Three Third Generation Attacks on the Format Preserving Encryption Scheme FF3

Ohad Amon[1⋆], Orr Dunkelman[2,⋆⋆], Nathan Keller[3,⋆⋆⋆], Eyal Ronen[1,†], and Adi Shamir[4]

[1] Computer Science Department, Tel Aviv University, Israel
{ohad.amon, eyal.ronen}@cs.tau.ac.il
[2] Computer Science Department, University of Haifa, Israel
orrd@cs.haifa.ac.il
[3] Department of Mathematics, Bar-Ilan University, Israel
nkeller@math.biu.ac.il
[4] Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Israel
adi.shamir@weizmann.ac.il

Abstract. Format-Preserving Encryption (FPE) schemes accept plaintexts from any finite set of values (such as social security numbers or birth dates) and produce ciphertexts that belong to the same set. They are extremely useful in practice since they make it possible to encrypt existing databases or communication packets without changing their format. Due to industry demand, NIST had standardized in 2016 two such encryption schemes called FF1 and FF3. They immediately attracted considerable cryptanalytic attention with decreasing attack complexities. The best currently known attack on the Feistel construction FF3 has data and memory complexity of $O(N^{11/6})$ and time complexity of $O(N^{17/6})$, where the input belongs to a domain of size $N \times N$.

In this paper, we present and experimentally verify three improved attacks on FF3. Our best attack achieves the tradeoff curve $D = M = \tilde{O}(N^{2-t})$, $T = \tilde{O}(N^{2+t})$ for all $t \leq 0.5$. In particular, we can reduce the data and memory complexities to the more practical $\tilde{O}(N^{1.5})$, and at the same time, reduce the time complexity to $\tilde{O}(N^{2.5})$.

We also identify another attack vector against FPE schemes, the *related-domain* attack. We show how one can mount powerful attacks when the adversary is given access to the encryption under the same key in different domains, and show how to apply it to efficiently distinguish FF3 and FF3-1 instances.

⋆ The first author is supported in part by Len Blavatnik and the Blavatnik Family foundation and by the Blavatnik ICRC.

⋆⋆ The second author was supported in part by the Center for Cyber, Law, and Policy in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office and by the Israeli Science Foundation through grants No. 880/18 and 3380/19.

⋆⋆⋆ The third author was supported by the European Research Council under the ERC starting grant agreement n. 757731 (LightCrypt) and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

† The fourth author is a member of CPIIS.

1 Introduction

Standard block ciphers such as DES [19] and AES [11] are designed to encrypt and decrypt fixed length binary strings. However, there are many cases in which the data we want to encrypt has a different format such as a decimal number (e.g., a social security number) or a string of English letters (e.g., a name). While we can try to map such inputs to binary strings, we are usually faced with the problem that the number of possible inputs is not a perfect power of 2. In these cases, the size of the encrypted values will be larger than the size of the original values. This can pose a severe problem when we try to protect existing databases or communication packets which have a fixed format and whose fields cannot be expanded even by a single bit, since we will not be able to simply replace each original value by its encrypted version.

A solution to the problem was proposed 23 years ago by Brightwell and Smith who introduced the concept of *Format-Preserving Encryption (FPE)* [10]. More precisely, FPE is a cipher that encrypts any predefined domain into itself, even when it is not represented as a fixed length binary string. For example, we want that the encryption of a credit card number to look like another credit card number, following the same syntactic restrictions on its format. FPE has been used and deployed by numerous companies, e.g., Voltage, Veriphone, Ingenico, Cisco, as well as by major credit-card payment organizations.

In the last 20 years, numerous FPE schemes were proposed. The first cipher to support the FPE functionality was the AES candidate Hasty Pudding Cipher [22] which was submitted by Schroeppel and Orman. In 2002, Black and Rogaway [8] proposed three different methods for offering FPE functionality: Cycle walking, prefix cipher, and a Feistel-based construction, where in cycle walking schemes we iteratively encrypt the plaintext under the secret key, until a ciphertext that resides in the domain is found. In 2008, Spies proposed the Feistel Finite Set Encryption Mode (FFSEM) [24], which is an AES based balanced Feistel network that uses the idea of cycle walking. This has become the underlying approach for many FPE schemes.

In the subsequent years, several groups submitted to the US National Institute of Standards and Technology (NIST) proposals for FPE schemes: Bellare et al. proposed FFX [2,3] (called by NIST "FF1"), Vance [25] proposed VAES3 (called by NIST "FF2") and Brier, Peyrin and Stern [9] proposed BPS (whose central component was called by NIST "FF3"). All these proposals are block ciphers, based on types of a Feistel network.

In 2016, NIST published a special publication (SP800-38G [13]) that specified the aforementioned FF1 and FF3 as two modes of operation for format-preserving encryption. The domain in these schemes consists of $M \times N$ possible inputs, but for the sake of simplicity we assume that $M = N$ in all our complexity estimates.

The first analysis of FF3 was published shortly afterwards by Bellare et al. [1] who developed an efficient message recovery attacks for small domains. A year later, Durak and Vaudenay [12] presented at Crypto'17 a new slide attack [7] against the FF3 scheme. The attack makes it possible to compute new cipher-

texts, but without finding the scheme's 128-bit cryptographic key (note that in FPE's the number of possible keys is typically much larger than the number of possible plaintexts). Its data complexity of $O(N^{11/6})$ is slightly smaller than the N^2 size of the codebook, and its time complexity is $O(N^5)$, regardless of the schemes's key size. The attack is based on the fact that the tweak-key schedule allows for a simple related-tweak attack that reduces the number of rounds we have to attack from 8 to only 4 rounds.

Following this attack, NIST had revised their recommendation by modifying the way the tweak is used in the scheme, calling the new scheme FF3-1 (see SP 800-38G Rev. 1 [14]). Despite this revision, the security of the original FF3 against slide attacks continued to stir a great deal of interest. In particular, at Eurocrypt 2019, Hoang, Miller and Trieu [17] presented a second generation slide attack which improved the first generation attack of Durak and Vaudenay by using better algorithms for detecting slid pairs. The resulting attack has the same data complexity of $O(N^{11/6})$ but a greatly reduced time complexity of $O(N^{17/6})$.

1.1 Our contributions

In this paper we present three third generation slide attacks on FF3:

1. A *symmetric slide attack* that follows the general strategy of Hoang et al.'s attack [17] but simultaneously improves all its complexity measures – from $D = M = N^{11/6}$ and $T = N^{17/6}$ to $D = M = \tilde{O}(N^{7/4})$ and $T = \tilde{O}(N^{5/2})$. It can be generalized to any point along the time/data tradeoff curve $D = M = \tilde{O}(N^{7/4-t})$ and $T = \tilde{O}(N^{5/2+2t})$, for any $0 \leq t \leq 1/4$.
2. A new type of *asymmetric slide attack* which exploits the asymmetry of the classical distinguisher on 4-round Feistel schemes to reduce the complexity even further – to $D = M = \tilde{O}(N^{3/2})$ and $T = \tilde{O}(N^{5/2})$, and more generally, to the tradeoff curve $D = M = \tilde{O}(N^{2-t})$ and $T = \tilde{O}(N^{2+t})$, for all $0 \leq t \leq 1/2$ (including the point $D = M = T = \tilde{O}(N^2)$). The reduction in data complexity is especially important, since it pushes the amount of required data significantly farther from the entire codebook ($\tilde{O}(N^{3/2})$ instead of $O(N^{11/6})$, out of N^2), while keeping the time complexity at $\tilde{O}(N^{5/2})$ – lower than the complexity of Hoang et al.'s attack.
3. A *slide attack using the cycle structure* which matches the second attack at the lowest overall complexity point – $D = M = T = N^2$. This attack is particularly interesting since it is the first practical application of the *slide attack using the cycle structure* technique [4], which was previously believed to be purely academic due to its huge data complexity, but can be applied in the context of FPE schemes due to their small input domains. Its successful application demonstrates the importance of developing new "theoretical" attack techniques which are often criticized for having hopelessly high complexities, since they may suddenly become practical in a different setting.

Our new attacks also utilize an improved PRF reconstruction phase. Durak and Vaudenay presented that the actual round functions can be reconstructed

given $\tilde{O}(N^{10/6})$ input/output pairs in time $O(N^3)$ [12]. The time complexity of the reconstruction attack was improved by Hoang, Miller, and Trieu to $O(N^{5/3})$. Both algorithms rely on finding cycles of length 3 in a graph (defined by the data). We show an improved cycles detection algorithm (based on meet in the middle approach), that allows finding longer cycles (in our case of length 4 and 5) while reducing the data complexity of this phase to $\tilde{O}(N^{3/2})$ as well as the time complexity to $\tilde{O}(N^{3/2})$.

A comparison of the complexities of our complete attacks with the complexities of previous attacks is presented in Table 1.

Attack & Source	Complexity		
	Data	Time	Memory
First Generation [12]	$O(N^{11/6})$	$O(N^5)$	$O(N^{11/6})$
Second Generation [17]	$O(N^{11/6})$	$O(N^{17/6})$	$O(N^{11/6})$
Symmetric Slide (Sect. 4.1)	$\tilde{O}(N^{7/4-t})$	$\tilde{O}(N^{5/2+2t})$	$\tilde{O}(N^{7/4-t})$
Cycle Detection Slide (Sect. 4.2)	N^2	$\tilde{O}(N^2)$	N^2
Asymmetric Slide (Sect. 4.3)	$\tilde{O}(N^{2-t})$	$\tilde{O}(N^{2+t})$	$\tilde{O}(N^{2-t})$

Table 1: Comparison of Complete Attacks on FF3

We experimentally verified all of our attacks and their complexity (source code is available at `https://github.com/OhadAm7/FF3-code`). Table 2 compares the concrete number of data queries required for our asymmetric slide attack and the second generation attack. We show that our attack outperforms the previous state-of-the-art in all parameters.

