Finite Set Format-Preserving Encryption Algorithm

Authors: Wang Xiaofeng, Ye Jun, Liu Wenzheng, Sun Bing, Wu Huahui

Date: 2025-10-16T00:00:00+00:00

Abstract

Format-Preserving Encryption (FPE) is widely employed for sensitive data privacy protection; however, existing standard algorithms exhibit vulnerabilities to slide attacks and linear cryptanalysis while demonstrating suboptimal efficiency. This paper proposes the Finite Basis Format-Preserving Encryption (FR-FPE) algorithm, which introduces a parameter vector encryption mechanism based on CBC mode and a structured tweakable full-correlation encryption model to effectively resist slide attacks and linear cryptanalysis. Furthermore, it designs a lightweight deterministic encryption structure supporting plaintexts of up to 192 bits and tweaks of up to 96 bits, achieving a 45% reduction in block cipher invocations (9 invocations) compared to FF1 while maintaining equivalent security strength to NIST FF1. Utilizing the Game-Hopping proof technique, the paper establishes that the FR-FPE algorithm provides Strong Pseudo-Random Permutation (SPRP) security. Experimental evaluation demonstrates that under identical plaintext and tweak length configurations, FR-FPE achieves on average 41.56% more encryptions per second and 32.79% higher encryption data throughput relative to FF1.

Full Text

Preamble

Finite Radix Format-Preserving Encryption (FR-FPE) Algorithm

Xiaofeng Wang[1], Jun Ye[2], Wenzheng Liu[3], Bing Sun[4], Huahui Wu[5]
[1] (College of Computer, National University of Defense Technology, Changsha 410000, Hunan, China)
[2] (Hunan Secuber Information Technology Co., Ltd., Changsha 410000, Hunan, China)
[3] (Changsha University of Science and Technology, Changsha 410000, Hunan, China)

[4] (College of Science, National University of Defense Technology, Changsha 410000, Hunan, China)
[5] (Hunan Rural Credit Cooperative, Changsha 410000, Hunan, China)

Abstract: Format-Preserving Encryption (FPE) has extensive applications in protecting sensitive data privacy. However, existing standard algorithms struggle to defend against slide and linear cryptanalysis attacks while suffering from low efficiency. This paper proposes the Finite Radix Format-Preserving Encryption (FR-FPE) algorithm, which designs a CBC-mode-based parameter vector encryption mechanism and a structured tweak parameter fully-associated encryption model to effectively resist slide and linear cryptanalysis attacks. A lightweight deterministic encryption structure is developed to support encryption of plaintexts up to 192 bits and tweak parameters up to 96 bits. While ensuring the same security strength as NIST FF1, FR-FPE reduces block cipher invocations by 45% (9 fewer calls). Based on the Game-Hopping model, the paper proves that FR-FPE achieves Strong Pseudorandom Permutation (SPRP) security. Experiments demonstrate that under identical plaintext and tweak parameter lengths, FR-FPE achieves an average 41.56% higher encryption operations per second and 32.79% higher data throughput compared to FF1.

Keywords: Format-Preserving Encryption; Slide Attack; Linear Cryptanalysis; Strong Pseudorandom Permutation (SPRP)
Classification: TN918

With the development of information technology, various industries face substantial demands for encrypting fixed-format data, such as credit card numbers and identity card numbers. Traditional block cipher algorithms (e.g., SM4) typically require padding data into meaningless binary strings, which fails to preserve the length, format, and semantics of the plaintext data. This format alteration prevents ciphertext data from being compatible with existing systems or databases, necessitating costly and complex system modifications. To address this, Format-Preserving Encryption (FPE) technology was proposed to ensure that ciphertext length and type (e.g., digits, letters, Chinese characters) remain identical to the plaintext.

FPE technology has garnered significant attention from both academia and industry. As early as 1981, the U.S. National Bureau of Standards published FIPS 74, proposing methods for string encryption where plaintext and ciphertext share the same format [1]. Black et al. introduced three FPE construction methods: Prefix, Cycle-Walking, and Generalized-Feistel [2]. The U.S. National Institute of Standards and Technology (NIST) released the NIST SP 800-38G draft, proposing two FPE algorithm standards, FF1 and FF3 [3], which have been continuously updated. The American National Standards Institute published ANSI X9.124-2-2018 [4], proposing FPE algorithms based on counter keystream modes. South Korea also released FPE algorithm standards FEA-1 and FEA-2 [5].

