Z}_M \times \mathbb{Z}_N$ are *t-wise right-matching* if they satisfy the following constraints:

- If we partition $X_1, \ldots, X_m$ into groups $P_1, \ldots, P_d$ according to their right segments then $d \leq t$.
- Within each partition P_i, the left segments of the messages in P_i are uniformly distributed over $\mathbb{Z}_M$, subject to the constraint that those left segments are distinct.

Our attack LHD takes as input m messages $(X_1, \ldots, X_m)$ that are t-wise right-matching and their ciphertexts $(C_1, \ldots, C_m)$, where $m = \left\lceil \frac{p}{N} \cdot \lceil 32N^{1/6} \rceil \right\rceil$ and $t = \lceil \frac{mM}{p} \rceil$. The code of LHD is given in Fig. 5. Informally, LHD will compute *count*, the number of pairs X_i and X_j, with $i < j$, such that $\mathsf{RH}(X_i) = \mathsf{RH}(X_j)$ and $\mathsf{LH}(C_i) \boxminus \mathsf{LH}(C_j) = \mathsf{LH}(X_i) \boxminus \mathsf{LH}(X_j)$. If the ciphertexts are produced by a 4-round Feistel network then from Lemma 3, the expected value of *count* is $\frac{M+N-1}{MN} \cdot \mathit{size}$, where *size* is the number of pairs X_i, X_j such that $i < j$ and $\mathsf{RH}(X_i) = \mathsf{RH}(X_j)$. If the ciphertexts are produced by a truly random permutation on $\mathbb{Z}_M \times \mathbb{Z}_N$ then the expected value of *count* is $\frac{N}{MN-1} \cdot \mathit{size}$. The algorithm LHD will return 1 if *count* is greater than the weighted average $\left(\frac{1}{5} \cdot \frac{M+N-1}{MN} + \frac{4}{5} \cdot \frac{N}{MN-1}\right)\mathit{size}$, otherwise it will return 0.

Procedure $\mathsf{LHD}(X_1, \ldots, X_m, C_1, \ldots, C_m)$
Partition $X_1, \ldots, X_m$ by the right segments into groups $P_1, \ldots, P_d$
$\mathit{count} \leftarrow 0$; $\Delta \leftarrow \frac{1}{5} \cdot \frac{M+N-1}{MN} + \frac{4}{5} \cdot \frac{N}{MN-1}$; $\mathit{size} \leftarrow \sum_{\ell=1}^{d} \frac{|P_\ell|(|P_\ell|-1)}{2}$
For $\ell \leftarrow 1$ to d do
  For $X_i, X_j \in P_\ell$ with $i < j$ do
    If $\mathsf{LH}(C_i) \boxminus \mathsf{LH}(C_j) = \mathsf{LH}(X_i) \boxminus \mathsf{LH}(X_j)$ then $\mathit{count} \leftarrow \mathit{count} + 1$
If $\mathit{count} \geq \Delta \cdot \mathit{size}$ then return 1 else return 0

Fig. 5. Distinguishing attack LHD on four-round Feistel.

Implementing LHD. The code in Fig. 5 describes just the conceptual view of LHD for ease of understanding. Implementing it efficiently requires some care. First, messages $X_1, \ldots, X_m$ will be grouped according to their right segments, by a one-time preprocessing that we will describe in Section 3.3. Thus the partitioning takes only linear time. Let $P_1, \ldots, P_d$ be the resulting partitions, and let $|m_\ell| = |P_\ell|$, for every $\ell \leq d$. In the for loops, if we naively follow the code, then the running time would be

$$\sum_{\ell=1}^{d} \Omega(m_\ell^2) = \Omega(m^2/d) = \Omega(M^{1/3}N^{7/6}),$$

which is expensive. Instead, we will execute as in Fig. 6. That is,

- For each fixed $\ell \leq d$, we want to find count_ℓ, the number of pairs (i, j) such that $i < j$ and $X_i, X_j \in P_\ell$ and $\mathsf{LH}(C_i) \boxminus \mathsf{LH}(X_i) = \mathsf{LH}(C_j) \boxminus \mathsf{LH}(X_j)$. We then can compute count via $\mathit{count}_1 + \cdots + \mathit{count}_d$.
- Thus for each $\ell \leq d$, we create an empty hash table H_ℓ of key-value pairs and initialize $\mathit{count}_\ell \leftarrow 0$. We process P_ℓ so that eventually, for each entry in H_ℓ, its key is a number $Z \in \mathbb{Z}_M$ and its value indicates how many $X_k \in P_\ell$ that $\mathsf{LH}(C_k) \boxminus \mathsf{LH}(X_k) = Z$.
- Finally, we iterate through all keys of H_ℓ. For each key Z, we find its value v and update $\mathit{count}_\ell \leftarrow \mathit{count}_\ell + \frac{v(v-1)}{2}$.

The total running time of this implementation is $O(m) = O(M^{2/3}N^{1/6})$.

Analysis of LHD. Lemma 4 below bounds the probability that LHD outputs 1 in the ideal world, for *generic* m and t, and also for a *generic* weighted average $\Delta = \lambda \cdot \frac{M+N-1}{MN} + (1-\lambda)\frac{N}{MN-1}$; see the full version for the proof. If we pick $m = \left\lceil \frac{p}{N} \cdot \lceil 32N^{1/6} \rceil \right\rceil$, $t = \lceil \frac{mM}{p} \rceil$, and $\lambda = \frac{1}{5}$ as suggested then this probability is about $\frac{N^{5/6}}{M^{4/3}}$.

Lemma 4. *Let $M \geq N \geq 8$ be integers, and let $0 < \lambda < 1$ be a real number. Let $m > t \geq 1$ be integers. Let $X_1, \ldots, X_m$ be t-wise right-matching messages, and let $C_1, \ldots, C_m$ be their ciphertexts, respectively, under an ideal random permutation on $\mathbb{Z}_M \times \mathbb{Z}_N$. Let V be the random variable of the number of pairs X_i and X_j, with $i < j$, such that $\mathsf{RH}(X_i) = \mathsf{RH}(X_j)$ and $\mathsf{LH}(C_i) \boxminus \mathsf{LH}(C_j) = \mathsf{LH}(X_i) \boxminus \mathsf{LH}(X_j)$.*

Procedure $\mathsf{LHD}(X_1, \ldots, X_m, C_1, \ldots, C_m)$

// $X_1, \ldots, X_m$ are already grouped according to their right segments
Partition $X_1, \ldots, X_m$ by the right segments into groups $P_1, \ldots, P_d$
$\mathit{count} \leftarrow 0$; $\Delta \leftarrow \frac{1}{5} \cdot \frac{M+N-1}{MN} + \frac{4}{5} \cdot \frac{N}{MN-1}$; $\mathit{size} \leftarrow \sum_{\ell=1}^{d} \frac{|P_\ell|(|P_\ell|-1)}{2}$
For $\ell \leftarrow 1$ to d do
  $\mathit{count}_\ell \leftarrow 0$; Initialize a hash table H_ℓ
  For $X_k \in P_\ell$ do
    $Z \leftarrow \mathsf{LH}(C_k) \boxminus \mathsf{LH}(X_k)$; $v \leftarrow H_\ell[Z]$
    If $v = \perp$ then $H_\ell[Z] \leftarrow 1$ else $H_\ell[Z] \leftarrow v + 1$
  For each key Z in H_ℓ do $v \leftarrow H_\ell[Z]$; $\mathit{count}_\ell \leftarrow \mathit{count}_\ell + \frac{v(v-1)}{2}$
$\mathit{count} \leftarrow \mathit{count}_1 + \cdots + \mathit{count}_d$
If $\mathit{count} \geq \Delta \cdot \mathit{size}$ then return 1 else return 0

Fig. 6. Implementation of LHD.

Let size be the number of pairs X_i, X_j such that $i < j$ and $\mathsf{RH}(X_i) = \mathsf{RH}(X_j)$, and $\Delta = \lambda \cdot \frac{M+N-1}{MN} + (1-\lambda)\frac{N}{MN-1}$. Then

$$\Pr[V \geq \Delta \cdot \mathit{size}] \leq \frac{N^2}{\lambda^2(M-2)^2}\Big(\frac{1}{MN-2} + \frac{2MN-2}{N(m^2/t-m)}\Big) \ .$$

Lemma 5 below bounds the probability that LHD fails to output 1 in the real world, again for generic m and t, and for a generic weighted average $\Delta = \lambda \cdot \frac{M+N-1}{MN} + (1-\lambda)\frac{N}{MN-1}$; see the full version for the proof. If we use $m = \lceil \frac{p}{N} \cdot \lceil 32N^{1/6} \rceil \rceil$, $t = \lceil \frac{mM}{p} \rceil$, and $\lambda = \frac{1}{5}$ as suggested then this probability is about $\frac{\sqrt{N}}{8M} + \frac{9.7}{M} + \frac{0.88N^{3/4}}{M^{3/2}}$.