In the last part of the paper, we introduce a new class of distinguishing attacks that can only be applied to FPE schemes, which we call *related domain attacks*. We first show that if the cipher uses cycle walking *during the encryption process* of a block, then one can offer a simple key recovery attack. We then

	Asymmetric Slide (Sect. 4.3)			Second Generation [17])		
N	Number of Queries	Time Complexity	Success Rate	Number of Queries	Time Complexity	Success Rate
2^7	13752	2^{18}	0.58	16384	2^{20}	0.39
2^8	48302	2^{20}	0.69	52012	2^{23}	0.5
2^9	161676	2^{23}	0.69	165140	2^{26}	0.33

Table 2: A comparison of our Asymmetric Slide attack (with $t = 0.5$ and $L = 3$) and the previous Second Generation attack

show that it is possible to apply use this type of an attack to offer efficient and practical distinguishers on FF3 and FF3-1 using related-domain attacks. Finally, we identify a very simple design principal which can protect any FPE scheme from such attacks. This design principal was already used in various FPE schemes, e.g., FF1 [14].

1.2 Paper Organization

The paper is organized as follows: We describe FF3 in Section 2. The existing attacks against FF3 are summarized in Section 3. Our new attacks are given in Section 4. The experimental verification of these attacks is given in Section 5. We introduce the related-domain attack on cycle walking FPE schemes in Section 6, and discuss a specific set of distinguishing attacks for the case of FF3 and FF3-1 in Section 7. Finally, Section 8 concludes this paper.

2 FF3

FF3 is a Format Preserving Encryption based on the FFX methodology proposed by Brier, Peyrin, and Stern [9]. It is a Feistel construction which accepts a plaintext in a domain of size $N \times M$ and produces a ciphertext in that domain. The plaintext P is divided into two parts (which we refer to as halves even though they may have different sizes) L and R, each composed of u and v, respectively, characters over some alphabet. In each round one half enters a PRP (a full AES encryption) together with a tweak, the key, and a round constant (which is equal to the round number). The output is numerically added modulo the respective size to the other half, their roles are then swapped for the next round.[1]

Formally, the encryption algorithm takes a 64-bit tweak $T = T_L||T_R$, where T_L and T_R are 32-bit each. Then, an 8-round Feistel construction is used, as depicted in Algorithm 1. In each round, half of the data is encoded into 96 bits (padding it with 0's if needed) using the naive lexicographic transformation.[2]

The encoded value is appended to the XOR of the 32-bit tweak and the round constant. The resulting 128-bit string is then encrypted under AES with the key K. The AES' output is then added using modular addition to the other half.

It is important to note that following the previous attacks of [12], a new version of FF3 called FF3-1 had been proposed in [14]. In FF3-1, the tweaks T_L and T_R are chosen such that they always have 0 in the 4 bits which accept the round counter i. This tweak destroys the related-tweak slide property which lies in the core of the slide distinguishers, and thus prevents the attack of [12],

[1] As the two halves may not be of equal size, following previous works that try to avoid possible confusion, throughout this paper we avoid the swap after the round function.

[2] FF3 is defined for strings over some alphabet; it uses the transformation $Encode96(X)$ which computes the location of X in the lexicographic order of all the possible strings, and encodes this number as a 96-bit binary string.

Algorithm 1: The Encryption Algorithm of FF3

Input : Message P of domain of size $M \times N$, Key K, Tweak $T = T_L||T_R$
Output: Ciphertext C of domain size $M \times N$

$(L, R) \leftarrow P$;
for $i \leftarrow 0$ **to** 7 **do**
  if $i \mod 2 = 0$ **then**
    $L \leftarrow L \boxplus AES_K(Encode96(R)||T_R \oplus i) \mod M$;
  else
    $R \leftarrow R \boxplus AES_K(Encode96(L)||T_L \oplus i) \mod N$;
return $C \leftarrow L||R$;

as well as its extensions [17] and our results presented in Section 4. All these attacks are only applicable to the original FF3 scheme.

On the other hand, our results presented in Section 7 are independent of the tweak schedule. Hence, the related-domain distinguishing attack applies both to FF3 and to FF3-1.

2.1 Our Notations

Throughout the paper we use several notations related to FF3: We use the term plaintexts and ciphertexts to refer to the inputs and outputs of 8-round FF3. As our attacks are usually mounted on 4-round FF3, we use the term inputs and outputs to denote those.

In addition, a plaintext is $P = (L_0, R_0)$, where the values after the ith round are (L_i, R_i), i.e., the ciphertext are (L_8, R_8). We use $LH(\cdot)$ to denote the left half of a value, and similarly $RH(\cdot)$ to denote the right half of a value.

The notation $\binom{n}{2}$ is the binomial coefficients for n choose 2, which is the number of possible pairs in a group of size x.

3 Previous Attacks

We now describe the previously published attacks against FF3. We note that they exploit the relatively small size of the input domain, and do not attempt to recover the 128-bit cryptographic key of the AES function. Consequently, their complexity is stated as a function of the scheme's domain size (which is N^2 when $M = N$) rather than as a function of the key size.

3.1 A Message Recovery Attack [1]

The first work analyzing FF3 is by Bellare, Hoang, and Tessaro [1]. The proposed attack is a message recovery attack for small domain sizes. The attack takes $3 \cdot 24 \cdot (n + 4) \cdot 2^{6n}$ data to attack FF3 with $2n$-bit blocks (where each triplet is encrypted using a single tweak value). It is based on a simple differential

distinguisher — given an input difference $(x, 0)$ the output difference is also $(x, 0)$ with a slightly higher probability than the expected probability (which is $1/(2^{2n} - 1)$).

The differential characteristic is quite straightforward. Given an input difference $(x, 0)$, the first round maintains the $(x, 0)$ difference with probability 1. The second round has a non-zero input difference, but with probability 2^{-n} (or $1/M$ of M if not a power of 2) the round function which is a PRF has an output difference of 0. This is an iterative differential, which suggests that the plaintext difference $(x, 0)$ becomes the ciphertext difference $(x, 0)$ with probability $1/(2^{2n} - 1) + 2^{-4n}$ for the 8-round FF3.

The attack is given a plaintext $X' = (L', R)$ and tries to recover the plaintext $X = (L, R)$ for some unknown $L \neq L'$. This is done by asking for the encryption of (X, X') under many different tweaks (the adversary in this scenario does not know X but can still obtain the corresponding ciphertexts). The differential characteristic suggests that the difference of the left half of the ciphertexts is equal to the difference in the left half of the plaintexts. As the ciphertexts can be observed, the adversary can compute the ciphertext difference. Since the input X' is known to the adversary, then the value of the left half of X can be recovered.

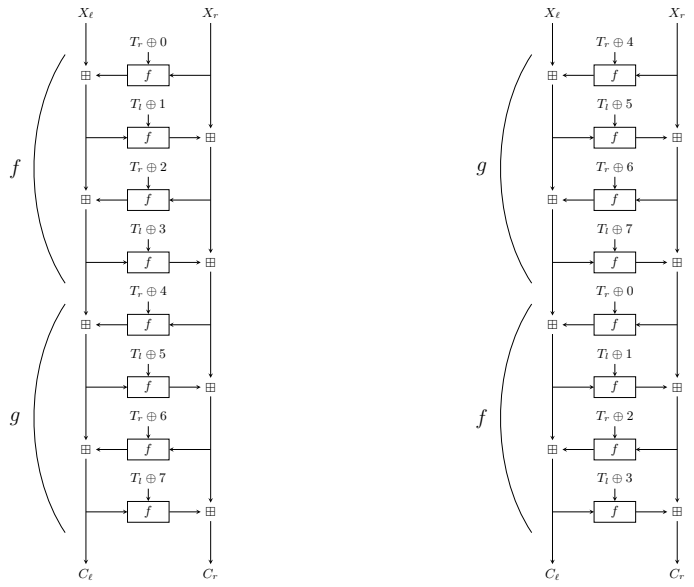
A similar idea can be used to recover the right hand side. The main difference is that only ciphertexts for which the left halves agree are used in the counting process (as the differential characteristic in use is based on the left hand side). The two attacks can be combined to recover an unknown X by probing its ciphertext together with two related plaintexts X' and X^* under many tweaks. Using the relation between X and X' one can recover its left half, and using the between X and X^* one can recover the right half.

3.2 A First Generation Related-Tweak Slide and PRF Recovery [12]

The original idea of the related-tweak slide attack was proposed at Crypto'17 by Durak and Vaudenay [12]. It can reconstruct the full table of $AES_K(x)$ for different inputs and for a given tweak, allowing the encryption/decryption of all plaintexts/ciphertexts with that tweak (and in some cases even under additional tweaks which are related to the original tweak).

The attack itself uses $O(N^{11/6})$ adaptive chosen plaintexts (for domains of size $N \times N$) which are encrypted under two tweaks: $T = T_L||T_R$ and $T' = T_L \oplus 4||T_R \oplus 4$. As seen in Figure 1, for the same key, if one can write the 8-round encryption under K with the tweak T as $g \circ f$ (each of 4 rounds), then the encryption under K with the tweak T' is equal to $f \circ g$.