Despite widespread adoption by international standards bodies and extensive

applications [6-9], the security of FPE algorithms remains a research focus, with slide attacks and linear cryptanalysis posing severe threats to current algorithms. Amon et al. [14] constructed three slide attack methods, where the slide attack exploiting cyclic structures achieves optimal data and time complexities of (ˆ{11/6}) and (ˆ{17/6}), respectively. Beyne [15] exploited the characteristic of round tweak parameters alternating between two values to launch message recovery attacks that probabilistically infer partial plaintext content. This linear cryptanalysis reduces the data complexity of FF3-1 message recovery attacks to (ˆ{2.5}), leading to FF3' s removal from the new NIST standard and compromising the security of South Korea' s FEA-1 and FEA-2 standards.

Currently, NIST FF1 is the only FPE block cipher standard that can resist linear cryptanalysis attacks. However, its algorithm structure designed for arbitrary-length plaintext results in numerous block cipher invocations and low efficiency. The Chinese Cryptography Industry Standardization Technical Committee' s GM/Y 5007-2024 research report introduced the TE-FPE(SM4) algorithm, which encrypts and truncates the tweak parameter as round function input based on the FF3 algorithm, thereby resisting linear analysis and slide attacks. However, TE-FPE has a complex and non-fixed structure: different plaintext lengths require different Feistel round numbers (8, 10, 12 rounds, etc.), different round function parameter lengths (56, 64 bits), and different round number field sizes (4, 8 bits), complicating hardware and software implementation. TE-FPE employs a short 56-bit tweak parameter, providing weak resistance against brute-force attacks in small-domain scenarios. Additionally, TE-FPE truncates the tweak parameter ciphertext after encryption, reducing ciphertext entropy and weakening the algorithm' s pseudorandom permutation (PRP) security strength.

To address these issues, this paper proposes the Finite Radix Format-Preserving Encryption (FR-FPE) algorithm, which supports encryption of plaintexts up to 192 bits and tweak parameters up to 96 bits with a fixed structure. FR-FPE significantly reduces block cipher invocations while resisting slide attacks and linear cryptanalysis. The main contributions of FR-FPE include:

1. **CBC-Mode-Based Parameter Vector Association Encryption Mechanism:** The algorithm constructs an initial vector P from plaintext parameters, block cipher algorithm identifiers, and high-order bytes of the tweak parameter. This vector is encrypted via block cipher (e.g., SM4) to obtain a pre-encryption value F = CIPH_K(P), which participates in all Feistel rounds through CBC mode. This strongly associates critical parameters throughout the entire encryption process, effectively resisting slide attacks.

2. **Structured Tweak Parameter Fully-Associated Encryption Model:** Unlike existing approaches that split the tweak parameter (or its encrypted version) into two parts for odd and even Feistel rounds, FR-FPE simultaneously involves both high-order and low-order bytes of the tweak parameter in all Feistel rounds (high-order bytes via CBC

mode, low-order bytes XORed with round number before concatenation with plaintext). This ensures the pseudorandom permutation property of tweak parameter encryption and its association with overall plaintext encryption, effectively resisting linear cryptanalysis. FR-FPE supports up to 96-bit tweak parameters, significantly increasing the difficulty of brute-force attacks on small domains [15].

3. **Lightweight Deterministic Encryption Structure:** FR-FPE adopts the same fixed 10-round Feistel structure as FF1. However, for plaintexts up to 192 bits and tweak parameters up to 96 bits, FR-FPE reduces block cipher invocations by 45% (9 fewer calls) compared to FF1, substantially improving algorithm efficiency while maintaining the same security strength as FF1.

Based on the Game-Hopping model, the paper proves that FR-FPE achieves Strong Pseudorandom Permutation (SPRP) security. Experiments show that under identical plaintext and tweak parameter lengths, FR-FPE achieves an average 41.56% higher encryption operations per second and 32.79% higher encrypted data throughput compared to FF1. Compared to TE-FPE, FR-FPE is 28.6% faster for plaintexts shorter than 30 bits but 15.1% slower for plaintexts longer than 30 bits (FR-FPE uses 2 more Feistel rounds, providing higher security).