Lemma 5. *Let $M \geq N \geq 8$ be integers and let $0 < \lambda < 1$ be a real number. Let $m > t \geq 1$ be integers. Let $X_1, \ldots, X_m$ be t-wise right-matching messages and let $C_1, \ldots, C_m$ be their ciphertexts, respectively, under $\overline{\mathsf{F}} = \mathbf{Feistel}[4, M, N, \boxplus]$ with a uniformly random key. Let V be the random variable of the number of pairs X_i and X_j, with $i < j$, such that $\mathsf{RH}(X_i) = \mathsf{RH}(X_j)$ and $\mathsf{LH}(C_i) \boxminus \mathsf{LH}(C_j) = \mathsf{LH}(X_i) \boxminus \mathsf{LH}(X_j)$. Let $\Delta = \lambda \cdot \frac{M+N-1}{MN} + (1-\lambda)\frac{N}{MN-1}$, and let size be the number of pairs X_i, X_j such that $i < j$ and $\mathsf{RH}(X_i) = \mathsf{RH}(X_j)$. Then*

$$\Pr[V \leq \Delta \cdot \mathit{size}] \leq \frac{2(M+N-1)MN}{(1-\lambda)^2(m^2/t-m)(M-2)^2} + \frac{6.2(M-1)(N-1)}{(1-\lambda)^2 N(M-2)^2} + \frac{4MN}{(1-\lambda)^2(M-2)^2\sqrt{(m^2/t-m)}} \ .$$

Using LHD. The LHD attack requires m chosen plaintexts, but recall that for each slid-pair candidate, we only have p known plaintext/ciphertext pairs for f,

and also p known pairs for g.[9] To find m messages that are $\lceil \frac{mM}{p} \rceil$-wise right-matching, the naive approach is to partition p given messages according to their right segments, and let $P_1, \ldots, P_M$ be the (possibly empty) partitions, with $|P_1| \geq \cdots \geq |P_M|$. We then output m messages from the first (and also biggest) $s = \left\lceil \frac{mM}{p} \right\rceil$ partitions. Since there are at most M partitions, our chosen partitions contain at least $\lceil s \cdot \frac{p}{M} \rceil \geq m$ messages. Moreover, as the given p messages are sampled uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$, within each partition P_i, the left segments of the messages in P_i are sampled uniformly without replacement from $\mathbb{Z}_M$.

The naive approach above is however very expensive. Totally, for $\Theta(MN)$ slid-pair candidates, it uses $\Omega(MNp) = \Omega(M^{5/3}N^2)$ time just to find their right-matching messages. In the next section we'll describe a one-time preprocessing of $O(MN)$ time such that later for each slid-pair candidate, we need only $O(m)$ time to find their right-matching messages to run LHD tests. Only for candidates that survive the LHD tests that we extract their p plaintext/ciphertext pairs from the corresponding chains to run TF.

Eliminating false negatives. Since our distinguishing test of slid-pair candidates above might occasionally produce false negatives, we still have to use TF to eliminate the survived false candidates. The TF algorithm, after recovering the round functions (G_1, G_2, G_3, G_4), will compute the outputs of the first $3M \leq p$ plaintexts under a 4-round Feistel network with the round functions (G_1, G_2, G_3, G_4), and compare them with the corresponding ciphertexts. By a simple counting argument, one can show that it is extremely likely that TF will reject all these false candidates. Specifically, for a false candidate, view its $3M$ associated ciphertexts as the outputs of the $3M$ plaintexts under an ideal permutation on $\mathbb{Z}_M \times \mathbb{Z}_N$. On the one hand, there are at most $M^{2N}N^{2M} \leq M^{2M}N^{2N}$ choices of four round functions for a 4-round Feistel network on $\mathbb{Z}_M \times \mathbb{Z}_N$. On the other hand, if we sample $3M$ ciphertexts uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$, there are

$$MN \cdots (MN - 3M + 1) \geq (MN - 3M)^{3M} \geq M^{3M}(N-3)^{3N}$$

equally likely outputs. Since we have to deal with at most MN false candidates, the chance that the TF algorithm fails to eliminate all false candidates is at most

$$\frac{MN \cdot M^{2M}N^{2N}}{M^{3M}(N-3)^{M+2N}} = \frac{N^{2N+1}}{M^{3M-1}(N-3)^{3N}} \leq \frac{1}{M^{3M-1}} \ .$$

Relation to prior Feistel attacks. Our LHD attack generalizes Patarin's distinguishing attack on 4-round balanced, boolean Feistel [24]; Patarin's result was later rediscovered by Aiello and Venkatesan [1]. To attack Feistel networks on $2n$-bit strings, those papers suggest using messages $X_1, \ldots, X_m$ of the same right half, with $m = \Theta(2^{n/2})$. However, both papers only compute the expected value

[9] Recall that in our attack, we require $M \geq N \geq 64$. This ensures that $m \leq p$, so that we can select m right-matching messages from p known messages.

of the number of pairs (i, j) such that $1 \leq i < j \leq m$ and $\mathsf{LH}(C_i) \oplus \mathsf{LH}(C_j) = \mathsf{LH}(X_i) \oplus \mathsf{LH}(X_j)$ in both the real and ideal worlds. Consequently, they cannot analyze the advantage of their attack, and can only suggest an asymptotic value of m. Our Lemma 4 and Lemma 5 allow one to fill this gap. By using $N = M = 2^n$, $t = 1$, $m = c \cdot 2^{n/2}$, and $\lambda = \frac{1}{2}$, the attack in [24, 1] achieves advantage around $1 - \frac{24}{c^2} - \frac{29}{2^n} - \frac{16}{c \cdot 2^{n/2}}$. Thus to achieve advantage $1/2$, for $n \geq 16$, we can use $c = 7$.

The attack in [24, 1] however cannot be used in place of LHD. Recall that we want a distinguishing attack that outputs 1 with probability around $1/\sqrt{N}$ or smaller in the ideal world, so that it can be used to eliminate most false slid-pair candidates. Using $m = \Theta(\sqrt{N})$ messages as suggested in [24, 1] does not meet this requirement, as the attack will output 1 with constant probability in the ideal world, according to Lemma 4.

3.3 Slide-then-Differential Attack

In this Section, we describe how to combine DV's slide attack with the LHD attack above, resulting in our Slide-then-Differential attack.

Speeding up with preprocessing. Recall that we have $\Theta(MN)$ slid-pair candidates, and for each such candidate we have to process $\Theta(p)$ pairs of plaintext/ciphertext. At the first glance it seems that we are doomed with $\Omega(pMN) = \Omega(M^{5/3}N^2)$ time. However, we will perform a one-time preprocessing using $O(MN)$ time. After this preprocessing, for every slid-pair candidate, we can extract m right-matching messages for f in $O(m) = O(M^{2/3}N^{1/6})$ time, and even better, those messages are already grouped according to their right segments. The same running time would be needed to extract messages for g. We then can run LHD to eliminate most false slid-pair candidates.

Proceeding into details, suppose that we have a $\mathtt{U}$-chain $U_0 \to U_1 \to \cdots \to U_{2p}$ and a $\mathtt{V}$-chain $V_0 \to V_1 \to \cdots \to V_{2p}$, and we want to check if (U_i, V_0) is a slid pair, for every $k \leq \{0, 1, \ldots, p-1\}$. For each slid-pair candidate (U_i, V_0), the known plaintext/ciphertext pairs for f are (U_{i+k}, V_k) for $k \leq 2p - i$, and the known plaintext/ciphertext pairs for g are (V_k, U_{i+k+1}), for $k \leq 2p - k - 1$.

- In order to preprocess these p slid-pair candidates for f, note that they all use plaintexts $U_{p+1}, \ldots, U_{2p}$. So we will partition these plaintexts by their right segments into (possibly empty) groups $P_1, \ldots, P_M$, with $|P_1| \geq \cdots \geq |P_M|$. We then store m messages U_j from the first $\lceil 32N^{1/6} \rceil$ partitions, together with their indices j, in a list L. Later, for a slid-pair candidate (U_i, V_0), we iterate through pairs (U_j, j) in L, and for each such pair, the corresponding ciphertext of U_j for f is V_{j-i}, which takes $O(1)$ time to find if we store $(U_0, \ldots, U_{2p})$ and $(V_0, \ldots, V_{2p})$ in arrays.
- Preprocessing for g is similar, but note that the p candidates all use plaintexts $V_0, V_1, \ldots, V_{p-1}$.

For a pair of $\mathtt{U}$ chain and $\mathtt{V}$ chain, partitioning takes $O(p + M)$ time, and by using a max-heap, we can find the $\lceil 32N^{1/6} \rceil$ biggest partitions in $O(M)$ time,

Procedure $\mathsf{Slide}(\boldsymbol{U}, \boldsymbol{V})$
$(U_0, \ldots, U_{2p}) \leftarrow \boldsymbol{U}$; $(V_0, \ldots, V_{2p}) \leftarrow \boldsymbol{V}$; $(Z_1, \ldots, Z_{MN}) \leftarrow \mathbb{Z}_M \times \mathbb{Z}_N$
$L \leftarrow \mathsf{Process}((U_{p+1}, p+1), \ldots, (U_{2p}, 2p))$
$L' \leftarrow \mathsf{Process}((V_0, 0), \ldots, (V_{p-1}, p-1))$
For $i = 0$ to $p-1$ do // Check if (U_i, V_0) is a slid pair
  If $(\mathsf{Dist}(L, \boldsymbol{V}, -i) \wedge \mathsf{Dist}(L', \boldsymbol{U}, i+1))$ then
    $\boldsymbol{X} \leftarrow (U_p, \ldots, U_{2p-1})$; $\boldsymbol{Y} \leftarrow (V_{p-i}, \ldots, V_{2p-1-i})$
    $\boldsymbol{X}^* \leftarrow (V_0, \ldots, V_{p-1})$; $\boldsymbol{Y}^* \leftarrow (U_{i+1}, \ldots, U_{i+p})$
    $f \leftarrow\!\!\$\ \mathsf{Recover}(\boldsymbol{X}, \boldsymbol{Y})$; $g \leftarrow\!\!\$\ \mathsf{Recover}(\boldsymbol{X}^*, \boldsymbol{Y}^*)$
    If $f \neq \perp$ and $g \neq \perp$ then
      For $i = 1$ to MN do $C_i \leftarrow g(f(Z_i))$
      Return $(C_1, \ldots, C_{MN})$ // Codebook is $(Z_1, C_1), \ldots, (Z_{MN}, C_{MN})$