As a result, if a plaintext P is partially encrypted under f (the first four rounds of the encryption under K and T) into P', then its corresponding ciphertext, C is equal to the evaluation of g on P'. This property continues (as C', the ciphertext corresponding to P' is the result of applying f to C), and allows constructing long slid chains, as suggested by Furuya [16]. For such a slid chain, the adversary is left with attacking a 4-round Feistel construction, for which



(a) Encryption using T (b) Encryption using T'

Fig. 1: Encryption under related-tweaks

Durak and Vaudenay present a known plaintext attack with $O(N^{10/6})$ data and $O(N^3)$ time.[3]

The attack algorithm, given in Algorithm 2 is as follows: First,[4] $N^{1/6}$ possible chains of $2N^{10/6}$ values are generated by picking a random x_0^i value, and iteratively encrypting it under K, T, i.e., $x_1^i = FF3_{K,T}(x_0^i)$, $x_2^i = FF3_{K,T}(x_1^i), \ldots$. Similarly, $N^{1/6}$ chains of $2N^{10/6}$ values are generated from random y_0^i values, iteratively encrypted under K, T', i.e., $y_1^i = FF3_{K,T'}(y_0^i)$, $y_2^i = FF3_{K,T'}(y_1^i), \ldots$.

Then, the attack tries each pair of starting points (x_0^i, y_t^j) (for all possible i, j, and $0 \leq t \leq N^{10/6}$) as if they constitute the beginning of a slid chains. If indeed x_0^i and y_s^t are slid pairs (which suggests that $f(x_0^i) = y_t^j$) then so are the rest of the chain (i.e., $f(x_s^i) = y_{t+s}^j$). Hence, the adversary obtains at least N pairs of values for the recovery attack. If the recovery attack succeeds, then the considered chains were indeed slid chains (not that it matters, as the recovery part succeeded). Similarly, the attack can be applied against $g(\cdot)$ with the corresponding changes.

[3] There are other reconstruction attacks against Feistel ciphers, such as [20] or [6], but these usually require a chosen plaintext attack scenario, whereas in this case, a known plaintext attack is needed.

[4] We alert the reader that [12, Sect. 5] suggests that $\sqrt{N}$ chains of length $2N$ values are needed. However, given that the function recovery attack needs $N^{10/6}$ known plaintexts, then we report, similarly to [17] the correct values.

Algorithm 2: The Basic Attack algorithm on FF3 by Durak and Vaudenay [12]

Pick at random $N^{1/6}$ values x_0^i. Pick at random $N^{1/6}$ values y_0^i. **for** *all* $1 \leq i \leq N^{1/6}$ **do**
  for $j = 0$ *to* $2N^{10/6} - 1$ **do**
    Compute $x_{j+1}^i = FF3_{K,T}(x_j^i)$ Compute $y_{j+1}^i = FF3_{K,T}(y_j^i)$
for *all* $1 \leq i \leq N^{1/6}$ **do**
  for *all* $1 \leq j \leq N^{1/6}$ **do**
    for *all* $0 \leq t \leq N^{10/6}$ **do**
      Assume that (x_0^i, y_t^j) generate slid chains.
      Call the Function Recovery attack on f with (x_0^i, y_t^j).

The function recovery attack is based on trying to recover the input/output values for 4-round Feistel (each with a different round function). Specifically, let the input/output of the 4-round FF3 be denoted by (L_0, R_0) and (L_3, R_4), respectively, then this input/output pair defines four input/output pairs to the corresponding round function. We follow previous work (and the description of [13]) and do not perform the swap after each Feistel round):

$$\begin{aligned} L_1 &= L_0 + F_0(R_0) \\ R_2 &= R_0 + F_1(L_1) \\ L_3 &= L_1 + F_2(R_2) \\ R_4 &= R_2 + F_3(L_3) \end{aligned}$$

where the F_i represent the keyed and tweaked round function.

The recovery attack starts with $N^{3/2+1/2L}$, for a parameter L set to 3, input/output pairs $((L_0^i, R_0^i), (L_3^i, R_4^i))$ with equal L_3, i.e., $L_3^i = L_3^j$ (for which there is no difference in the input or output of $F_3(\cdot)$) and with the right hand difference $R_4^i - R_4^j = R_0^i - R_0^j$. Furthermore, a set of *good pairs* is defined as pairs for which $L_1^i = L_1^j$. For these pairs

$$F_0(R_0^j) - F_0(R_0^i) = L_0^j - L_0^i \tag{1}$$

holds as well. In other words, for the good pairs, one obtains information about the outputs of $F_0(\cdot)$.[5]

Now, the attack tries to identify the good pairs using the following idea: Let the set of vertices be all the pairs for which $L_3^i = L_3^j$. A directed edge (i, j) is added to the graph if $L_3^i = L_0^j$ with the label $L_0^j - L_0^i$. The graph has cycles in

[5] We alert the reader that there are multiple solutions to the problem of recovering F_0, F_1, F_2, F_3. However, by fixing one value for F_0 (or any other F_i), the solution becomes unique.

it if the sum of labels on the edges is zero (as $\sum_{(i,j)\in cycle} L_0^j - L_0^i = 0$). If the cycle is composed only of good pairs, then we also obtain information about the outputs of F_0 (as the label on the edges that sum to zero is also the output of the round function F_0, following Equation 1).

Hence, the attack tries to find such cycles of length L. Each R_0 input that appears in such cycles can then be part of the reconstruction phase, and thus we need all of them to be covered (i.e., appear in the graph). Moreover, we need that any R_0 input value will be connected (possibly via different cycles) to any other R_0 input value (as Equation 1 is differential in nature). Once enough inputs to F_0 are present, one can assign one output F_0 arbitrarily (which defines all the other outputs). Once F_0 is (partially) recovered, the attack needs to recover F_1, F_2, F_3, which is a much simpler problem (which is solved either by Patarin's attack [21] and/or ideas very similar to the ones for the 4-round recovery attack). Hence, the adversary takes the largest connected component found in the attack, and runs the 3-round attack for the values that can be recovered (if the 3-round attack fails, then at least one of the values is wrong).

Given $O(N^{3/2+1/2L})$ known plaintexts, we expect $O(N^{3+1/L})$ pairs, out of which $O(N^{1+1/L})$ satisfy the differential conditions (zero difference in the left half of the ciphertext and the input of the right half of the plaintext equal to that of the ciphertext). Hence from any of $O(N)$ vertices, we expect about $O(N^{1/L})$ edges. In their analysis, Durak and Vaudenay show that a cycle of length $L = 3$ is sufficient. To detect these cycles, they just use the Floyd-Warshall algorithm [15] that takes $O(N^3)$ for $L = 3$.

Finally, Durak and Vaudenay noted that there is a non-trivial tradeoff between the number of vertices/edges in the graph and the success rate: If there are too few edges (i.e., too little pairs to begin with), then the chance that F_0 is recovered is small (as there are only small connected components). On the other hand, if there are too many edges, then besides the cycles of good pairs, we expect to find many cycles of bad pairs as well (which cause the failure of the recovery attack and waste time).

3.3 A Second Generation Related-Tweak Slide and PRF Recovery [17]

The attack of Hoang, Miller and Trieu [17] improves the attack of Durak and Vaudenay using two main ideas: The first idea is to improve the detection of slid chains. The second idea is an improved (and more suitable) cycles detection algorithm, which allows for better complexity.

The improved detection of slid pairs is done using the ideas presented in [4] of identifying the respective offset of a slide using a differential distinguisher (which were further developed in [5]). They rely on the existence of a bias in the probability of the differential characteristics $(x, 0) \to (x, 0)$, as for the correct shift between the chains, the number of pairs with input difference $(x, 0)$ having ciphertext difference $(x, 0)$ is higher than when the shift is wrong.

Hence, the slid chains are identified not by running an attack but rather by an auxiliary distinguisher. Instead of running the full recovery attack for each

possible slid chains and possible offsets, the attack is repeated fewer times (about $O(N)$ for the parameters considered in [17]).

The combination of the slide with the differential is as follows: Collect $O(N^{1/6})$ chains of length $O(N^{5/3})$ each under T and under T'. For each pair of candidate slid chains (and respective offset) check whether the differential distinguisher succeeds, namely, check whether the input difference $(x, 0)$ leads to the output difference $(x, ?)$ with probability of $(2N-1)/N^2$ which is about twice as high as for the random case.

The distinguisher accepts m candidate input/output pairs (the inputs encrypted under T and the outputs under T'). These m input values are then divided into d bins according to the value of R_0^i. In each bin, all the inputs have the same value in the right hand side, and thus, input difference of 0 in that half. We note that bins with many such values offer many pairs, and thus can be used for the next step of the attack. For each bin with many inputs, the distinguisher checks how many times the difference in the left half of the inputs is equal to the difference in the left half of the outputs. The threshold was chosen to be $1/5 \cdot \frac{2N-1}{N^2} + 4/5 \cdot \frac{N}{N^2-1}$ of the number of candidate pairs.

We note that this threshold was chosen so that the probability of right slid chains to fail is negligible (i.e., $O(1/\sqrt{N})$) and that chance for a random permutation to pass the distinguisher is also $O(1/\sqrt{N})$. The latter claim is obtained using Chebyshev's inequality that suggests that the probability that the counter is k standard deviations larger than the mean value is at most $1/k^2$. The standard deviation is then upper bounded using the Cauchy-Schwartz inequality based on the sizes of the different bins.

The chains in use are of length $O(N^{5/6})$ and as in Durak and Vaudenay's attack one needs to consider $O(N^2)$ possible pairs of chains and corresponding offsets. Moreover, for each such possible chains and offsets, one can apply the same distinguisher for the last four rounds of FF3 (i.e., treating the outputs as inputs to four round FF3). Hence, a wrong chain/offset is expected to pass the two distinguishers with probability of at most $O(1/N)$. The time complexity of this part is $O(N^{17/6})$ and it dominates the running time of the attack.