2 Related Work

Recent research on FPE primarily encompasses four aspects: security models, algorithm models, attack techniques, and standardization.

Security Models: In 2002, Black et al. proposed that FPE is a special type of symmetric cipher [2], with fundamental building blocks being block ciphers and pseudorandom functions. Therefore, FPE security can be reduced to the security of these underlying modules, with the security goal being Pseudorandom Permutation (PRP) security. Combining this security goal with security models (e.g., non-adaptive chosen-plaintext attack CPA1, non-adaptive chosen-ciphertext attack CCA1) yields FPE security notions such as PRP-CPA1 and PRP-CCA1.

Algorithm Models: Black et al. proposed three FPE construction methods [16]: the Prefix method establishes permutations within the message space using pre-computed tables; the Cycle-Walking method ensures results fall within the valid domain through repeated encryption operations to meet format requirements; and the Generalized-Feistel method processes data halves through Feistel network iterations, becoming the widely adopted construction for subsequent FPE models. Spies et al. [16] proposed the FFSEM (Feistel Finite Set Encryption Modes) model based on balanced Feistel networks. Bellare et al. [18] introduced the FFX model based on Feistel networks, adding tweaks and establishing mapping tables between character data in message spaces and integer

domains. In 2012, Li Jingwei et al. [19] proposed an FPE scheme based on K-split type-2 Feistel networks to accommodate various data lengths.

Attack Techniques and Standardization: NIST published SP 800-38G in 2013, including two Feistel-based FPE algorithms, FF1 and FF3, with different round numbers, tweak parameters, and initial vector encryption methods. In 2014, Lee et al. [5] proposed FEA-1 and FEA-2 based on Feistel structures, supporting 128-bit, 192-bit, and 256-bit key lengths, which became South Korea' s FPE standard (TTAK.KO-12.0275).

Following the standard algorithms'proposal, Bellare et al. [20] presented message recovery attacks on Feistel-based FPE schemes at CCS 2016, demonstrating that for 8-bit messages, FF3 with 8-round General-Feistel structure could recover messages with only 2^{32} complexity. Durak and Vaudenay [21] improved attacks on FF1 and FF3 at CRYPTO 2017, showing that both schemes fail to provide 128-bit security for small message domains. Consequently, NIST released the revised SP 800-38G Revision 1 in 2019, correcting the applicability condition from radix^{min}*{len} >= 100 to radix^{min}*{len} >= 1,000,000 and adjusting FF3' s computation process, renaming it FF3-1.

At CRYPTO 2021 [22], Beyne reduced FF3-1' s message recovery attack data complexity to O(N^2.5) using tweak construction and linear analysis. Consequently, NIST' s second revision in 2025 removed FF3-1, retaining only FF1. The Chinese Cryptography Industry Standardization Technical Committee' s GM/Y 5007-2024 research report introduced TE-FPE(SM4), which encrypts and truncates the tweak parameter based on FF3 to resist linear analysis and slide attacks. However, TE-FPE has a non-fixed structure: different plaintext lengths require different Feistel round numbers (8, 10, 12 rounds, etc.), different round function parameter lengths (56, 64 bits), and different round number field sizes (4, 8 bits), complicating implementation. TE-FPE uses a short 56-bit tweak parameter, providing weak resistance against brute-force attacks in small-domain scenarios. Additionally, truncating the encrypted tweak parameter reduces ciphertext entropy, weakening the algorithm' s PRP security strength.

3 Preliminary Knowledge and Formal Definitions

This section defines the syntax and related cryptographic concepts for the Finite Radix Format-Preserving Encryption (FR-FPE) algorithm.

3.1 Algorithm Syntax

Key Space : The set of valid keys for the underlying block cipher CIPH, = {0,1}^{128}.

Format Space : The set of data format spaces, including length n, radix, and block cipher algorithm identifier cid. = (n, radix, cid).

Tweak Space : The set of permitted tweak parameters. = {0,1}^{96}.

Domain : The set of all possible plaintexts/ciphertexts. =

FR-FPE Encryption Function E: × × × → . For a fixed key K , format N , and tweak T , the function E_K,N,T is a permutation on _N, where E_K,N,T(•) E(K, N, T, •) maps _N to _N.