Procedure $\mathsf{Process}((X_1, r_1), \ldots, (X_p, r_p))$ // Preprocessing
$L \leftarrow \emptyset$; $m \leftarrow \lceil \frac{p}{N} \cdot \lceil 32N^{1/6} \rceil \rceil$
Partition $(X_1, r_1), \ldots, (X_p, r_p)$ by the right segments
Let $P_1, \ldots, P_M$ be the resulting partitions, with $|P_1| \geq \cdots \geq |P_M|$
For $i = 1$ to M, $(X, r) \in P_i$ do: If $|L| < m$ then $L \leftarrow L \cup \{(X, r)\}$
Return L

Procedure $\mathsf{Dist}(L, \boldsymbol{V}, k)$ // Running LHD with preprocessed list L
$i \leftarrow 1$, $(V_0, \ldots, V_{2p}) \leftarrow \boldsymbol{V}$; $m \leftarrow |L|$
For $(Z, r) \in L$ do $X_i \leftarrow Z$; $C_i \leftarrow V_{r+k}$; $i \leftarrow i + 1$
Return $\mathsf{LHD}(X_1, \ldots, X_m, C_1, \ldots, C_m)$

Procedure $\mathsf{Recover}(\boldsymbol{X}, \boldsymbol{Y})$
$(X_0, \ldots, X_{p-1}) \leftarrow \boldsymbol{X}$; $(Y_0, \ldots, Y_{p-1}) \leftarrow \boldsymbol{Y}$
$(F_1, F_2, F_3, F_4) \leftarrow\!\!\$\ \mathsf{TF}(X_0, \ldots, X_{p-1}, Y_0, \ldots, Y_{p-1})$
Let f be the 4-found Feistel of round functions (F_1, F_2, F_3, F_4)
// Function f will be $\perp$ if TF does not fully recover (F_1, F_2, F_3, F_4)
Return f

Fig. 7. The implementation of procedure Slide in the SD attack. The numbers M and N are global parameters. We assume that there is a global total ordering on the domain $\mathbb{Z}_M \times \mathbb{Z}_N$, so that we can write $(Z_1, \ldots, Z_{MN}) \leftarrow \mathbb{Z}_M \times \mathbb{Z}_N$.

and extracting m messages from those partitions takes $O(m)$ time. Summing up, for a pair of U chain and V chain, the running time of the preprocessing is $O(p)$. Hence, totally, for s^2 pairs of U chains and V chains, the overall running time of the preprocessing is $O(s^2 p) = O(MN)$.

PUTTING THINGS TOGETHER. By combining the LHD attack and the preprocessing, one can implement procedure Slide as in Fig. 7. Thus the SD attack uses $O(sp) = O(M^{5/6}N)$ queries and space, and its running time is $O(MN)$ for the

preprocessing, $O(MN \cdot m) = O(M^{5/3}N^{7/6})$ for running LHD, and expectedly, $O(M^{5/3}N)$ for running TF. Hence the total running time of SD is $O(M^{5/3}N^{7/6})$.

Improving the recovery rate. To improve the recovery rate of SD, one can run the attack several times with different tweak pairs $(T_1, T_1 \oplus \mathsf{Mask}), \ldots, (T_r, T_r \oplus \mathsf{Mask})$, where $\mathsf{Mask} = [4]_\tau \,\|\, [4]_\tau$. If $(T_i \oplus T_j)[1\colon \tau], (T_i \oplus T_j)[\tau + 1\colon 2\tau] \notin \{[0]_\tau, \ldots, [7]_\tau\}$ for every $i \neq j$ then those r instances of SD will call AES on different τ-bit prefixes. If we model AES as a good PRF then the results of those SD instances are independent. Hence if the recovery rate of SD is ϵ then running it for r times will have recovery rate $1 - (1 - \epsilon)^r$.

4 Attacking 4-round Feistel-based Blockciphers

In this section, we generalize and improve DV's known-plaintext codebook-recovery attack on Feistel-based, 4-round blockciphers where the Feistel network might be unbalanced. In particular, on a four-round balanced Feistel of domain size N^2, DV's attack runs in $O(N^3)$ expected time, but our attack, which we name *Triangle-Finding* (TF), runs in only $O(N^{5/3})$ expected time. Both our attack and DV's rely on a conjecture of Feistel networks that DV empirically verified for balanced Feistel of domain $\{0,1\}^{2n}$, for $n \in \{1, \ldots, 9\}$. We prove that this conjecture indeed holds, making both attacks unconditional.

Let

$$p = \max\Big\{ \lfloor 2^{1/3} M^{2/3} N \rfloor, \lceil M(\ln(M) + 5) \rceil \Big\} \ .$$

In our attack, we suppose that we are given p pairs $(X_1, C_1), \ldots, (X_p, C_p)$ of plaintext/ciphertext under a four-round Feistel network $\mathsf{F} = \mathbf{Feistel}[4, M, N, \boxplus]$ with a uniformly random key, where $M \geq N \geq 64$ and the plaintexts are chosen uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$. To simplify our asymptotic analysis, we assume that $N = \Omega(\sqrt{M})$, which applies to the setting of the FF3 scheme, since FF3 uses near-balanced Feistel. Thus $p \in O(M^{2/3}N)$.

In our attack, we need a graph representation of the first p plaintext/ciphertext pairs, and a few algorithms, which we will elaborate in Section 4.1. We then describe our TF attack in Section 4.2.

4.1 Differential Graph and Its Triangles

Differential graph. Let $\mathcal{G} = (V, E)$ be the following directed graph. First, for each $i \neq j$, we create a node $v_{i,j}$ with label $\mathsf{Label}(v_{i,j}) = \mathsf{RH}(C_i) \boxminus \mathsf{RH}(C_j)$ if $\mathsf{RH}(X_i) = \mathsf{RH}(X_j)$ and $\mathsf{LH}(C_i) \boxminus \mathsf{LH}(X_i) = \mathsf{LH}(C_j) \boxminus \mathsf{LH}(X_j)$. Next, for every two nodes $v_{i,j}$ and $v_{k,\ell}$ such that i, j, k, ℓ are distinct, we create a directed edge $(v_{i,j}, v_{k,\ell})$ if $\mathsf{LH}(C_j) = \mathsf{LH}(C_k)$ and the following non-degeneracy conditions hold:

(1) $\mathsf{LH}(C_i) \neq \mathsf{LH}(C_\ell)$,
(2) $\mathsf{RH}(X_i) \neq \mathsf{RH}(X_k)$,
(3) $\mathsf{RH}(X_j) \boxminus \mathsf{RH}(X_k) \neq \mathsf{RH}(C_j) \boxminus \mathsf{RH}(C_k)$, and

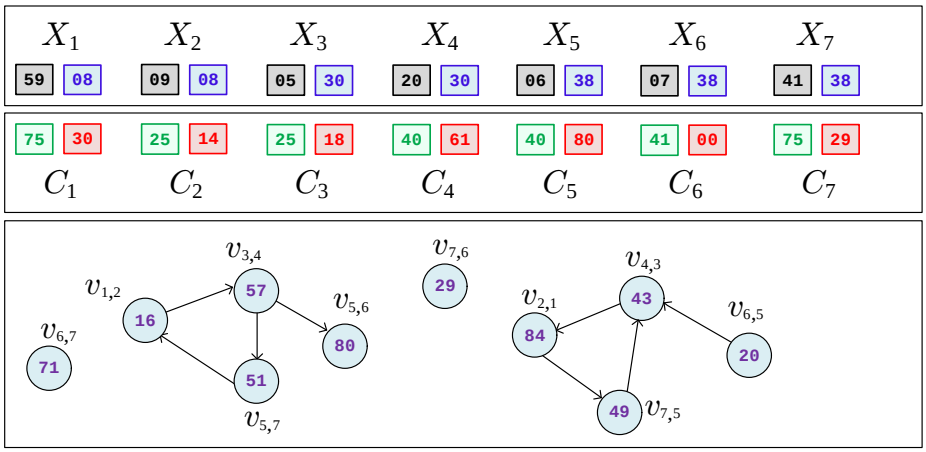

Fig. 8. Top: Seven pairs of plaintext/ciphertext $(X_1, C_1), \ldots, (X_7, C_7)$ on $\mathbb{Z}_M \times \mathbb{Z}_N$, with $M = N = 100$. **Bottom:** Differential graph $\mathcal{G}$ constructed from those seven pairs.

(4) $\mathsf{LH}(X_i) \boxminus \mathsf{LH}(X_j) \neq \mathsf{LH}(X_k) \boxminus \mathsf{LH}(X_\ell)$.

See Fig. 8 for an illustration of the graph $\mathcal{G}$. We call $\mathcal{G}$ the *differential graph* of the plaintext/ciphertexts pairs $(X_1, C_1), \ldots, (X_p, C_p)$.

GOOD VERSUS BAD NODES. For a node $v_{i,j}$ in the differential graph $\mathcal{G}$, we say that it is *good* if $\mathsf{RH}(X_i^2) = \mathsf{RH}(X_j^2)$; otherwise we say that it is *bad*. Lemma 6 below characterizes an important property of good nodes; it is a direct generalization of a result in DV's work; see the full version for the proof.