The second idea is to offer a better PRF reconstruction attack that runs in time $O(N^{5/3})$ instead of Durak and Vaudenay's original $O(N^3)$. As it targets cycles of length 3, the *Triangle-Finding* algorithm puts the input/output pairs in a hash table indexed by $L_3^i \oplus L_0^i || R_0^i$. Any collision in the table offers a pair of input/output pairs

$$((L_0^i, R_0^i), (L_3^i, R_4^i)), ((L_0^j, R_0^j), (L_3^j, R_4^j))$$

Each of them has an edge in the graph.

The attack then starts from an edge in the graph. This edge defines the two nodes which are connected. In the case of a triangle, the two nodes define the requirement from the third node (as the sum of the labels is 0). Hence, it is a simple matter to check whether there is such a third node in the data, i.e., whether the edge the attack starts from is indeed part of a good triangle.

As the attack is repeated $O(N)$ times, and takes $O(N^{5/3})$, this part of the attack takes $O(N^{8/3})$ time in total.

4 Improved Attacks on FF3

Similarly to Hoang et al.'s attack, our attack uses two subroutines: Identification of the correct slid chains and a PRF reconstruction phase. We offer three methods to identify the correct slid chains: The first method follows Hoang et al.'s approach which we call symmetric slide attack. Our improved version uses $\tilde{O}(N^{7/4})$ data and $\tilde{O}(N^{5/2})$ time, and is described in Section 4.1. We also extend this distinguisher with a time-memory tradeoff attack for which $\tilde{O}(N^{7/4-t})$ data is used with time of $\tilde{O}(N^{5/2+2t})$ for $t \in [0, 1/4]$. The second method, described in Section 4.2, uses a cyclic structure of slid pairs (as proposed in [4]), resulting in data and time complexities of $O(N^2)$. The third method uses an asymmetric slide attack, it also offers a time-data tradeoff with $\tilde{O}(N^{2-t})$ data and $\tilde{O}(N^{2+t})$ running time. Its memory complexity is $\tilde{O}(N^2)$, and is described in Section 4.3.

The PRF reconstruction, described in Section 4.4, is the same for all slid chain identification variants. Our PRF reconstruction procedure follows the same general idea suggested by [12,17], i.e., based on cycles. At the same time, we introduce a meet in the middle approach to the recovery itself, which significantly reduces its running time, thus allowing the use of larger cycles (which results in reducing the data, and hence, the time complexities).

4.1 Symmetric Slide Attack

In this attack, our data is composed of 2 sets of $\tilde{O}(N^{1/4})$ chains, each containing $\tilde{O}(N^{3/2})$ plaintexts. Similarly to [12,17], the first set of chains are encrypted under K and T and the second set is encrypted under K and T'.

We iterate over all $\tilde{O}(N^{1/2})$ pairs of chains created by taking a chain from each set. For each such pair of chains, we slid the first chain across the second one for $\tilde{O}(N^{3/2})$ different offsets. For each of the $\tilde{O}(N^2)$ resulting offsets, we utilize a distinguishing attack to checking whether the candidate slid chains (with offset) corresponds to 4 rounds of FF3 or not.

Actually, the distinguisher we use is very similar in nature to that of [17]. We rely on the fact that the truncated differential characteristic $(x, 0) \to (x, ?)$ for 4-round FF3 has probability of about $2/N$ rather than $1/N$ for the random case. Unlike [17] that divided the datasets between bins (according to the x value) and counted how many of them had "more pairs than expected in the random case", we argue that a single counter is sufficient (and more efficient). Namely, given m pairs with input difference $(x, 0)$ we expect $2m/N$ pairs with output difference $(x, ?)$ (compared with m/N for a random permutation).

The number of pairs that follow the truncated differential is distributed according to the Poisson distribution. Hence, $m = O(N \log(N)) = \tilde{O}(N)$ is sufficient to distinguish between the two distributions — one Poisson distribution with parameter $\lambda = m/N$ and another with parameter $\lambda = 2m/N$.

The above fact can also be explained by the following probabilistic explanation: Each pair with the required input difference has probability of about $2/N$ for 4-round FF3 or $1/N$ for a random permutation to have the required output difference. Hence, we can assign an indicator variable to whether a given

pair satisfies the differential. As all the indicators are independent (recall that 4-round Feistel is a PRP [18]) we can use the Chernoff bound: For the random permutation, the probability that the sum of indicators (which in our case corresponds to the number of pairs that satisfy the differential) is greater than $(1+\delta)m/N$ is no more than $(e^{\delta}/(1+\delta)^{1+\delta})^{m/N}$. For $m = c \cdot N \log N$ this bound is $(e^{\delta}/(1+\delta))^{c}$. Setting $\delta = 0.5$ this upper bound becomes $0.897^{c \cdot \log N}$. For example, taking $c > 5$ means that less than $1/\sqrt{N}$ of the sums of indicators for random permutations are greater than $1.5 \cdot 5 \cdot \log N$. The optimal threshold between the two distributions can be found either experimentally or by analyzing the Poisson distribution.

We note that similarly to [17], we can run the distinguisher twice: Once for the first 4 rounds, and another time for the second 4 rounds. Hence, the probability of a wrong slid chain to pass the distinguisher is less than $1/N$.

In contrast, for a 4-round FF3, the mean value for the sum of indicators is $2 \cdot c \log N$. Again, the number of right pairs is expected to be higher than $1.5 \cdot c \log N$ with high probability. This again can be achieved by a Chernoff analysis or by studying the Poisson distribution. However, as mentioned before, it is sufficient to set the threshold based on experiments (which confirm the Poisson distribution).

The attack follows the footsteps of [17], but with a significantly smaller number of pairs needed for the distinguisher as the statistical significance is larger. Hence, we start by taking $\tilde{O}(N^{1/4})$ chains of length $\tilde{O}(N^{3/2})$ each.

In each such chain, we insert all values (L_0^i, R_0^i) into a hash table according to the value of R_0^i. As there are $\tilde{O}(N^{3/2})$ values in the chain, we expect one of the bins to contain about $\tilde{O}(N^{1/2})$ values, which suggest $\tilde{O}(N)$ pairs, all with input difference $(x, 0)$. In practice, we need to take a constant number of bins.[6]

We take the actual values of L_0^i, and use them as the candidate inputs.

Then for each candidate chain (out of $\tilde{O}(N^{1/4})$ of them) and candidate offset (out of possible $\tilde{O}(N^{3/2})$ offsets) we extract the corresponding $\tilde{O}(N^{1/2})$ values which may serve as the candidate outputs for the above $\tilde{O}(N^{1/2})$ inputs, denoted by $(\hat{L}_3^i, \hat{R}_3^i)$. Then, for each bin, we store in a hash table the values $\hat{L}_3^i - L_0^i$, where each collision suggests a pair of inputs with difference $(x, 0)$ (the right-hand zero difference is guaranteed by the way the inputs were chosen) and the corresponding outputs have difference x in the left hand side. Hence, we can test in time $\tilde{O}(N^{1/2})$ whether two chains are slid chains in a given offset.

The resulting algorithm, given in Algorithm 3 takes $\tilde{O}(N^{3/2})$ data and $\tilde{O}(N^{2.5})$ time.

Offering a Time-Data Tradeoff We can offer a time-data tradeoff for the improved symmetric slide attack. The distinguisher takes $\tilde{O}(N^{7/4-t})$ data and has a running time of $\tilde{O}(N^{5/2+2t})$ for $t \in [0, 1/4]$.

The attack is based on taking shorter chains as in [17], but more of them. Given that the chains are shorter (of length $\tilde{O}(N^{3/2-2t})$) we need to collect plaintexts from N^{4t} bins to obtain enough pairs for the distinguisher. Then,

[6] Taking the 8 largest bins is empirically shown to suffice.

Algorithm 3: Improved Symmetric Slide Distinguisher for FF3

Input : $\tilde{O}(N^{1/4})$ chains C^r of $\tilde{O}(N^{3/2})$ plaintexts encrypted under K and $T = T_L||T_R$
Input : $\tilde{O}(N^{1/4})$ chains $\hat{C}^s$ of $\tilde{O}(N^{3/2})$ plaintexts encrypted under K and $T' = T_L \oplus 4||T_R \oplus 4$
Output: Slid chains $C^i, \hat{C}^j$ and their respective offset

for *all chains* C^r **do**
  Initialize a hash table H_1
  Insert all the plaintexts $(L_0^i, R_0^i) \in C^r$ into H_1 indexed by R_0^i
  Take a constant number d of bins (each with $O(\sqrt{N})$ plaintexts)
  Denote the plaintexts by $X_{i_1}, X_{i_2}, \ldots, X_{i_v}$
  for *all chains* $\hat{C}^s$ **do**
    for *all respective offsets* $u = 0, \ldots, N^{3/2}$ **do**
      Extract $(\hat{L}_0^{u+i_1}, \hat{R}_0^{u+i_1})$, $(\hat{L}_0^{u+i_2}, \hat{R}_0^{u+i_2})$,..., $(\hat{L}_0^{u+i_v}, \hat{R}_0^{u+i_v})$ from $\hat{C}^s$
      Denote these values as "ciphertexts" $C_{i_1}, C_{i_2}, \ldots, C_{i_v}$
      Initialize d hash tables H_2^j
      for *all k=1,...,v* **do**
        if X_{i_k} *is from bin* j **then**
          Store in H_2^j the value $LH(C_{i_k}) - LH(X_{i_k})$
      Count the number of collisions in all H_2^j
      if *number of collisions is greater than* $\frac{1.6}{N} \cdot \Sigma_{B \in bins} \binom{|B|}{2}$ **then**
        Call the PRF-recovery procedure with C^r as inputs and $\hat{C}^s$ shifted by u as the outputs.

when we process the second chain, we only consider a pair of outputs if they correspond to plaintexts from the same bin.