Decryption Function D: × × × → , where D_K,N,T is the inverse function of E_K,N,T.

N = (n, radix, cid) Σ_{radix}

3.2 Security Definitions

The essence of FPE is a pseudorandom permutation within a specific message space, with the security goal being PRP security. We present definitions for PRP security under non-adaptive chosen-plaintext attack (CPA1) [10] and SPRP (Strong PRP) security under non-adaptive chosen-ciphertext attack (CCA1) [17]:

Definition 1 (PRP Security). The PRP advantage of adversary A against FPE is defined as Adv_ ^{FPE}_{PRP}(A) = 2 • Pr[K←$;AO1(N，T，X) true]− 1 其中预言机 1(N,T,X)=EK,N,T(X)，为非适应性选择明文攻击下的查询预言机，要求攻击者在第一次查询预言机之前准备好所有问题，每次查询独立，且只允许查询加密预言机定义 2(SPRP 安全性). 攻击者 A 针对 FPE 的 SPRP 优势定义为 SPRP(A)=2 Pr[K←$; A^ 1(N,T,X); A^ 1

The definition of Adv_ ^{FPE}(A) is similar to Adv_ ^{FPE}_{SPRP}(A), but adversary A also has access to a decryption oracle 1^(-1)(N, T, Y) = D_K,N,T(Y). However, if Y is a result of a previous query _b(N, T, X), A cannot query 1^(-1)(N, T, X) true] - 1

1^(-1)(• , • , •), 1^(-1)(N, T, Y).

3.3 Attack Analysis Techniques

FPE security still faces numerous challenges. Current primary attack analysis techniques include:

1) **Small Domain Encryption Security:** Research by Durak [21], Bellare [20], Hoang [25][12] demonstrates that Feistel networks are more vulnerable to exhaustive or statistical analysis in small domains. When radix^{min}_{len} < 10^6, the actual security strength of Feistel-based FPE algorithms significantly weakens. This security degradation stems from small domain spaces making differential characteristics easier to capture, and round function output collision probabilities growing exponentially as domain size shrinks, causing actual security strength to fall far below the nominal strength provided by underlying block ciphers (e.g., AES-128). Attackers may recover plaintext by collecting few ciphertexts or conducting attacks with complexity far below exhaustive key search.

2) **Slide Attacks:** Biryukov et al. proposed slide attacks in 1999. At EUROCRYPT 2021, Amon et al. [14] constructed three slide attack methods against FPE schemes, where the cyclic-structure-based slide attack achieves optimal data and time complexities of (^{11/6}) and (^{17/6}), respectively. Attackers attempt to find plaintext/ciphertext pairs (X, C) and (X' , C') where X' is X encrypted through a few rounds and C' is C encrypted through the same few rounds. By finding such "slide pairs," attackers can transform attacks on the full r-round cipher into attacks on an (r-s)-round cipher (where s is the slide distance), dramatically reducing attack complexity. Attackers construct special tweak sequences (T_1, T_2, ···, T) to trigger round function internal state leakage. When tweak reuse exceeds 2^{32}, they can build plaintext-ciphertext correspondence databases to recover sensitive data with (√) complexity.

3) **Linear Cryptanalysis Threat:** By constructing approximate equations of the form a · L_0 b · R_0 a · L_r b · R_r = Δ, attackers [15] can exploit tweak-controlled input differential propagation paths. Due to limited round function input/output bit sizes, single-round linear approximations have large biases, and limited round numbers prevent accumulated biases from rapidly decaying to information-theoretically indistinguishable levels. This enables attackers to construct meaningful linear expressions and use the Piling-up Lemma to distinguish or guess the encryption process.

4 Algorithm Design

4.1 Symbols and Definitions

1) **Key K:** A symmetric key of 128 bits. Key K is used for underlying cryptographic operations CIPH_K and must be securely stored and protected.

2) **Radix:** Defines the character set size for plaintext string X. For example, radix = 10 for digits (10 characters total), radix = 26 for lowercase letters (26 characters total).

3) **Plaintext X, length n:** The original data to be encrypted, represented as a string X = X_1X_2···X consisting of n characters, where n [log_{radix}(10^6), 2(log_{radix}(2^{96}))].