Lemma 6. *Let $M, N \geq 10$ be integers and let $\mathsf{F} = \mathbf{Feistel}[4, M, N, \boxplus]$. Let $(X_1, C_1), \ldots, (X_p, C_p)$ be plaintext/ciphertext pairs under F with a uniformly random key, where the plaintexts are chosen uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$. Let $\mathcal{G} = (V, E)$ be the differential graph of those pairs, and let (G_1, G_2, G_3, G_4) be the functions specified by the secret key of F. Then for any good node $v_{i,j} \in V$, we have $\mathsf{Label}(v_{i,j}) = G_4(\mathsf{LH}(C_i)) \boxminus G_4(\mathsf{LH}(C_j))$.*

The following Lemma 7 computes the average number of good and bad nodes, and estimates the average number of edges in the differential graph; see the full version for the proof.

Lemma 7. *Let $M, N \geq 10$ be integers and let $\mathsf{F} = \mathbf{Feistel}[4, M, N, \boxplus]$. Let $(X_1, C_1), \ldots, (X_p, C_p)$ be plaintext/ciphertext pairs under F with a uniformly random key, where the plaintexts are chosen uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$. Let $\mathcal{G} = (V, E)$ be the differential graph of those pairs, and let Z be the random variable for the number of good nodes in $\mathcal{G}$. Then*

(a) $\mathbf{E}[|V|] = \frac{p(p-1)(M-1)(M+N-1)}{MN(MN-1)}$.
(b) $\mathbf{E}[Z] = \frac{p(p-1)(M-1)}{(MN-1)N}$.

(c) $\mathbf{E}[|E|] \leq \frac{p!}{(p-4)!} \cdot \frac{(M+N)^2}{M^3N^4}$.

Since $p = O(M^{2/3}N)$ and $M \geq N$, from Lemma 7, on average, the differential graph $\mathcal{G}$ contains about $O(M^{4/3})$ nodes, and the majority of them are good. In addition, there are on average $O(M^{5/3})$ edges in $\mathcal{G}$.

A FAST CONSTRUCTION OF DIFFERENTIAL GRAPHS. The naive approach to construct $\mathcal{G}$ would take $\Theta(p^2) = \Theta\big(M^{10/3}\big)$ time just to construct the node set V. We now show how to build $\mathcal{G}$ in $O\big(M^{5/3}\big)$ expected time; the code is given in Fig. 9.

- First, partition the pairs (X_i, C_i) based on $(\mathsf{LH}(C_i) \boxminus \mathsf{LH}(X_i), \mathsf{RH}(X_i))$. Using appropriate data structure, this takes $O(p)$ time.
- For each partition P, enumerate all distinct pairs $(X_i, C_i), (X_j, C_j) \in P$. Each such pair forms a node in V, and its label can be computed accordingly. This takes $O(|V|)$ time.
- Finally, partition V into M groups P_d with $d \in \mathbb{Z}_M$, such that each node $v_{i,j}$ goes to group $P_{\mathsf{LH}(C_j)}$. Also, partition V into M groups S_d with $d \in \mathbb{Z}_M$, such that each node $v_{k,\ell}$ goes to group $S_{\mathsf{LH}(C_k)}$. By enumerating elements in $P_d \times S_d$ (with some pruning) for every $d \in \mathbb{Z}_M$ via appropriate data structure, we can create the edge set E using

$$O\Big(|V| + M + \sum_{i,j,k,\ell} D_{i,j,k,\ell}\Big)$$

  time, where $D_{i,j,k,\ell}$ is the indicator random variable for the event that (i) $v_{i,j} \in V$, (ii) $v_{k,\ell} \in V$, and (iii) $\mathsf{LH}(C_j) = \mathsf{LH}(C_k)$. The summation is taken over all distinct $i, j, k \in \{1, \ldots, p\}$ and $\ell \in \{1, \ldots, p\} \backslash \{k\}$. By pretending that (i), (ii), (iii) are independent, we can heuristically estimate that $\Pr[D_{i,j,k,\ell} = 1] \lessapprox \frac{4}{MN^4}$.

Hence the total expected time is

$$O\Big(p + M + \mathbf{E}[|V|] + \frac{4p^4}{MN^4}\Big) = O\big(M^{5/3}\big) \ .$$

TRIANGLES. Recall that from Lemma 7, on average, the majority of nodes in the differential graph are good. We now describe a method to realize good nodes with high probability. A *triangle* of the graph $\mathcal{G}$ is a directed cycle of length 3. For a triangle $\mathcal{T} = (u_1, u_2, u_3, u_1)$, its *weight* $\mathsf{weight}(\mathcal{T})$ is defined as the sum of the labels, meaning that

$$\mathsf{weight}(\mathcal{T}) = \mathsf{Label}(u_1) \boxplus \mathsf{Label}(u_2) \boxplus \mathsf{Label}(u_3) \ .$$

DV observed that in the balanced setting, for a triangle $\mathcal{T}$, if all of its three nodes are good then its weight is 0. Lemma 8 below shows that their observation also holds in the unbalanced setting; see the full version for the proof.

Procedure $\mathsf{BuildGraph}(X_1, C_1, \ldots, X_p, C_p)$

$V, E \leftarrow \emptyset$; Initialize a hash table H
For $i = 1$ to p do $Z \leftarrow (\mathsf{LH}(C_i) \boxminus \mathsf{LH}(X_i), \mathsf{RH}(X_i))$; $P \leftarrow H[Z]$; $P \leftarrow P \cup \{i\}$
For each key Z in H do
  For $i, j \in H[Z]$, with $i \neq j$, do
    $v_{i,j} \leftarrow (i, j)$; $\mathsf{Label}(v_{i,j}) \leftarrow \mathsf{RH}(C_i) \boxminus \mathsf{RH}(C_j)$; $V \leftarrow V \cup \{v_{i,j}\}$
For $d \in \mathbb{Z}_M$ do $P_d, S_d \leftarrow \emptyset$
For $v \in V$ do $(i, j) \leftarrow v$; $P_{\mathsf{LH}(C_j)} \leftarrow P_{\mathsf{LH}(C_j)} \cup \{v\}$; $P_{\mathsf{LH}(C_i)} \leftarrow P_{\mathsf{LH}(C_i)} \cup \{v\}$
For $d \in \mathbb{Z}_M$ do
  Initialize a hash table H_d
  For $v = (k, \ell) \in S_d$ do $L \leftarrow H_d[k]$; $L \leftarrow L \cup \{v\}$
  For every $u = (i, j) \in P_d$ and every key $k \in H_d$ such that $k \notin \{i, j\}$ do
    For every $v = (k, \ell) \in H_d[k]$ such that $\ell \notin \{i, j\}$ do
      // Check non-degeneracy requirements
      If $\mathsf{LH}(C_i) \neq \mathsf{LH}(C_\ell)$ and $\mathsf{RH}(X_i) \neq \mathsf{RH}(X_k)$
        and $\mathsf{RH}(X_j) \boxminus \mathsf{RH}(X_k) \neq \mathsf{RH}(C_j) \boxminus \mathsf{RH}(C_k)$
        and $\mathsf{LH}(X_i) \boxminus \mathsf{LH}(X_j) \neq \mathsf{LH}(X_k) \boxminus \mathsf{LH}(X_\ell)$ then $E \leftarrow E \cup \{(u, v)\}$
Return $\mathcal{G} = (V, E)$

Fig. 9. Code for building the differential graph $\mathcal{G} = (V, E)$ of $X_1, C_1, \ldots, X_p, C_p$.

Lemma 8. *Let $M, N \geq 10$ be integers and let $\mathsf{F} = \mathbf{Feistel}[4, M, N, \boxplus]$. Let $(X_1, C_1), \ldots, (X_p, C_p)$ be plaintext/ciphertext pairs under F with a uniformly random key, where the plaintexts are chosen uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$. Let $\mathcal{G} = (V, E)$ be the differential graph of those pairs. For a triangle $\mathcal{T}$ in $\mathcal{G}$, if all the three nodes of $\mathcal{T}$ are good then $\mathsf{weight}(\mathcal{T}) = 0$.*

Above, we show that a triangle whose nodes are all good will have weight 0. The following Lemma 9 shows that the converse holds with very high probability; see the full version for the proof. This proves a conjecture in DV's work [15] that they empirically verified for the balanced, boolean case $M = N = 2^n$, with $n \in \{2, 3, \ldots, 9\}$. We also give a rigorous lower bound for the expected number of triangles whose all nodes are good, whereas DV could only give a heuristic estimation of this number.

Lemma 9. *Let $M, N \geq 19$ be integers and let $\mathsf{F} = \mathbf{Feistel}[4, M, N, \boxplus]$. Let $(X_1, C_1), \ldots, (X_p, C_p)$ be plaintext/ciphertext pairs under F with a uniformly random key, where the plaintexts are chosen uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$. Let $\mathcal{G} = (V, E)$ be the differential graph of those pairs.*

(a) For a triangle $\mathcal{T}$ in $\mathcal{G}$ of zero weight, the probability that all nodes in $\mathcal{T}$ are good is at least $\frac{1}{1+\epsilon}$, where $\epsilon = \frac{N}{M-9}\left(\frac{4}{M} + \frac{33N}{(N-2)M^2} + \frac{39}{M^2}\right)$.

Procedure $\mathsf{GetTriangles}(\mathcal{G}, X_1, C_1, \ldots, X_p, C_p)$

$(V, E) \leftarrow \mathcal{G}$; $L \leftarrow \emptyset$; Initialize hash tables H_0, H_1, H_2

For every edge $(u, v) \in E$ do

  $d \leftarrow \mathsf{Label}(u) \boxplus \mathsf{Label}(v)$; $P \leftarrow H_1[v, d]$; $P \leftarrow P \cup \{e\}$

  $s \leftarrow 0 \boxminus \mathsf{Label}(v)$; $S \leftarrow H_2[u, s]$; $S \leftarrow S \cup \{e\}$; $H_0[e] \leftarrow 1$

For every key (v, d) of H_1, every $u \in H_1[v, d], w \in H_2[v, d]$ do

  If $H_0[(w, u)] = 1$ then $\mathcal{T} \leftarrow (u, v, w)$; $L \leftarrow L \cup \{\mathcal{T}\}$

Return L

Fig. 10. Code to enumerate triangles of zero weight from the differential graph $\mathcal{G} = (V, E)$ of $X_1, C_1, \ldots, X_p, C_p$.