Repeating the above analysis shows that each step has to deal with shorter chains, but repeated more times. The result is indeed an attack whose data complexity is $\tilde{O}(N^{7/4-t})$ data and has a running time of $\tilde{O}(N^{5/2+2t})$ for $t \in [0, 1/4]$. The resulting algorithm is given in Algorithm 4.

The extreme case, with the minimal amount of data $\tilde{O}(N^{3/2})$, uses all the bins. The resulting attack uses $\tilde{O}(N^{1/2})$ chains of length $\tilde{O}(N)$. For each such chain, we insert all the plaintexts into a hash table indexed by the value of R_0^i, identify the $\tilde{O}(N)$ pairs (out of $\tilde{O}(N^2)$ possible ones) with input difference $(x, 0)$. Then, for any candidate chain counterpart (and any of the $\tilde{O}(N)$ possible offsets), we take the $\tilde{O}(N)$ corresponding values as ciphertexts, and check how many times the output differences are indeed x in the left half.

In other words, for each pair of candidate slid chains and offset, we just collect all the $\tilde{O}(N)$ pairs of inputs with difference $(x, 0)$ and test whether the corresponding outputs have difference x with the bias predicted for 4-round FF3. Identifying the pairs can be done in time $\tilde{O}(N)$ using a hash table. Hence, as

there are $\tilde{O}(N)$ pairs of slid chains, each with $\tilde{O}(N)$ possible offsets, the total running time of the distinguisher is $\tilde{O}(N^3)$.

There are two technical details to note: First, the PRF reconstruction attack described in Section 4.4 requires $\tilde{O}(N^{3/2})$ input/output pairs for the 4-round FF3 construction. As a result, in attacks that use shorter chains, we need to ask for the extension of the identified slid chains. Luckily, in an adaptive chosen plaintext and ciphertext attack scenario, that merely means we need to ask for at most two chains of $\tilde{O}(N^{3/2})$.

Second, while previous distinguishing attacks were sufficiently good when the probability of a wrong chain to pose a slid chain was $1/\sqrt{N}$, we need a better filter. This filter is needed as to avoid the increase in the data complexity explained earlier. Hence, we need to ask that the probability of a wrong candidate to pass the distinguisher is no more than $(1/N^{1-t})$. The distinguisher can be applied twice, and thus out of the N^2 wrong slid chains/offsets, we get $\tilde{O}(N^{2t})$ candidate slid chains. This is sufficient to ensure the complete attack does not use more than $\tilde{O}(N^{7/4-t})$ data and $O(N^{5/2+2t})$ time.

Algorithm 4: Time-Data Tradeoff Variant of the Symmetric Slide for FF3

Input : $\tilde{O}(N^{1/4+t})$ chains C^r of $\tilde{O}(N^{3/2-2t})$ plaintexts encrypted under K and $T = T_L||T_R$

Input : $\tilde{O}(N^{1/4+t})$ chains $\hat{C}^s$ of $\tilde{O}(N^{3/2-2t})$ plaintexts encrypted under K and $T' = T_L \oplus 4||T_R \oplus 4$

Output: Slid chains $C^i, \hat{C}^j$ and their respective offset

for *all chains* C^r **do**
  Initialize a hash table H_1
  Insert all the plaintexts $(L_0^i, R_0^i) \in C^r$ into H_1 indexed by R_0^i
  Take $O(N^{4t})$ bins (each with $O(N^{1/2-2t}$ plaintexts)
  Denote the plaintexts by $X_{i_1}, X_{i_2}, \ldots, X_{i_v}$
  for *all chains* $\hat{C}^s$ **do**
    for *all respective offsets* $u = 0, \ldots, N^{3/2-2t}$ **do**
      Extract $(\hat{L}_0^{u+i_1}, \hat{R}_0^{u+i_1})$, $(\hat{L}_0^{u+i_2}, \hat{R}_0^{u+i_2})$,..., $(\hat{L}_0^{u+i_v}, \hat{R}_0^{u+i_v})$ from $\hat{C}^s$
      Denote these values as "ciphertexts" $C_{i_1}, C_{i_2}, \ldots, C_{i_v}$
      Initialize $O(N^{4t})$ hash tables H_2^j
      for *all k=1,...,v* **do**
        if X_{i_k} *is from bin* j **then**
          Store in H_2^j the value $LH(C_{i_k}) - LH(X_{i_k})$
      Count the number of collisions in all H_2^j
      if *number of collisions is greater than* $\frac{1.6}{N} \cdot \Sigma_{B \in bins} \binom{|B|}{2}$ **then**
        Ask for the extension of C^r and $\hat{C}^s$ to $\tilde{O}(N^{3/2})$ values.
        Call the PRF-recovery procedure with C^r as inputs and $\hat{C}^s$ shifted by u as the outputs.

4.2 Cycle Structure Attack

The second attack follows the footsteps of [4] to find candidate slid chains. Consider a related-tweak slid pair (L_0, R_0) and $(\hat{L}_0, \hat{R}_0)$, i.e., 4-round FF3 with the key K and T partially encrypts (L_0, R_0) into $(\hat{L}_0, \hat{R}_0)$. If we start a chain of encryption from (L_0, R_0), we are assured to reach (L_0, R_0) again after some number of encryptions $t \leq N^2$. Due to the slid property, the same is true also for $(\hat{L}_0, \hat{R}_0)$, i.e., after t encryptions under K and T', we are assured to reach $(\hat{L}_0, \hat{R}_0)$ again. It is easy to see that this value does not repeat before t encryptions (as otherwise, (L_0, R_0) would also close the chain earlier). Hence, there is no point to check whether two chains can be slid chains, if their cycle length is not equal.

The attack thus tries to find chains which are actually cycles, of length $\tilde{O}(N^{3/2})$ (as this is the amount of data needed for the PRF reconstruction). We note that following Shepp and Lloyd's results [23] it is reasonable to assume that (a) such a cycle exists, and (b) that it is unique. Of course, if by chance the unlikely event happens, and there are two cycles in the encryption under K and T of exactly the same length of $\tilde{O}(N^{3/2})$, we can just try all pairs of chains, or just take the next larger cycle.

Once this pair of cycles is identified, one can run the distinguisher used before for all possible $\tilde{O}(N^{3/2})$ offsets. As the cost of the distinguisher is $\tilde{O}(N^{1/2})$, the total time complexity needed to identify the exact offset between the chains is $\tilde{O}(N^2)$. When the correct offset is identified, it is possible to run the PRF reconstruction attack as we have obtained $\tilde{O}(N^{3/2})$ input/output pairs for 4-round FF3.

Given that the PRF reconstruction takes $\tilde{O}(N^{3/2})$ time, we can call it at most $\tilde{O}(\sqrt{N})$ times. This requires that the filtering is set such that the probability of a random permutation to pass the threshold be below $\tilde{O}(1/\sqrt{N})$ (as the distinguisher can be applied twice in each offset, this rate is sufficient to discard all but a fraction of $\tilde{O}(1/N)$ of the wrong offsets).

The data complexity of the attack is about $O(N^2)$ encryptions: An adaptive chosen-plaintext attack would be based on picking a random plaintext, generating a cycle from it, and then, check whether the cycle has the right length. If not, an unseen plaintext needs to be picked, and the process is repeated. It is easy to see that the process is expected to finish after exploring almost all plaintexts (as most of the values lie in the larger cycles, e.g., the largest one of size about $(1 - 1/e) \cdot N^2$). A simple analysis suggests that about $\tilde{O}(N^{3/2})$ of the values remain "unseen" once the cycle of length $\tilde{O}(N^{3/2})$ is identified.

Another approach is to collect $N^2 - \sqrt{N}$ known plaintext pairs. If all the values in the cycle of length $\tilde{O}(N^{3/2})$ are not in the missing $\sqrt{N}$ ones, which happens with constant probability, then the cycle can be identified and used for the attack.

Hence, to conclude, this first phase of the attack (for the detection of slid pairs) takes data $O(N^2)$ and time $\tilde{O}(N^2)$. The resulting attack algorithm is given in Alg. 5 (we describe the known plaintext variant, but it is very similar to the adaptive chosen plaintext one).