4) **Tweak T:** A public parameter used to adjust plaintext encryption. The same key K and plaintext X produce different ciphertexts under different tweaks. FR-FPE' s tweak parameter length is at most 96 bits.

5) **Block Cipher Algorithm Identifier (CipherID) cid:** An index identifying the underlying block cipher algorithm selection, typically a fixed value (e.g., SM4 = 1, SM3-HMAC = 2, AES = 3).

6) **Standard Block Cipher Function CIPH_K(X):** Encrypts plaintext X using a standard block cipher algorithm and key K, such as SM4, SM3-HMAC, or AES.

7) **String-to-Integer Conversion Function NUM_{radix}(S):** Converts a string S with base radix to a non-negative integer.

8) **Binary Bit String-to-Integer Conversion Function NUM(S):** Converts a binary bit string S to a non-negative integer.

9) **Integer-to-String Conversion Function STR_{radix}(x):** Converts a non-negative integer x to a string of length m with base radix.

4.2 Algorithm Structure

[Figure 1: see original paper] FR-FPE Algorithm Encryption/Decryption Overall Architecture

The overall encryption/decryption framework of the Finite Radix Format-Preserving Encryption (FR-FPE) algorithm is shown in Figure 1. The algorithm splits data into left and right halves, combines cryptographically secure block cipher functions, initial vector P, and tweak parameter T, and processes them through multiple rounds of Feistel network iterations to ensure the encrypted data format remains identical to the plaintext. FR-FPE adopts the same fixed 10-round Feistel structure as FF1, supporting encryption of plaintexts up to 192 bits and tweak parameters up to 96 bits.

FR-FPE constructs the algorithm' s initial vector P from plaintext parameters, block cipher algorithm identifiers, and high-order bytes of the tweak parameter. After block cipher encryption (e.g., SM4), it participates in all Feistel rounds via CBC mode. Simultaneously, the algorithm XORs low-order bytes of the tweak parameter with the round number before concatenating with plaintext, then performs CBC encryption with initial vector P. Through CBC mode, the algorithm ensures that both initial vector P and the entire tweak parameter T participate in all Feistel encryption/decryption rounds, guaranteeing pseudorandom permutation properties and association with overall plaintext encryption, effectively resisting slide attacks and linear cryptanalysis.

4.3 Encryption Algorithm

The encryption algorithm Encrypt(K, X, T, radix, cid) → C operates as follows:

Algorithm 1: FR-FPE Encryption Algorithm

Input: Key K, plaintext string X of length n, algorithm identifier cid, tweak parameter T, radix
Output: Ciphertext string Y of length n

1. u ← n/2 , v ← n - u

2. A ← X[1 ···u], B ← X[u + 1 ···n]

3. P = [1]$_1$ [T_{len}]$_1$ [radix]$_3$ [u mod 256]$_1$ [n]$_1$ [cid]$_1$ [T_H]$_8$
4. F ← CIPH_K(P)
5. for i ← 0 to 9, do
6. Q ← ([T_L]$_4$ [i]$_4$) [NUM_{radix}(B)]$_{12}$
7. R ← CIPH_K(F Q)
8. y ← NUM(R)
9. if i is even then m ← u else m ← v
10. c ← (NUM_{radix}(A) + y) mod radix^m
11. C ← STR_{radix}^m(c)
12. A ← B, B ← C
13. end for
14. return A||B

Given key K, plaintext X, tweak parameter T, algorithm identifier cid, and radix, the algorithm' s main processes include:

1) **Plaintext and Tweak Splitting:**
Compute left and right half lengths as u = n/2 and v = n - u, where n is plaintext length. Split plaintext X into left part A = X[1 ···u] and right part B = X[u + 1 ···n]. Pad tweak parameter T with zeros to 96 bits and split into high 64-bit bytes T_H = T[1···64] and low 32-bit bytes T_L = T[65···96].