(b) The expected number of triangles in $\mathcal{G}$ whose all nodes are good is at least $\frac{p!}{3(p-6)!} \cdot \frac{1}{M^3N^6}\left(1 - \frac{7}{N} - \frac{14}{M}\right)$.

Discussion. While the core idea of differential graphs is from DV's work, there are important differences between our definition and DV's:

- Because of the symmetry of Feistel, one can view $\mathsf{Rev}(X_1), \ldots, \mathsf{Rev}(X_p)$ as the "ciphertexts" of $\mathsf{Rev}(C_1), , \ldots, \mathsf{Rev}(C_p)$ under a four-round Feistel $\overline{\mathsf{F}} = \mathbf{Feistel}[4, N, M, \boxminus]$ where $\mathsf{Rev}(X) = (R, L)$ for any $X = (L, R) \in \mathbb{Z}_M \times \mathbb{Z}_N$. In this sense, DV's notion is the dual of ours. While the two definitions yield no difference in the balanced setting, if $M \gg N$, DV's notion would give a much poorer bound[10] in a dual version of Lemma 9, leading to an inferior recovery rate of the TF attack.
- Our notion also adds some non-degeneracy requirements, allowing us to prove DV's conjecture on differential graph in Lemma 9 further below.

Enumerating zero-weight triangles. From Lemma 9, a simple way to realize good nodes is to enumerate all triangles of zero weight. We now show how to do that in $O(M^{5/3})$ expected time; the code is given in Fig. 10.

- First, for each node $v \in V$, partition its set of incoming edges such that each edge (u, v) goes to group $P_{v,d}$, with $d = \mathsf{Label}(u) \boxplus \mathsf{Label}(v)$, and also partition the set of outgoing edges into groups such that each edge (v, w) goes to group $S_{v,s}$, where $s = 0 \boxminus \mathsf{Label}(w)$.
- Next for each (v, d), enumerate all pairs $(u, w) \in P_{v,d} \times S_{v,d}$ such that there is a directed edge $(w, u) \in E$. Each triple (v, u, w) is a triangle of zero weight.

[10] In fact, the dual version of Lemma 9 would yield the bound $\frac{1}{1+\epsilon^*}$, where $\epsilon^* = \frac{M}{N-9}\left(\frac{4}{N} + \frac{33M}{(M-2)N^2} + \frac{39}{N^2}\right)$. Concretely, for $M = 100$ and $N = 10$, the bound in our Lemma 9 is 0.9947, whereas its dual is much poorer, just 0.0089.

Using appropriate data structure, the first step takes $O(|V|+|E|)$ time, whereas the cost of the second step is in the order of

$$\begin{aligned}\sum_{v\in V}\sum_{d\in\mathbb{Z}_N}|P_{v,d}|\cdot(1+|S_{v,d}|) &\le \Big(\sum_{v\in V}\sum_{d\in\mathbb{Z}_N}|P_{v,d}|\Big)+\Big(\sum_{v\in V}\sum_{d\in\mathbb{Z}_N}|P_{v,d}|\cdot|S_{v,d}|\Big)\\ &= \Big(\sum_{v\in V}\mathsf{indeg}(v)\Big)+\Big(\sum_{v\in V}\sum_{d\in\mathbb{Z}_N}|P_{v,d}|\cdot|S_{v,d}|\Big)\\ &= |E|+\Big(\sum_{v\in V}\sum_{d\in\mathbb{Z}_N}|P_{v,d}|\cdot|S_{v,d}|\Big)\ ,\end{aligned}$$

where $\mathsf{indeg}(v)$ is the incoming degree of node v. Since $\mathbf{E}[|V|]\in O(M^{4/3})$ and $\mathbf{E}[|E|]\in O(M^{5/3})$, what remains is to show that

$$\mathbf{E}\Big[\sum_{v\in V}\sum_{d\in\mathbb{Z}_N}|P_{v,d}|\cdot|S_{v,d}|\Big]\in O(M^{5/3})\ . \tag{2}$$

For each tuple $\mathcal{L}=(i,j,k,\ell,r,s,d)\in(\{1,\ldots,p\})^6\times\mathbb{Z}_N$ such that i,j,k,ℓ,r,s are distinct, let $B_{\mathcal{L}}$ denote the Bernoulli random variable such that $B_{\mathcal{L}}=1$ if and only if $(v_{i,j},v_{k,\ell})$ and $(v_{r,s},v_{i,j})$ are edges of $\mathcal{G}$, and $v_{k,\ell}\in S_{v_{i,j},d}$ and $v_{r,s}\in P_{v_{i,j},d}$. Then

$$\mathbf{E}\Big[\sum_{v\in V}\sum_{d\in\mathbb{Z}_N}|P_{v,d}|\cdot|S_{v,d}|\Big]=\mathbf{E}\Big[\sum_{\mathcal{L}}B_{\mathcal{L}}\Big]=\sum_{\mathcal{L}}\mathbf{E}[B_{\mathcal{L}}]=\sum_{\mathcal{L}}\Pr[B_{\mathcal{L}}=1]\ .$$

Note that for each $\mathcal{L}=(i,j,k,\ell,r,s,d)$, the event $B_{\mathcal{L}}=1$ happens only if the following events happen: (1) $v_{i,j}\in V$, (2) $v_{j,k}\in V$, (3) $v_{r,s}\in V$, (4) $\mathsf{LH}(C_j)=\mathsf{LH}(C_k)$, (5) $\mathsf{LH}(C_s)=\mathsf{LH}(C_i)$, (6) $\mathsf{Label}(v_{i,j})\boxplus\mathsf{Label}(v_{r,s})=d$, and (7) $\mathsf{Label}(v_{k,\ell})=0\boxminus d$. By pretending that these seven events are independent, from Lemma 3, we can heuristically estimate

$$\Pr[B_{\mathcal{L}}=1]\lessapprox\Big(\frac{M+N-1}{MN}\cdot\frac{1}{N}\Big)^3\Big(\frac{1}{M}\Big)^2\Big(\frac{1}{N}\Big)^2\le\frac{(M+N)^3}{M^5N^8}\le\frac{8}{M^2N^8}\ .$$

Hence

$$\sum_{\mathcal{L}}\mathbf{E}[B_{\mathcal{L}}]\lessapprox\frac{p!\cdot N}{(p-6)!}\cdot\frac{8}{M^2N^8}\in O(M^2/N)\ .$$

Moreover, due to our assumption that $N=\Omega(\sqrt{M})$, it follows that $M^2/N\in O(M^{1.5})$. We then conclude that

$$\mathbf{E}\Big[\sum_{v\in V}\sum_{d\in\mathbb{Z}_N}|P_{v,d}|\cdot|S_{v,d}|\Big]\in O(M^{1.5})$$

and thus justify Equation (2).

Remarks. DV also considered the problem of finding zero-weight triangles for the balanced case $M=N$. They first enumerated all triangles via sparse matrix multiplications, and then computed the sum of labels for each triangle. In this balanced setting, DV's algorithm takes $O(N^3)$ time, whereas our algorithm takes $O(N^{5/3})$ time.

4.2 The TF Attack

We begin with a simple but useful observation of DV on four-round Feistel.

AN OBSERVATION. Let (F_1, F_2, F_3, F_4) be the round functions of F. For any $\Delta \in \mathbb{Z}_N$, let $\mathsf{Shift}(\mathsf{F}, \Delta)$ denote a 4-round Feistel network $\overline{\mathsf{F}} = \mathbf{Feistel}[4, M, N, \boxplus]$ of round functions $(\overline{F}_1, \overline{F}_2, \overline{F}_3, \overline{F}_4)$ such that $\overline{F}_1 = F_1$, $\overline{F}_2(K, x) = F_2(K, x) \boxminus \Delta$, $\overline{F}_3(K, y) = F_3(K, y \boxplus \Delta)$, and $\overline{F}_4(K, x) = F_4(K, x) \boxplus \Delta$, for any $x \in \mathbb{Z}_M$, $y \in \mathbb{Z}_N$, and any key K. Note that for any choice of Δ, scheme $\overline{\mathsf{F}} = \mathsf{Shift}(\mathsf{F}, \Delta)$ ensures that $\overline{\mathsf{F}}.\mathsf{E}(K, X) = \mathsf{F}.\mathsf{E}(K, X)$ for any key K and any $X \in \mathbb{Z}_M \times \mathbb{Z}_N$. Therefore, in a codebook recovery attack against $\mathsf{F}.\mathsf{E}(K, \cdot)$, without loss of generality, one can pick a designated point $x^* \in \mathbb{Z}_M$ and *assume* that $F_4(K, x^*) = 0$.

THE ATTACK. Let (G_1, G_2, G_3, G_4) be the functions specified by the secret key of F. We will recover even the tables of those functions, instead of just the codebook.