Algorithm 5: The Cycle Structure Slide Distinguisher for FF3

Input : $N^2 - N$ known plaintexts (P^i, C^i) encrypted under K and $T = T_L || T_R$
Input : $N^2 - N$ known plaintexts $(\hat{P}^i, \hat{C}^i)$ encrypted under K and $T' = T_L \oplus 4 || T_R \oplus 4$
Output: Slid chains $C, \hat{C}$

Initialize a bitmap B of N^2 bits to 0.
while *no cycle C of size $\tilde{O}(N^{3/2})$ was found* **do**
  Pick the first plaintext whose bit is not set in B — P_0.
  Set $B[P_0] = 1$, Set $t = 0$
  repeat
    Set $P_{t+1} = C_t (= E_{K,T}(P_t))$
    if P_{t+1} *is not in the dataset* **then**
      break (goto 2)
    Set $B[P_{t+1}] = 1$; Set $t = t + 1$
  until $P_t = P_0$;
  if $t = \tilde{O}(N^{3/2})$ **then**
    Set C to be $P_0, P_1, \ldots, P_{t-1}$
Initialize a bitmap B of N^2 bits to 0.
while *no cycle $\hat{C}$ of size t was found* **do**
  Pick the first plaintext whose bit is not set in B — $\hat{P}_0$.
  Set $B[\hat{P}_0] = 1$, Set $s = 0$
  repeat
    Set $\hat{P}_{s+1} = \hat{C}_s (= E_{K,T}(\hat{P}_s))$
    if $\hat{P}_{s+1}$ *is not in the dataset* **then**
      break (goto 2)
    Set $B[\hat{P}_{s+1}] = 1$; Set $s = s + 1$
  until $\hat{P}_s = \hat{P}_0$;
  Set $\hat{C}$ to be $\hat{P}_0, \hat{P}_1, \ldots, \hat{P}_{t-1}$
for *all possible offsets* **do**
  Call the differential distinguisher for any offset between C and $\hat{C}$
  if *the distinguisher succeeds* **then**
    Call the PRF reconstruction attack with C,$\hat{C}$, and the offset

4.3 Asymmetric Slide Attack

The new attack follows the footsteps of the low data distinguisher presented in Section 4.1, but offers an improved distinguishing algorithm as well as a tradeoff curve. The data and memory complexity of the attack is $\tilde{O}(N^{2-t})$ with time complexity $\tilde{O}(N^{2+t})$ for $t \in [0, 1/2]$.

This related-tweak slide differential distinguisher uses the minimal amount of pairs ($O(N \log N)$) similarly to the one of Section 4.1. The key element in it is the algorithmic gain, coming from searching the pairs from *the plaintext's side*.

Consider an input chain of $\tilde{O}(N)$ values. We preprocess the chain by computing for each of the $\tilde{O}(N)$ input pairs with a common right half P_i, P_j the value $LH(P_j) - LH(P_i), j - i$ and storing it in a hash table. In other words, we store for each pair the difference in the left half and the location difference. To prepare this table, we need $\tilde{O}(N)$ memory, where each cell contains about $\tilde{O}(1)$ values.[7] The table can be calculated in $\tilde{O}(N)$ time by using a supporting hash table keyed according to the right-hand-side of the plaintexts.

From the output side, we take a chain of length $\tilde{O}(N)$. We initialize $\tilde{O}(N)$ counters to zero. Then, we can compute for each such pair $(C_{i'}, C_{j'})$ the value $(LH(C_{j'}) - LH(C_{i'}), j' - i')$, and find the offset it proposes in the table. We then increment the $\tilde{O}(1)$ counters related to the offset.[8] For the correct offset the amount of pairs that "succeed" is expected to be $2m/N$ out of m pairs, compared with m/N for wrong offsets (or wrong chains). If the preprocessed input chains are all keyed into the same hash table, this search can be done simultaneously against all $\tilde{O}(N^{1-t})$ of them, taking only $\tilde{O}(N^2)$ time per output chain.

The attack is thus based on taking $\tilde{O}(N^t)$ output chains of length $\tilde{O}(N)$ and $\tilde{O}(N^{1-t})$ input chains of length $\tilde{O}(N)$. For each input chain we perform $\tilde{O}(N)$ preprocessing. Then we try $\tilde{O}(N^t)$ chains in time $\tilde{O}(N^{2+t})$, i.e., a time of $\tilde{O}(N^2)$ per output chain. This results in time complexity of N^{2+t} and data complexity which is $\max\{\tilde{O}(N^{1+t}), \tilde{O}(N^{2-t})\}$ (which if $t \in [0, 1/2]$ suggests $\tilde{O}(N^{2-t})$. The memory complexity is comprised of $\tilde{O}(N^{1-t})$ preprocessed plaintext tables of size $\tilde{O}(N)$ each, meaning $\tilde{O}(N^{2-t})$ in total (the amount of counters in the in the sliding part of the attack is also $\tilde{O}(N^{2-t})$).

The full attack algorithm is given in Algorithm 6.

4.4 The PRF Reconstruction Procedure

Our PRF reconstruction procedure follows the foot steps of Durak and Vaudenay and of Hoang et al. We use a graph where cycles are searched for. We follow Hoang et al.'s approach, and call the PRF reconstruction fewer times than there are candidate slid chains. However, to reduce the data and time complexities of our attack (which is needed as our slid chain detection is more efficient) we use cycles of larger size, i.e., we pick $L = 4$ and $L = 5$ rather than $L = 3$.

This means that for finding sufficiently large connected component between all the values, it is sufficient that from any node in the graph, there will be only $\tilde{O}(N^{1/L})$ outgoing edges (instead of $\tilde{O}(N^{1/3})$ needed for $L = 3$).

Hence, we are left with the problem of finding cycles of length L in a graph of $\tilde{O}(N)$ nodes, with an average out degree of $\tilde{O}(N^{1/L})$. Our algorithm just

[7] The number of actual values per cell follows a Poisson distribution, i.e., there may be a few cells with $\tilde{O}(\log N)$ values.

[8] We note that the table is expected to have 1 value on average in each cell. However, the actual number is distributed according to a Poisson distribution with this mean. Hence, some cells will be empty, and a few will have several possible offsets. When there are multiple offsets, we just increment all the counters corresponding to the offsets.

Algorithm 6: The Asymmetric Slide Distinguisher for FF3

Input : $\tilde{O}(N^{1-t})$ chains C^r of $q = \tilde{O}(N)$ plaintexts encrypted under K and $T = T_L||T_R$
Input : $\tilde{O}(N^t)$ chains $\hat{C}^s$ of $q = \tilde{O}(N)$ plaintexts encrypted under K and $T' = T_L \oplus 4||T_R \oplus 4$
Output: Slid chains $C^i, \hat{C}^j$ and their respective offset

```
Initialize a hash table H_1
for all chains C^r_k ∈ C^r do
    for all i, j where R^j_k = R^i_k do
        Store in H_1 in location (j − i, L^j_k − L^i_k) the value (k, i)
for all chains Ĉ^s_{k'} ∈ Ĉ^s do
    Initialize Õ(N^{2−t}) counters
    for all i' = 0, 1, ..., Õ(N) do
        for all j' = i' + 1, i' + 2, ..., Õ(N) do
            for all k, i ∈ H_1[j' − i', L^{j'}_{k'} − L^{i'}_{k'}] do
                if i' < i then
                    Increment the counter of chain k and offset i − i'
    Identify k, v such that counter[k][v] is maximal
    if counter[k][v] > 1.6 · (q choose 2) · 1/N^2 then
        Ask for the extension of C^r_k and Ĉ^s_k to Õ(N^{3/2}) values.
        Call the PRF reconstruction with the chains C^r_k, Ĉ^s_{k'}, with offset v
```

performs a simple meet in the middle procedure: From each node we detect all possible $\tilde{O}(N^{1/L})^{\lfloor L/2 \rfloor}$ nodes in distance $\lfloor L/2 \rfloor$, and then detect all the possible $\tilde{O}(N^{1/L})^{\lceil L/2 \rceil}$ nodes in distance minus $\lceil L/2 \rceil$ (i.e., when walking on the reversed edges graph) and find a collision between these sets (which correspond to a cycle of length L).

Similarly to [12,17], once the cycles are found, all the involved nodes are assumed to be good nodes, and they can be used to determine values for F_0. Heuristically, we found out that filling in $1/3 \cdot log(N)\sqrt{N}$ values of F_0 gives a high chance of success for the recovery attack on F_1, F_2, F_3 (exactly as proposed in [17]). If indeed the reconstruction is consistent with the slid chains, then we continue to reconstruct the missing values in F_0 (as we know the full F_1, F_2, F_3), and apply the recovery attack to the second half (i.e., swapping the order of the slid chains w.r.t. input/output).

On the other hand, if the results are inconsistent with the slid chain, we try a different value to start the assignment from (from a different connected component), or try a different slid chain (when there are other candidates). This part is similar to that of [17].

5 Experimental Verification

We implemented all of our attacks and experimentally verified their correctness and success probability. The code and instructions to reproduce our results is available at `https://github.com/OhadAm7/FF3-code`.

5.1 Experimental Verification of the Symmetric Slide Attack

We experimentally verified the full implementation of the symmetric slide attack (described in Section 4.1) for both $t = 0.25$ and $t = \frac{3}{4log(N)}$ (which leads to using 8 bins in the distinguisher). We tested the attacks for various values of $N = 2^n$ (n is half the bit size of the encryption domain) and various cycle sizes L. To calculate each attack's overall success probability for each parameter choice, we repeated the attacks 100 times using different random keys and tweaks. The results can be seen in Table 3 and Table 4.

"PRF Reconstruction" in the following tables denotes the success rate of the PRF Reconstruction subroutine over all calls. "Combined Reconstruction" denotes the rate at which both calls to the PRF Reconstruction subroutine for a single slide succeed, resulting in the full codebook being recovered.

Note that the PRF reconstruction rate for smaller domain sizes is lower than expected. This is due to overlap between the different chains that mean there is a correlation between multiple reconstruction attempts. As we continue trying different chains after failed reconstruction attempts but stop after the first successful one, the reconstruction rate is skewed to lower values.

5.2 Experimental Verification of the Cycle Structure Attack

We also tested the cycle structure attack (described in Section 4.2) for various values of $N = 2^n$ and L. These experiments were also repeated 100 times each using random keys and tweaks. The results are presented in Table 5.

Note that the success rate has a slight drop above $N = 2^9$. This is due to runs that fail to find a cycle of length between q and e^2q (where q is some $\widetilde{O}(N^{\frac{3}{2}})$ required for the distinguisher). With our parameters, $e^2q > N^2$ so there is no upper bound, and the probability of finding a cycle is very high. For $N > 2^9$ the probability drops to a constant but lower probability.