2) **Initial Vector Construction:**
Select T_{len}, radix, u, n, cid, high 64 bits of tweak parameter T_H, and constant terms, encoding them into fixed-length characters and concatenating into a 128-bit combined vector P:
P = [1]$_1$ [T_{len}]$_1$ [radix]$_3$ [u mod 256]$_1$ [n]$_1$ [cid]$_1$ [T_H]$_8$
where [1]$_1$ encodes constant 1 as a 1-byte character, [T_{len}]$_1$ is the 1-byte representation of tweak T' s byte length, [radix]$_3$ is the 3-byte representation of radix, [u mod 256]$_1$ is the single-byte representation of u modulo 256, [n]$_1$ is the single-byte length n, [cid]$_1$ is the 1-byte block cipher identifier, and [T_H]$_8$ is the high 8 bytes of the tweak parameter.

3) **Symmetric Encryption of Initial Vector:**
Encrypt the initial vector to obtain ciphertext F = CIPH_K(P), where CIPH_K(P) denotes encryption of P using block cipher CIPH with key

K, with the specific algorithm determined by cid.

4) **Execute 10 Feistel Rounds:** In round i:

   a) Construct round encryption plaintext Q by XORing low 4 bytes of tweak $[T\_L]_4$ with 4-byte round encoding $[i]_4$, then concatenating with 12-byte encoding of right plaintext B converted to integer: Q = ($[T\_L]_4$ $[i]_4$) $[NUM\_\{radix\}(B)]_{12}$.

   b) Compute CBC round ciphertext R: XOR initial vector ciphertext F with round plaintext Q, then encrypt using CIPH_K: R = CIPH_K(F Q).

   c) Convert R to integer y = NUM(R).

   d) Determine left input A string length m: if i is even, m = u, otherwise m = v.

   e) Compute round update data C for left input A: Convert radix-based string A to integer NUM_{radix}(A), compute integer c = (NUM_{radix}(A) + y) mod radix^m, then convert integer c to radix-based string C using STR_{radix}^m.

   f) Update round outputs: A = B, B = C.

5) **Output Ciphertext:** After 10 rounds, concatenate final A and B to obtain output ciphertext Y = A B.

4.4 Decryption Algorithm

The decryption algorithm Decrypt(K, X, T, radix, cid) → Y operates as follows:

Algorithm 2: FR-FPE Decryption Algorithm

Input: Key K, ciphertext string X of length n, algorithm identifier cid, tweak parameter T, radix
Output: Plaintext string Y

1. u ← n/2 , v ← n - u
2. A ← X[1 ···u], B ← X[u + 1 ···n]
3. P = $[1]_1$ $[T\_\{len\}]_1$ $[radix]_3$ $[u \bmod 256]_1$ $[n]_1$ $[cid]_1$ $[T\_H]_8$
4. F ← CIPH_K(P)
5. for i ← 9 to 0, do

6. Q ← ([T_L]$_4$ [i]$_4$) [NUM_{radix}(A)]$_{12}$
7. R ← CIPH_K(F Q)
8. y ← NUM(R)
9. if i is even then m ← u else m ← v
10. c ← (NUM_{radix}(B) - y) mod radix^m
11. C ← STR_{radix}^m(c)
12. B ← A, A ← C
13. end for
14. return A||B

Given key K, ciphertext X, tweak parameter T, algorithm identifier cid, and radix, the algorithm' s main processes include:

1) **Ciphertext and Tweak Splitting:**
Compute left and right half lengths as u = n/2 and v = n - u. Split ciphertext X into left part A = X[1 ···u] and right part B = X[u + 1 ··· n]. Pad tweak parameter T with zeros to 96 bits and split into high 64-bit bytes T_H = T[1···64] and low 32-bit bytes T_L = T[65···96].

2) **Initial Vector Construction:**
Construct the same 128-bit combined vector P as in encryption:
P = [1]$_1$ [T_{len}]$_1$ [radix]$_3$ [u mod 256]$_1$ [n]$_1$ [cid]$_1$ [T_H]$_8$
with identical field encoding.

3) **Symmetric Encryption of Initial Vector:**
Compute initial vector ciphertext F = CIPH_K(P) identically to encryption.