Our attack TF is based on a known-plaintext codebook-recovery attack RY on three-round Feistel that we describe in the full version. If we run RY on $\ell \geq \max\{\lceil N(\ln(N)+\ln(2)+\lambda)\rceil, \lceil M(\ln(M)+\delta)\rceil\}$ known plaintext/ciphertext pairs, for any $\lambda, \delta > 0$, the RY attack will take $O(\ell)$ time, and recovers all the round functions of the three-round Feistel with probability around $e^{-e^{-\lambda}} - e^{-\delta}$. If we just have $\ell \geq \lceil N(\ln(N)+\ln(2)+\lambda)\rceil$ then RY will recover the top round function with probability at least $e^{-e^{-\lambda}}$. We note that RY is used in a modular way; one does not need to know the technical details of RY to understand the TF attack.

While TF somewhat resembles DV's attack on 4-round Feistel, there are important changes to improve efficiency and recovery rate, which we will elaborate further below. The code of TF is given in Fig. 11; below we will describe the attack.

In the TF attack, we will first construct the differential graph $\mathcal{G} = (V, E)$ of the plaintext/ciphertext pairs $(X_1, C_1), \ldots, (X_p, C_p)$, and then enumerate all triangles of $\mathcal{G}$ of zero weight. Let S be the set of the nodes of those triangles; each node S is very likely to be good, due to Lemma 9. For each $v_{i,j} \in S$, from Lemma 6, if $v_{i,j}$ is indeed good then $\mathsf{Label}(v_{i,j}) = G_4(\mathsf{LH}(C_i)) \boxminus G_4(\mathsf{LH}(C_j))$.

Our first step is to recover several (but possibly not all) entries of G_4. In order to do that, construct the following undirected graph $\mathcal{G}^* = (V^*, E^*)$ of $|V^*| = M$ nodes. Nodes in V^* are distinctly labeled by elements of $\mathbb{Z}_M$. For each node $v_{i,j} \in S$, we create an edge between nodes $\mathsf{LH}(C_i)$ and $\mathsf{LH}(C_j)$ of V^*, indicating that we know the difference between $G_4(\mathsf{LH}(C_i))$ and $G_4(\mathsf{LH}(C_j))$. Once the graph $\mathcal{G}^*$ is constructed, we pick an arbitrary node $x^* \in V^*$ that belongs to the biggest connected component of $\mathcal{G}^*$, and set $G_4(x^*) \leftarrow 0$. We then recover $G_4(u)$ for every node $u \in V^*$ reachable from x^* using breadth-first search (BFS), but stop when $\left\lfloor \frac{3M}{\sqrt{N}} \right\rfloor$ entries of G_4 are recovered. Let $\mathcal{I} \subseteq \{1, 2, \ldots, p\}$ be the set of indices i such that $G_4(\mathsf{LH}(C_i))$ is recovered at this point.

Our next step is to recover the entire table of G_1 using RY. For each $i \in \mathcal{I}$, recover the round-3 intermediate output Y_i of X_i via $\mathsf{LH}(Y_i) = \mathsf{LH}(C_i)$ and

Procedure $\mathsf{TF}(X_1, C_1, \ldots, X_p, C_p)$

$\mathcal{G} \leftarrow \mathsf{BuildGraph}(X_1, C_1, \ldots, X_p, C_p)$ // Build the differential graph
$L \leftarrow \mathsf{GetTriangles}(\mathcal{G}, X_1, C_1, \ldots, X_p, C_p)$ // Enumerate zero-weight triangles
$V^*, E^*, S \leftarrow \emptyset$; Initialize a hash table H
For every $\mathcal{T} \in L$ do $(u, v, w) \leftarrow \mathcal{T}$; $S \leftarrow S \cup \{u, v, w\}$
For $i \in \mathbb{Z}_M$ do $V^* \leftarrow V^* \cup \{i\}$
For every $v \in S$ do
  $(i, j) \leftarrow v$; $e \leftarrow \{\mathsf{LH}(C_i), \mathsf{LH}(C_j)\}$; $E^* \leftarrow E^* \cup \{e\}$
  $H[e] \leftarrow (\mathsf{LH}(C_i), \mathsf{Label}(v))$
$\mathcal{G}^* \leftarrow (V^*, E^*)$; Let $\mathcal{C}$ be the biggest connected component of $\mathcal{G}^*$
For $i \leftarrow 1$ to μ do
  $(G_1, G_2, G_3, G_4) \leftarrow \mathsf{Restore}(X_1, C_1, \ldots, X_p, C_p, \mathcal{C})$
  For $j \leftarrow 1$ to $3M$ do // Checking consistency of (G_1, G_2, G_3, G_4)
    $(L, R) \leftarrow X_j$
    For $k \leftarrow 1$ to 4 do
      If $(k \bmod 2 = 1)$ then $L \leftarrow L \boxplus G_k(R)$ else $R \leftarrow R \boxplus G_k(L)$
    $C_j^* \leftarrow (L, R)$
  If $(C_1, \ldots, C_{3M}) = (C_1^*, \ldots, C_{3M}^*)$ then return (G_1, G_2, G_3, G_4)

Fig. 11. The TF attack (parameterized by a small number μ) on 4-round Feistel, which is based on another attack RY on 3-round Feistel and a procedure Restore in Fig. 12.

$\mathsf{RH}(Y_i) = G_4(\mathsf{LH}(C_i)) \boxminus \mathsf{RH}(C_i)$. Then run RY on the pairs $\{(X_i, Y_i) \mid i \in \mathcal{I}\}$ to recover G_1, and then recover the round-1 intermediate outputs $Z_1, \ldots, Z_p$ of $X_1, \ldots, X_p$. In addition, observe that $\mathsf{Rev}(C_i)$ is the ciphertext of $\mathsf{Rev}(Z_i)$ under a 3-round Feistel $\overline{\mathsf{F}} = \mathbf{Feistel}[3, N, M, \boxplus]$ of round functions G_2, G_3, G_3 (note that the roles of M and N are now reversed), where for $Z = (A, B) \in \mathbb{Z}_M \times \mathbb{Z}_N$, we write $\mathsf{Rev}(Z)$ to denote the pair $(B, A) \in \mathbb{Z}_N \times \mathbb{Z}_M$. We then run RY on $\mathsf{Rev}(Z_1), \ldots, \mathsf{Rev}(Z_p), \mathsf{Rev}(C_1), \ldots, \mathsf{Rev}(C_p)$ to recover G_2, G_3, G_4.

To amplify the recovery rate, instead of using just one random node x^*, we try μ independent choices of x^*; in our implementation, we pick $\mu = 10$.[11] As analyzed in Section 3.3, we can decide which node yields a correct output by evaluating $3M$ plaintexts on a Feistel network with the recovered round functions, and comparing them with the corresponding ciphertexts.

Analysis. We now analyze the advantage of the TF attack; the key ideas in our analysis are largely from DV's work. We however tighten some of their arguments to improve the bounds.

[11] We note that here $\mu = 10$ means that the attack will iterate up to 10 times, each time with an independent choice of the initial node x^*, until it succeeds in recovering the entire codebook. The expected number of the iterations is often smaller than 10. For example, with $M = 1000$ and $N = 100$, empirically the attack would succeed at the first iteration, and thus it only performs a single iteration.

Procedure $\mathsf{Restore}(X_1, C_1, \ldots, X_p, C_p, \mathcal{C})$

Pick a node x^* uniformly at random from $\mathcal{C}$
Initialize tables G_1, G_2, G_3, G_4; $G_4(x^*) \leftarrow 0$; $count \leftarrow 0$
Run a breadth-first search on $\mathcal{C}$ from x^* and let T be the corresponding BFS tree
Let $(v_0, \ldots, v_t)$ be the visiting order of the nodes in the BFS above
For ($k = 1$ to t) and $count \le \lfloor 3M/\sqrt{N} \rfloor$ do
  $count \leftarrow count + 1$; Let u be the parent of v_k in T
  $e \leftarrow \{u, v_k\}$; $(w, R) \leftarrow H[e]$
  If $u = w$ then $G_4(v_k) \leftarrow G_4(u) \boxminus R$ else $G_4(v_k) \leftarrow G_4(u) \boxplus R$
$\mathcal{I} \leftarrow \{i \mid G_4(\mathsf{LH}(C_i)) \neq \perp\}$ // Consider the entire set $\{1, \ldots, p\}$ to find $\mathcal{I}$
// Recover the round-3 intermediate outputs for X_i with $i \in \mathcal{I}$
For $i \in \mathcal{I}$ do $Y_i \leftarrow \big(\mathsf{LH}(C_i), \mathsf{RH}(C_i) \boxminus G_4(\mathsf{LH}(C_i))\big)$
Run RY on (X_i, Y_i) for $i \in \mathcal{I}$ to recover G_1 // Here RY attacks domain $\mathbb{Z}_M \times \mathbb{Z}_N$
For $i = 1$ to p do // Recover round-1 intermediate outputs for every X_i
  $(L_0, R_0) \leftarrow X_i$; $L_1 \leftarrow L_0 \boxplus G_1(R_0)$; $R_1 \leftarrow R_0$; $Z_i \leftarrow (L_1, R_1)$
For $i = 1$ to p do $Z'_i \leftarrow \mathsf{Rev}(Z_i)$; $C'_i \leftarrow \mathsf{Rev}(C_i)$
Run RY on (Z'_i, C'_i) to recover (G_2, G_3, G_4) // Now RY attacks domain $\mathbb{Z}_N \times \mathbb{Z}_M$
Return (G_1, G_2, G_3, G_4)

Fig. 12. Procedure $\mathsf{Restore}$ in the TF attack. Here for $Z = (A, B) \in \mathbb{Z}_M \times \mathbb{Z}_N$, we write $\mathsf{Rev}(Z)$ to denote the pair $(B, A) \in \mathbb{Z}_N \times \mathbb{Z}_M$.