5.3 Experimental Verification of the Asymmetric Slide Attack

We also performed experimental verification of the Asymmetric Slide Attack (described in Section 4.3). This was tested both for a constant number of ciphertext chains (3) and for $t = 0.5$. We ran both experiments for 100 times each on random keys and tweaks. The results are presented in Table 6 and Table 7, respectively.

N	L	Queries	Success Rate	PRF Reconstruction	Combined Reconstruction	Distinguisher (Cipher)	Distinguisher (Rand)
2^6	3	3679	0.26	0.527	0.286	0.919	1.0
2^7	3	12631	0.35	0.798	0.614	0.891	1.0
2^8	3	41578	0.37	0.963	0.925	0.93	1.0
2^9	3	203414	0.79	1.0	1.0	0.952	1.0
2^{10}	3	695431	0.76	1.0	1.0	1.0	1.0
2^6	4	3679	0.12	0.34	0.128	0.922	1.0
2^7	4	12631	0.14	0.484	0.226	0.912	1.0
2^8	4	41546	0.22	0.744	0.564	0.929	1.0
2^9	4	201246	0.75	0.963	0.926	0.953	1.0
2^{10}	4	670125	0.75	1.0	1.0	1.0	1.0
2^6	5	3679	0.0	0.042	0.0	0.922	1.0
2^7	5	12631	0.0	0.07	0.0	0.901	1.0
2^8	5	41546	0.0	0.025	0.0	0.93	1.0
2^9	5	201246	0.02	0.182	0.015	0.971	1.0
2^{10}	5	670125	0.31	0.557	0.32	1.0	1.0

Table 3: Symmetric Slide Attack Experiment Results ($num\ bins = 8.0$)

N	L	Queries	Success Rate	PRF Reconstruction	Combined Reconstruction	Distinguisher (Cipher)	Distinguisher (Rand)
2^6	3	3608	0.24	0.53	0.242	0.915	0.999
2^7	3	13472	0.49	0.765	0.59	0.896	0.999
2^8	3	48432	0.66	0.971	0.943	0.928	1.0
2^9	3	173349	0.69	1.0	1.0	0.921	1.0
2^{10}	3	594010	0.73	1.0	1.0	1.0	1.0
2^6	4	3492	0.11	0.283	0.096	0.909	0.998
2^7	4	12932	0.26	0.5	0.255	0.892	0.999
2^8	4	44252	0.5	0.799	0.61	0.924	1.0
2^9	4	154090	0.67	0.965	0.931	0.921	1.0
2^{10}	4	516131	0.74	1.0	1.0	1.0	1.0
2^6	5	3416	0.0	0.031	0.0	0.908	0.999
2^7	5	12438	0.0	0.021	0.0	0.895	0.999
2^8	5	42902	0.0	0.057	0.0	0.929	1.0
2^9	5	150015	0.04	0.163	0.033	0.949	1.0
2^{10}	5	488380	0.31	0.548	0.27	0.984	1.0

Table 4: Symmetric Slide Attack Experiment Results ($t = 0.25$)

N	L	Queries	Success Rate	PRF Reconstruction	Combined Reconstruction	Distinguisher (Cipher)	Distinguisher (Rand)
2^6	3	3851	0.24	0.542	0.289	0.874	1.0
2^7	3	15386	0.61	0.824	0.67	0.929	1.0
2^8	3	60686	0.89	0.954	0.908	0.98	1.0
2^9	3	244674	0.99	1.0	1.0	0.99	1.0
2^{10}	3	948308	0.9	1.0	1.0	0.978	1.0
2^{11}	3	3825251	0.85	1.0	1.0	0.988	1.0
2^{12}	3	16303811	0.86	1.0	1.0	1.0	1.0
2^6	4	3851	0.05	0.307	0.06	0.874	1.0
2^7	4	15386	0.18	0.462	0.198	0.929	1.0
2^8	4	60686	0.7	0.837	0.714	0.98	1.0
2^9	4	244674	0.91	0.965	0.929	0.99	1.0
2^{10}	4	948308	0.78	1.0	1.0	0.975	1.0
2^{11}	4	3923124	0.88	1.0	1.0	1.0	1.0
2^{12}	4	16366382	0.88	1.0	1.0	0.989	1.0
2^6	5	3851	0.01	0.072	0.012	0.874	1.0
2^7	5	15386	0.0	0.027	0.0	0.929	1.0
2^8	5	60686	0.01	0.102	0.01	0.98	1.0
2^9	5	244674	0.04	0.187	0.04	0.99	1.0
2^{10}	5	948308	0.2	0.538	0.256	0.975	1.0
2^{11}	5	3923124	0.62	0.841	0.705	1.0	1.0
2^{12}	5	16366382	0.86	0.989	0.977	0.989	1.0

Table 5: Cycle Structure Attack Experiment Results ($num\ bins = 8.0$)

6 A New Class of Attacks on Cycle Walking FPE Schemes

In this section we point out that generic FPE schemes which are based on the cycle walking idea may be highly vulnerable to a new class of attacks which we call *Related Domain Attacks*. These attacks are similar to related key or related tweak attacks, but can only be applied to FPE schemes in which we can dynamically change the declared size of the input domain.

To demonstrate the basic form of these attacks, consider an iterated FPE scheme which consists of an arbitrarily large number k of round functions, each one of which is a different keyed permutation over the input domain. Furthermore, we assume that the round function itself takes a value from a domain of size N and processes it using the cycle walking idea. In other words, the round function may cause the value to be outside the domain, in which case the same round function is applied again and again until the value resides again in the domain. For the sake of discussion, we assume that extracting the secret key from known or chosen plaintext/ciphertext pairs is infeasible, but that each keyed round function by itself is sufficiently simple that finding the key from the two-

N	L	Queries	Success Rate	PRF Reconstruction	Combined Reconstruction	Distinguisher Success Rate
2^6	3	3863	0.31	0.5	0.295	0.417
2^7	3	13883	0.47	0.709	0.527	0.368
2^8	3	52526	0.63	0.921	0.9	0.326
2^9	3	193894	0.7	0.959	0.959	0.339
2^{10}	3	752222	0.63	0.955	0.955	0.304
2^{11}	3	2946123	0.7	1.0	1.0	0.344
2^6	4	3828	0.12	0.303	0.101	0.436
2^7	4	13775	0.2	0.451	0.194	0.373
2^8	4	51264	0.51	0.794	0.637	0.327
2^9	4	190204	0.68	0.929	0.883	0.347
2^{10}	4	733040	0.64	0.97	0.97	0.302
2^{11}	4	2874664	0.7	0.986	0.986	0.344
2^6	5	3824	0.0	0.035	0.0	0.435
2^7	5	13622	0.0	0.047	0.0	0.39
2^8	5	51099	0.0	0.078	0.0	0.34
2^9	5	189543	0.07	0.225	0.069	0.354
2^{10}	5	733453	0.28	0.562	0.35	0.297
2^{11}	5	2868815	0.53	0.813	0.639	0.355

Table 6: Asymmetric Slide Attack Experiment Results with constant number of 3 ciphertext chains

round version of this scheme is practically doable (this assumption is similar in nature to that of the slide attack [7]).

Assume further that the permutation P used in each round follows the cycle walking paradigm: If we declare that the input domain is $\{1, 2, \ldots, N\}$, then for any input x which is in this domain, we output the first value y that follows x along its cycle in P which is in the domain (possibly going all the way until we reach x again). This guarantees that all the intermediate values encountered during the encryption are valid values in the domain, and that any such y can be uniquely decrypted to x.

The related domain attack uses two very similar domains: The first one is defined as $\{1, 2, \ldots, N\}$ and the second one is defined as $\{1, 2, \ldots, N-1\}$. When we use a keyed round permutation on the first domain, we skip over all the possible values of the permutation which are larger than N. When we use the same keyed round permutation on the second domain, we skip over all the values which are larger than $N-1$. The two permutations are almost identical, and the only difference between them is related to the single value N which is allowed in the first permutation but forbidden in the second permutation. More specifically, given the preimage of N in the first permutation, we compute its output as N in the first permutation, but as the postimage of N in the second permutation.

N	L	Queries	Success Rate	PRF Reconstruction	Combined Reconstruction	Distinguisher Success Rate
2^6	3	3856	0.34	0.47	0.252	0.408
2^7	3	13752	0.58	0.766	0.615	0.268
2^8	3	48302	0.69	0.934	0.908	0.182
2^9	3	161676	0.69	0.932	0.932	0.142
2^{10}	3	543516	0.77	0.891	0.885	0.129
2^{11}	3	1769542	0.79	0.975	0.975	0.09
2^6	4	3828	0.18	0.315	0.116	0.404
2^7	4	13757	0.26	0.432	0.195	0.292
2^8	4	46880	0.54	0.787	0.621	0.181
2^9	4	153589	0.68	0.914	0.895	0.139
2^{10}	4	504930	0.8	0.982	0.976	0.128
2^{11}	4	1624932	0.81	0.988	0.988	0.092
2^6	5	3808	0.0	0.04	0.0	0.418
2^7	5	13702	0.0	0.047	0.0	0.296
2^8	5	47486	0.01	0.081	0.007	0.194
2^9	5	153608	0.08	0.25	0.068	0.137
2^{10}	5	498716	0.3	0.5	0.254	0.111
2^{11}	5	1564668	0.7	0.863	0.737	0.087

Table 7: Asymmetric Slide Attack Experiment Results ($t = 0.5$)

To apply our new adaptive chosen message attack, we perform the full k - round encryption of N in the first domain, getting the ciphertext $z = E_1(N)$. With high probability, z is different than N, and thus we can request its decryption $w = E_2^{-1}(z)$ as a member of the second domain (using the same unknown key and known tweak). Consider now the composition of these functions $w = E_2^{-1}(E_1(N))$. With high probability, none of the intermediate values will be N, and thus we can cancel almost all the $2k$ rounds in matching pairs. The only thing left will be the two round version of the problem in which N encrypted by the first round of E_1 and then decrypted by the first round of E_2 is equal to w. By repeating this process several times with shrinking domains, we can get sufficiently many input/output pairs, which are presumably enough to find the key used by this first round of the original scheme. If each round permutation uses a different key, we can easily repeat the process in order to find the keys of all the subsequent rounds. Note that this kind of attack can also be used against Feistel structures.