4) **Execute 10 Feistel Rounds in Reverse Order:** In round i:

   a) Construct round encryption plaintext Q by XORing low 4 bytes [T_L]$_4$ with 4-byte round encoding [i]$_4$, then concatenating with 12-byte encoding of left input A converted to integer: Q = ([T_L]$_4$ [i]$_4$) [NUM_{radix}(A)]$_{12}$.

   b) Compute round ciphertext R: XOR initial vector ciphertext F with round plaintext Q, then encrypt using CIPH_K: R = CIPH_K(F Q).

   c) Convert R to integer y = NUM(R).

d) Determine right input B string length m: if i is even, m = u, otherwise m = v.

e) Compute round update data C for right input B: Convert radix-based string A to integer NUM_{radix}(A), compute integer c = (NUM_{radix}(A) - y) mod radix^m, then convert integer c to radix-based string C using STR_{radix}^m.

f) Update round outputs: B = A, A = C.

5) **Output Plaintext:** After 10 rounds, concatenate final A and B to obtain output plaintext Y = A B.

4.5 Algorithm Correctness

For any valid key K, format N = (n, radix, cid) N, tweak T , and any plaintext X , let X' = E_K,N,T(X) be the encryption result. Then the decryption result D_K,N,T(X') must equal the original plaintext X, i.e.:

D_K,N,T(E_K,N,T(X)) = X

We prove this by tracking the decryption process state, showing that the state (A_0 ', B_0') after decryption completes matches the initial pre-encryption state (A_0, B_0) exactly.

Let X' = E_K,N,T(X). The decryption algorithm D_K,N,T(X') takes X' as input. Let X' = A_{10}' | B_{10} '. By definition of the encryption process, we have (A_{10}, B_{10}') = (A_{10}, B_{10}).

The decryption process computes the initial vector ciphertext F using the same K, N, T as encryption. Therefore, the F values in decryption and encryption are identical.

We now analyze the decryption loop from i = 9 to i = 0, tracking variables A and B. Let the state before processing round i be (A_{cur}, B_{cur}).

Consider decryption round i (i = 9, ···, 0):

State Initialization: When entering round i, assume the current state (A_{cur}, B_{cur}) = (A_{i+1}, B_{i+1}).

Rewriting Round Encryption Plaintext Input: The algorithm uses current A (i.e., A_{cur} = A_{i+1}) to compute Q_i. Recalling encryption' s state transition: (A_{i+1}, B_{i+1}) = (B_i, C_i). Thus, the current A in decryption (A_{i+1}) is actually B_i from before encryption round i.

Decryption Computation: Q_i' (A_{cur}) = ([T_L]$i)_4$ *([T_R]*i)$_4$ (NUM*{radix}(A*{cur})). Since A_{cur} = B_i, we have Q_i' (A_{cur}) = ([T_L]_i)$_4$ ([T_R]*i)$_4$ *(NUM*{radix}(B_i)), meaning decryption round i uses identical input as encryption round i.

Rewriting Round Ciphertext R Output: Decryption computes R_i' = CIPH_K(F Q_i' (A_{cur})). Since F is identical and Q_i' (A_{cur}) = Q_i(B_i), we necessarily obtain R_i' = R_i. Therefore, the converted integer is also identical: y_i' = NUM_{radix}(R_i') = NUM_{radix}(R_i) = y_i.

Decryption then computes c' = (NUM_{radix}(B_{cur}) - y_i') mod radix^{m_i}. The current B_{cur} is B_{i+1}. B_{i+1} = C_i = STR_{radix}^{m_i}((NUM_{radix}(A_i) + y_i) mod radix^{m_i}), so NUM_{radix}(B_{cur}) = NUM_{radix}(C_i) = (NUM_{radix}(A_i) + y_i) mod radix^{m_i}.

Substituting into c' computation:
c' = ((NUM_{radix}(A_i) + y_i) mod radix^{m_i} - y_i) mod radix^{m_i} = NUM_{radix}(A_i) mod radix^{m_i}

Here we use the modular arithmetic property: (a + b) mod N - b mod N = a mod N.

Since A_i itself is a string of length u or v, its value NUM_{radix}(A_i) must be less than radix^{m_i}. Therefore NUM_{radix}(A_i) mod radix^{m_i} = NUM_{radix}(A_i), and c' = NUM_{radix}(A_i). C = STR_{radix}^{m_i}(c') = STR_{radix}^{m_i}(NUM_{radix}(A_i)) = A_i.