(M, N)	$(2^7, 2^6)$	$(2^7, 2^7)$	$(2^8, 2^7)$	$(2^8, 2^8)$	$(2^9, 2^8)$	$(2^9, 2^9)$	$(10^2, 10^2)$	$(10^3, 10^2)$
$\lvert E^* \rvert$	217 ± 6	203 ± 6	270 ± 6	321 ± 6	802 ± 11	965 ± 11	156 ± 5	1576 ± 16

Table 3. Empirical estimation of $|E^*|$ over 100 trials. The first row indicates the values of M and N. The second row shows the 95% confidence interval of $|E^*|$.

▷ We begin by estimating $\mathbf{E}[|E^*|]$. A direct generalization of DV's analysis would yield $\mathbf{E}[|E^*|] \approx \frac{p^6}{M^3 N^6}$, which is rather loose. Consider the empirical estimation of $|E^*|$ in Table 3. For $M = N = 100$, the 95% confidence interval of $|E^*|$ is 156 ± 5, but the approximation above suggests that $\mathbf{E}[|E^*|] \approx 400$.

We now provide a tighter analysis. Let W be the number of triangles in $\mathcal{G}$ whose all three nodes are good. From part (a) of Lemma 9, when we enumerate zero-weight triangles in $\mathcal{G}$, most of them will have three good nodes. Thus those triangles will contribute approximately $3W$ distinct good nodes. Note that if $\mathcal{T} = (v_{i,j}, v_{k,\ell}, v_{r,s})$ is a triangle then $\mathcal{T}^* = (v_{j,i}, v_{s,r}, v_{\ell,k})$ is also a triangle, and $\mathsf{weight}(\mathcal{T}^*) = 0 \boxminus \mathsf{weight}(\mathcal{T})$. (See Fig. 9 for an illustration.) Hence if $\mathcal{T}$ is a zero-weight triangle then so is $\mathcal{T}^*$, but these two triangles will produce the same three edges for E^*. Taking into account this duplication, $\mathbf{E}[|E^*|] \approx \frac{3\mathbf{E}[W]}{2}$. From

part (b) of Lemma 9,

$$\mathbf{E}[W] \gtrapprox \frac{p^6}{3M^3N^6}\Big(1 - \frac{7}{N} - \frac{14}{M}\Big) \geq \frac{2p^6}{9M^3N^6}$$

for $M \geq N \geq 64$. Hence

$$\mathbf{E}[|E^*|] \gtrapprox \frac{p^6}{3M^3N^6} \gtrapprox \frac{4M}{3} \quad . \tag{3}$$

This lower bound is consistent with Table 3. For example, for $M = N = 100$, we estimate that $\mathbf{E}[|E^*|] \gtrapprox 133$, and recall that empirically, the 95% confidence interval of $|E^*|$ is 156 ± 5.

$\triangleright$ Next, following DV, we model the graph $\mathcal{G}^*$ as a random graph according to the Erdős-Rényi model, in which each of the $\binom{M}{2}$ possible edges will have probability ρ to appear in E^*, independent of other edges. To determine the parameter ρ, note that according to the model above, the expected number of edges in E^* is

$$\binom{M}{2}\rho = \frac{M(M-1)\rho}{2} \quad . \tag{4}$$

From Equation (3) and Equation (4), we have $M\rho = \frac{2\mathbf{E}[|E^*|}{M-1} \gtrapprox \frac{8M}{3}$. From the theory of random graph (see, for example, Chapter 2 of Durrett's book [17]), the graph $\mathcal{G}^*$ will almost surely contain a giant component of size about $(1-c)M$ or bigger, where $c \approx 0.0878$ is the unique solution of the equation $e^{\frac{8}{3}(t-1)} = t$ in the interval $(0, 1)$. In DV's attack, one recovers $G_4(u)$ for every node u in the giant component, but since S *may* contain a few bad nodes, some entries of G_4 that we recover might be incorrect. Instead, we only recover $G_4(u)$ for nodes u in a connected subgraph of the giant component of size $\left\lfloor \frac{3M}{\sqrt{N}} \right\rfloor$. Those nodes are produced by about $\left\lfloor \frac{M}{\sqrt{N}} \right\rfloor$ triangles of zero-weight, and thus from Lemma 9, the chance that the nodes of these triangles are good is at least $1 - \frac{\sqrt{N}}{(M-9)} \cdot \Big(4 + \frac{33N}{(N-2)M} + \frac{39}{M}\Big)$. On the other hand, expectedly, we obtain about

$$\frac{3p}{\sqrt{N}} \geq 3 \cdot 2^{1/3} M^{2/3} \sqrt{N} \geq N(\ln(N) + \ln(2) + 2.7)$$

pairs of plaintext/ciphertext for RY, where the second inequality is due to the fact that $M \geq N \geq 64$. Thus we can run RY to recover G_1 with probability at least $e^{-e^{-2.7}} > 0.935$. We then can run RY with

$$p \geq M(\ln(M) + 5) \geq \max\{N(\ln(N) + 5), M(\ln(M) + \ln(2) + 4.3)\}$$

inputs, and thus can recover (G_2, G_3, G_4) with probability at least $e^{-e^{-4.3}} - e^{-5} > 0.975$. Summing up, if we just try one node x^* then our recovery rate is at least $0.91 - \frac{\sqrt{N}}{(M-9)} \cdot \Big(4 + \frac{33N}{(N-2)M} + \frac{39}{M}\Big)$. Using μ independent choices of x^* can only

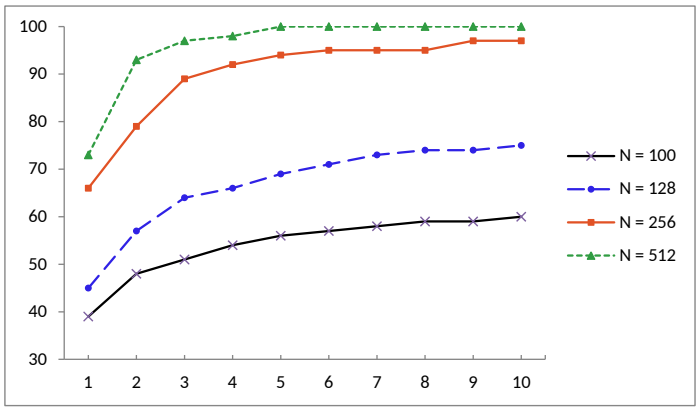

Fig. 13. The performance of TF with respect to μ, on balanced domains $\mathbb{Z}_N \times \mathbb{Z}_N$, over 100 trials. The x-axis indicates the values of μ, and the y-axis shows how many trials, out of 100 ones, that TF can recover the entire codebook.

improve the success probability. In fact, as illustrated in Fig. 13, empirically, the improvement when we increase μ from 1 to 10 is substantial.

▷ As mentioned in Section 4.1, constructing the differential graph $\mathcal{G}$ takes $O(M^{5/3})$ expected time, and so does enumerating zero-weight triangles. The graph $\mathcal{G}^*$ has M nodes, and expectedly, around $\frac{p^6}{3M^3N^6} \in O(M)$ edges. Thus identifying the connected components of $\mathcal{G}^*$ and doing BFS on its largest component takes $O(|V^*| + \mathbf{E}[|E^*|]) = O(M)$ expected time. Running RY takes $O(p)$ time. Hence the total running time is $O(M^{5/3})$.

Comparison with DV's attack. While our attack is inspired by DV's attack, there are important changes:

- First, as mentioned earlier, compared to DV's notion of differential graphs, we actually use a dual definition, for better recovery rate. Our notion also adds some non-degeneracy requirements, allowing us to find a proof for Lemma 9 and resolve DV's conjecture.
- Next, our attack has a much faster way to enumerate triangles of zero weight, reducing the running time from $O(N^3)$ to $O(N^{5/3})$ in the balanced setting.
- Recall that some zero-weight triangles may contain bad nodes, creating some noise in the attack. DV mentioned that their attack hardly succeeded for $2N^{5/3}$ or more plaintext/ciphertext pairs, and posed an open question to eliminate the noise. To resolve this issue, we introduce the trick of exploring the giant component from μ random places, and from each place, we stop after visiting $\lceil 3M/\sqrt{N} \rceil$ nodes.
- DV only run RY once to recover (G_1, G_2, G_3), and then derive the round-3 intermediate W_i values of X_i, and use (W_i, C_i) pairs to recover G_4. This works well for DV, as they consider just the balanced setting $M = N$. However, in unbalanced settings, for the first run of RY, we only have $3p/\sqrt{N} \ll M \ln(M)$ inputs. Thus the chance that one can recover (G_1, G_2, G_3) by using one RY

(M, N)	$(2^7, 2^6)$	$(2^7, 2^7)$	$(2^8, 2^7)$	$(2^8, 2^8)$	$(2^9, 2^8)$	$(2^9, 2^9)$	$(10^2, 10^2)$	$(10^3, 10^2)$
Real	100	100	100	100	100	100	100	100
Ideal	0	0	0	0	0	0	1	0

Table 4. Empirical performance of the LHD attack over 100 trials. The first row indicates the values of M and N. The second row shows how many times, over 100 trials, that LHD correctly outputs 1 in the real world. The last row shows how many times, again over 100 trials, that LHD incorrectly outputs 1 in the ideal world.

---

call is poor. We therefore only use the first RY call to get G_1, and run RY another time to recover (G_2, G_3, G_4).

5 Experiments

In this section, we empirically evaluate the LHD, SD, and TF attacks.