To protect FPE schemes against this new kind of attack, we propose to use the declared size of the domain as part of the tweak in each round function. This is a very simple modification which costs almost nothing but will make sure that any change in the domain size will result in a new and unrelated round function.

7 A Related Domain Distinguishing Attack on FF3 and FF3-1

We now present a distinguishing attack on both FF3 and FF3-1. The distinguishing attack is a related-domain attack that highlights the importance of domain separation between different instances of the encryption algorithm. For example, FF1 [14] uses the input size parameters as an input to the round function, thus avoiding this attack.

The distinguishing attack is quite efficient. Given about $c \cdot 2^4$ pairs of chosen plaintexts, we can distinguish whether two FF3 or FF3-1 instances were applied with related domain sizes (using the same key and tweak) for binary domains. Note that FF3 supports plaintexts encoded in any base (denoted as *radix* in the standard). One can easily expand the distinguishing attack to use $c \cdot radix^4$ pairs of chosen plaintext to handle different *radix*-sizes.

Hence, for the sake of simplicity we will describe an attack on a binary domain. As mentioned before, one can easily extend it to any *radix*. Let D_1 be a domain that includes $2n$-bit plaintexts, whereas domain D_2 includes $2n+1$-bit plaintexts. In other words, in D_1, the plaintexts have n-bit halves, whereas in D_2 the halves are n-bit and $n+1$-bit, respectively.

The adversary is given access to two encryption oracles $\mathcal{O}_1$ (over D_1) and $\mathcal{O}_2$ (over D_2). Similarly to the related cipher scenario [26], either these two oracles are two independent random permutations (of different sizes), or they are $FF3$ instantiated with the same key K and tweak T. We note that the attack also works against FF3-1 without any change, since the only difference between them is in the way they deal with the tweak.

Consider a plaintext (x, y) encrypted using FF3 (or FF3-1) in the smaller domain D_1 to (z, w). During its encryption, there are 8 invocations of the AES function using the same key K and tweak T, but with different inputs and round constants. When we encrypt $(0||x, y)$ in the larger domain D_2, we also get 8 invocations of AES. In the first round, we get the same input (y in both cases), but this time, $n+1$ bits of the AES output are used in the modular addition (instead of the previous n). Let the n-bit output (for D_1) be denoted by α and for the $n+1$-bit output be denoted by $b||\alpha$. It is easy to see that if $x + \alpha < 2^{10}$ then $(0||x) + (b||\alpha) \bmod 2^{11} = (b||x + \alpha)$, otherwise, $(0||x) + (b||\alpha) \bmod 2^{11} = (\bar{b}||x + \alpha \bmod 2^{10})$. It is easy to see that independent of the value of b, with probability of $1/2$, the addition's output is $(0||x + \alpha \bmod 2^{10})$. When this happens, the actual input to the AES invocation in the second round is the same for both the smaller domain D_1 and the larger domain D_2, which suggests the same output of the second round function. Hence, the input to the third round is also the same. The third round is similar to the first, and indeed, with probability $1/2$, the MSB of the output of addition is also 0. This also repeats in rounds 5 and 7. In other words, with probability $1/16$, if $(0||x, y)$ is encrypted to $(0||z, w)$ in the larger domain, when $(x||y)$ is encrypted to (z, w) in the smaller domain.

It is easy to see that the probability that this holds for two random permutation (over $2n$-bit and $2n+1$-bit values, respectively) is much smaller (namely

2^{-2n}). This produces a very efficient distinguisher: Pick $c \cdot 16$ random plaintexts $P_i = (x_i, y_i) \in D_1$, and ask for their encryptions under $\mathcal{O}_1$, resulting in C_i. Then, ask for the encryptions of the $c \cdot 16$ plaintexts $(0||x_i, y_i)$ under $\mathcal{O}_2$, resulting in $\hat{C}_i$. If for about $c/16$ of the ciphertexts $\hat{C}_i = 0||C_i$, conclude that this is FF3; otherwise, conclude that the two oracles are independent random permutations. For small values of c, where the probability of obtaining a right pair in the random case is negligible, the success rate of the attack is about $1 - e^{-c}$. Hence, setting $c = 2$ (and using 32 pairs in total) results in a success rate of 86.5%.

The generalization of the above attack to larger radices is trivial. We just need to assume that the most significant character (rather than bit) is 0, which happens with probability $1/radix$. Hence, one can construct an efficient distinguisher with data and time complexity of $c \cdot radix^4$ chosen plaintext pairs.

8 Conclusions

In this paper we studied the FF3 format preserving encryption algorithm. Building on top of the previous ideas of using related-tweak slide attack against FF3, we presented three attacks: An improved symmetric slide attack which enjoys better time, data and memory complexity compared with previous results, a cycle detection slide attack, and a asymmetric slide attack which outperforms the symmetric one.

We also presented two related-domain attacks. The first, a generic attack against cycle walking schemes, which reduces the problem of breaking them into the problem of attacking two rounds of the construction. The second, which is applicable to FF3 (and FF3-1) offers efficient distinguishing and shows how to expand the knowledge on the encryption in the smaller domain, to recover the PRFs used in the bigger one.

References

1. 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, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 444–455. ACM, 2016.
2. M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-Preserving Encryption. In M. J. J. Jr., V. Rijmen, and R. Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2009.
3. M. Bellare, P. Rogaway, and T. Spies. The FFX Mode of Operation for Format-Preserving Encryption (Draft 1.1). Available online at `csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec.pdf`, 2010.
4. E. Biham, O. Dunkelman, and N. Keller. Improved Slide Attacks. In A. Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*, pages 153–166. Springer, 2007.

5. E. Biham, O. Dunkelman, and N. Keller. A Unified Approach to Related-Key Attacks. In K. Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 73–96. Springer, 2008.
6. A. Biryukov, G. Leurent, and L. Perrin. Cryptanalysis of Feistel Networks with Secret Round Functions. In O. Dunkelman and L. Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 102–121. Springer, 2015.
7. A. Biryukov and D. A. Wagner. Slide Attacks. In L. R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999.
8. J. Black and P. Rogaway. Ciphers with Arbitrary Finite Domains. In B. Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, volume 2271 of *Lecture Notes in Computer Science*, pages 114–130. Springer, 2002.
9. E. Brier, T. Peyrin, and J. Stern. BPS: a Format-Preserving Encryption Proposal. Available online at `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf`, 2010.
10. M. Brightwell and H. Smith. Using datatype-preserving encryption to enhance data warehouse security. volume pp., pages 141–149, 1997. Available at `http://csrc.nist.gov/niccs/1997`.
11. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
12. F. B. Durak and S. Vaudenay. Breaking the FF3 format-preserving encryption standard over small domains. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 679–707, 2017.
13. M. Dworkin. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. *NIST Special Publication*, 800-38G, 2016.
14. M. Dworkin. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. *NIST Special Publication*, SP 800-38G Rev. 1, 2019.
15. R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.
16. S. Furuya. Slide Attacks with a Known-Plaintext Cryptanalysis. In K. Kim, editor, *Information Security and Cryptology - ICISC 2001, 4th International Conference Seoul, Korea, December 6-7, 2001, Proceedings*, volume 2288 of *Lecture Notes in Computer Science*, pages 214–225. Springer, 2001.
17. V. T. Hoang, D. Miller, and N. Trieu. Attacks only Get Better: How to Break FF3 on Large Domains. In Y. Ishai and V. Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 85–116. Springer, 2019.
18. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
19. National Bureau of Standards. DATA ENCRYPTION STANDARD (DES). Technical report, 1977. Federal Information Processing Standards Publication 46.

20. J. Patarin. Generic Attacks on Feistel Schemes. *IACR Cryptol. ePrint Arch.*, 2008:36, 2008.
21. J. Patarin. Security of balanced and unbalanced Feistel Schemes with Linear Non Equalities. *IACR Cryptol. ePrint Arch.*, 2010:293, 2010.
22. R. Schroeppel and H. Orman. The hasty pudding cipher. *AES candidate submitted to NIST*, page M1, 1998.
23. L. Shepp and S. Lloyd. Ordered cycle lengths in a random permutation. *Transactions of the American Mathematical Society*, 121(2):340–357, 1966.
24. T. Spies. Feistel Finite Set Encryption Mode, 2008. NIST submission.
25. J. Vance. VAES3 scheme for FFX: An addendum to "The FFX Mode of Operation for Format-Preserving Encryption": A parameter collection for encipher strings of arbitrary radix with subkey operation to lengthen life of the enciphering key. Available online at `csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-ad-VAES3.pdf`, 2010.
26. H. Wu. Related-Cipher Attacks. In R. H. Deng, S. Qing, F. Bao, and J. Zhou, editors, *Information and Communications Security, 4th International Conference, ICICS 2002, Singapore, December 9-12, 2002, Proceedings*, volume 2513 of *Lecture Notes in Computer Science*, pages 447–455. Springer, 2002.