Benchmarking environment. We implemented our attack in C++ , and ran experiments using 72 threads in a server of dual Intel(R) Xeon(R) CPU E5-2699 v3 2.30GHz CPU and 256 GB RAM. We evaluate our attacks in both balanced and unbalanced settings, and for both binary and decimal domains. Specifically, we consider every $(M, N) \in \{(2^7, 2^6), (2^7, 2^7), (2^8, 2^7), (2^8, 2^8), (2^9, 2^8), (2^9, 2^9), (10^2, 10^2), (10^3, 10^2)\}$. For each choice of (M, N), we let

$$p = \max\left\{\lfloor 2^{1/3} M^{2/3} N \rfloor, \lceil M(\ln(M) + 5) \rceil\right\}$$

as specified in Equation (1).

Evaluating LHD. For each domain $\mathbb{Z}_M \times \mathbb{Z}_N$, we sample p messages uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$, and then extract $m = \lceil \frac{p}{N} \cdot \lceil 32N^{1/6} \rceil \rceil$ t-wise right-matching plaintexts, with $t = \lceil \frac{mM}{p} \rceil$. In the real world, we encrypt the plaintexts using the 4-round version of FF3 with the all-zero tweak to produce m ciphertexts. In contrast, the ciphertexts are chosen uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$ in the ideal world. The results of our experiments, given in Table 4, show that LHD is nearly perfect, which is much better than our theoretical estimation in Section 3.2. This is not surprising, since our analysis is very conservative.

Evaluating TF. For each domain $\mathbb{Z}_M \times \mathbb{Z}_N$, we sample p messages uniformly without replacement from $\mathbb{Z}_M \times \mathbb{Z}_N$, and generate p ciphertexts using 4-round FF3 with the all-zero tweak. We consider all choices of μ from 1 to 10. The results of our experiments, given in Table 5, are consistent with the theory. For example, with $M = N = 128$ and $\mu = 1$, the attack is supposed to recover the entire codebook with probability around

$$0.91 - \frac{\sqrt{N}}{M - 9} \cdot \left(4 + \frac{33N}{(N-2)M} + \frac{39}{M}\right) \approx 47.6\%$$

(M, N)	$(2^7, 2^6)$	$(2^7, 2^7)$	$(2^8, 2^7)$	$(2^8, 2^8)$	$(2^9, 2^8)$	$(2^9, 2^9)$	$(10^2, 10^2)$	$(10^3, 10^2)$
Recover, $\mu = 1$	70	45	75	66	84	73	39	100
Recover, $\mu = 2$	77	57	88	79	93	93	48	100
Recover, $\mu = 3$	81	64	91	89	95	97	51	100
Recover, $\mu = 4$	84	66	94	92	99	98	54	100
Recover, $\mu = 5$	85	69	95	94	100	100	56	100
Recover, $\mu = 6$	86	71	96	95	100	100	57	100
Recover, $\mu = 7$	87	73	97	96	100	100	58	100
Recover, $\mu = 8$	87	74	97	97	100	100	59	100
Recover, $\mu = 9$	88	74	98	97	100	100	59	100
Recover, $\mu = 10$	88	75	98	97	100	100	60	100

Table 5. Empirical performance of the TF attack over 100 trials. The first row indicates the values of M and N. Each subsequent row shows how many trials, over 100 ones, that TF correctly recovers the entire codebook, for the given choice of μ.

(M, N)	$(2^7, 2^6)$	$(2^7, 2^7)$	$(2^8, 2^7)$	$(2^8, 2^8)$	$(2^9, 2^8)$	$(2^9, 2^9)$	$(10^2, 10^2)$	$(10^3, 10^2)$
Have slid pairs	61	65	53	53	38	33	75	27
Survive LHD tests	61	65	53	53	38	33	75	27
Recover	50	39	51	50	38	33	22	27

Table 6. Empirical performance of the SD attack over 100 trials. The first row indicates the values of M and N. The second row shows how many trials, over 100 ones, have at least one slid pair, the third row shows how many of them survive the LHD tests, and the last row shows how many of them can successfully recover the entire codebook.

(M, N)	$(2^7, 2^6)$	$(2^7, 2^7)$	$(2^8, 2^7)$	$(10^2, 10^2)$
No. of candidates	74	72	53	89

Table 7. The total number of survived (possibly false) candidates after using LHD tests in SD, over 100 trials. The first row indicates the values of M and N. The second row shows the total number of survived (possibly false) candidates after using LHD tests in SD, over 100 trials.

---

and in the experiments, 45 out of 100 trials yield the correct codebook. Increasing μ will improve the performance substantially. For example, with $\mu = 10$, the recovery rate goes up to 75%.

Evaluating SD. To save time in evaluating SD, we use the FF3 key to find true candidates and run the LHD tests and the TF attack on them. Table 6 reports the empirical performance of SD over 100 trials, in which we use $\mu = 10$ for the underlying TF attack. The recovery rate is reasonable, ranging from 22% to 51%, and we never miss any true candidate using LHD tests. In addition, we also run the full SD attack on $(M, N) \in \{(2^7, 2^6), (2^7, 2^7), (2^8, 2^7), (10^2, 10^2)\}$ to evaluate the performance of LHD test on false slid-pair candidates. As shown in Table 7, our test is a nearly perfect filtering, leaving on average a single (possibly false) slid-pair candidate in each trial.

Acknowledgments

We thank anonymous reviewers of EUROCRYPT 2019 for insightful feedback. Viet Tung Hoang was supported by NSF grants CICI-1738912 and CRII-1755539. Ni Trieu was supported by NSF award #1617197.

References

1. W. Aiello and R. Venkatesan. Foiling birthday attacks in length-doubling transformations - Benes: A non-reversible alternative to Feistel. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 307–320. Springer, Heidelberg, May 1996.
2. A. Bar-On, E. Biham, O. Dunkelman, and N. Keller. Efficient slide attacks. *Journal of Cryptology*, 31(3):641–670, July 2018.
3. M. Bellare and V. T. Hoang. Identity-based Format-Preserving Encryption. In *CCS 2017*, pages 1515–1532, 2017.
4. M. Bellare, V. T. Hoang, and S. Tessaro. Message-recovery attacks on feistel-based format preserving encryption. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 444–455. ACM Press, Oct. 2016.
5. M. Bellare, V. T. Hoang, and S. Tessaro. Message-recovery attacks on feistel-based format preserving encryption. In *CCS 2016*, 2016.
6. M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. In M. J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini, editors, *SAC 2009*, volume 5867 of *LNCS*, pages 295–312. Springer, Heidelberg, Aug. 2009.
7. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.
8. E. Biham, A. Biryukov, O. Dunkelman, E. Richardson, and A. Shamir. Initial observations on Skipjack: Cryptanalysis of Skipjack-3XOR (invited talk). In S. E. Tavares and H. Meijer, editors, *SAC 1998*, volume 1556 of *LNCS*, pages 362–376. Springer, Heidelberg, Aug. 1999.
9. A. Biryukov, G. Leurent, and L. Perrin. Cryptanalysis of feistel networks with secret round functions. In O. Dunkelman and L. Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 102–121. Springer, Heidelberg, Aug. 2016.
10. A. Biryukov and D. Wagner. Slide attacks. In L. R. Knudsen, editor, *FSE'99*, volume 1636 of *LNCS*, pages 245–259. Springer, Heidelberg, Mar. 1999.

11. A. Biryukov and D. Wagner. Advanced slide attacks. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 589–606. Springer, Heidelberg, May 2000.
12. J. Black and P. Rogaway. Ciphers with arbitrary finite domains. In B. Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 114–130. Springer, Heidelberg, Feb. 2002.
13. E. Brier, T. Peyrin, and J. Stern. BPS: a format-preserving encryption proposal. Submission to NIST, 2010. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf`.
14. S. Dara and S. Fluhrer. FNR: Arbitrary length small domain block cipher proposal. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 146–154. Springer, 2014.
15. F. B. Durak and S. Vaudenay. Breaking and repairing the FF3 format preserving encryption over small domain. In *CRYPTO 2017*, pages 679–707. Springer, 2017.
16. F. B. Durak and S. Vaudenay. Generic round-function-recovery attacks for feistel networks over small domains. In *Applied Cryptography and Network Security*, pages 440–458, 2018.
17. R. Durrett. *Random Graph Dynamics*. Cambridge University Press, 2008.
18. V. T. Hoang, B. Morris, and P. Rogaway. An enciphering scheme based on a card shuffle. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 1–13. Springer, Heidelberg, Aug. 2012.
19. V. T. Hoang and P. Rogaway. On generalized Feistel networks. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 613–630. Springer, Heidelberg, Aug. 2010.
20. V. T. Hoang, S. Tessaro, and N. Trieu. The curse of small domains: New attacks on format-preserving encryption. In *CRYPTO 2018*, pages 221–251. Springer, 2018.
21. U. Mattsson. Format controlling encryption using datatype preserving encryption. Cryptology ePrint Archive, Report 2009/257, 2009. `http://eprint.iacr.org/2009/257`.
22. B. Morris and P. Rogaway. Sometimes-recurse shuffle - almost-random permutations in logarithmic expected time. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 311–326. Springer, Heidelberg, May 2014.
23. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
24. J. Patarin. New results on pseudorandom permutation generators based on the DES scheme. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 301–312. Springer, Heidelberg, Aug. 1992.
25. J. Patarin. Generic attacks on Feistel schemes. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 222–238. Springer, Heidelberg, Dec. 2001.
26. T. Ristenpart and S. Yilek. The mix-and-cut shuffle: Small-domain encryption secure against N queries. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 392–409. Springer, Heidelberg, Aug. 2013.
27. A. Saltykov. The number of components in a random bipartite graph. *Discrete Mathematics and Applications*, 5(6):515–524, 1995.
28. J. Vance and M. Bellare. Delegatable Feistel-based Format Preserving Encryption mode. Submission to NIST, Nov 2015.