State Update: Decryption executes B ← A, where A's current value is A_{cur} = A_{i+1} = B_i, so the new B becomes B_i. Decryption then executes A ← C, where C' s value is A_i, making the new A become A_i. Thus, after completing round i, the decryptor' s state becomes (A_i, B_i).

Induction Basis and Conclusion:
Basis: Decryption begins with input state (A_{10}, B_{10}') initialized to encryption' s output (A_{10}, B_{10}).

Induction: We proved that if the state entering round i' s loop body is (A_{i+1}, B_{i+1}), then the state after completing the round is (A_i, B_i).

Applying Induction: Starting at i = 9 with input state (A_{10}, B_{10}), after decryption round 9 the state becomes (A_9, B_9). Entering round 8 with state (A_9, B_9), after round 8 it becomes (A_8, B_8). This continues until i = 0, where entering round 0 with state (A_1, B_1) yields (A_0, B_0) after round 0.

Final Result: The decryption loop terminates after executing i = 0. The decryptor's final state (A_{cur}, B_{cur}) is (A_0, B_0). The decryption algorithm returns A_{cur} B_{cur} = A_0 B_0. Since (A_0, B_0) is the original plaintext X' s initial split, the returned result equals X.

Thus, we have proven that D_K,N,T(E_K,N,T(X)) = X holds for all valid inputs.

4.6 Algorithm Capability Comparison Analysis

Table 1 compares FR-FPE with NIST standards FF1, FF3-1, and the national standard research report SM4-TE-FPE.

FR-FPE Comparison with Classic Format-Preserving Encryption Algorithms

Feature	FR-FPE	FF1	FF3-1	TE-FPE
Plaintext & Tweak Parameters	Plaintext ≤ 192 bits, Tweak ≤ 96 bits	Plaintext ≤ 232 bits, Tweak length unlimited	Plaintext ≤ 192 bits, Tweak ≤ 56 bits	Plaintext ≤ 192 bits, Tweak ≤ 56 bits
Pseudorandom Permutation Security	Resists parameter slide attacks & linear analysis	Resists parameter slide attacks & linear analysis	Cannot resist parameter slide attacks & linear analysis	Resists parameter slide attacks & linear analysis
Tweak Parameter Encryption	T_H globally associated via CBC, T_L XORed with round number then globally associated	Global association encryption	Global association encryption	Encrypted T_L/T_R participate in odd/even rounds respectively
Block Cipher Invocations	11 times	≥ 20 times	9-23 times	Dynamic: 9, 11, 17, 23
Feistel Rounds	Fixed 10 rounds	Fixed 10 rounds	Dynamic 8, 10, 16, 22 rounds	Dynamic 8, 10, 16, 22 rounds

Regarding plaintext and tweak parameter lengths, FR-FPE' s plaintext length limit of 192 bits matches FF3-1 and TE-FPE. FR-FPE' s tweak parameter limit of 96 bits significantly exceeds FF3-1 and TE-FPE' s 56 bits, providing higher security for small-domain encryption.

In terms of security, FR-FPE, FF1, and TE-FPE all resist parameter slide attacks and linear analysis, with FR-FPE providing rigorous SPRP security proof. FF3-1 cannot resist these attacks and has been removed from standards.

For tweak parameter encryption, FR-FPE supports up to 96-bit tweaks, with both high-order and low-order bytes participating in all Feistel rounds (high-order via CBC mode, low-order XORed with round number before plaintext con-

catenation), ensuring pseudorandom permutation properties and global plaintext encryption association, comparable to FF1' s global tweak encryption. TE-FPE uses a shorter 56-bit tweak, encrypting and truncating it into T_L/T_R for odd/even round participation. This shorter length and truncation approach reduce ciphertext entropy, weakening PRP security strength.

Regarding Feistel rounds and block cipher invocations, FR-FPE uses the same fixed 10-round structure as FF1 but requires only 11 block cipher calls versus FF1' s minimum of 20—a 45% reduction (9 fewer calls)—substantially improving efficiency while maintaining equivalent security. TE-FPE uses dynamic round numbers (8, 10, 16, 22) with dynamic block cipher calls (9, 11, 17, 23), resulting in complex, non-fixed structure.

5 Security Analysis

This chapter analyzes FR-FPE' s security. First, we prove its Strong Pseudorandom Permutation (SPRP) security based on the Game-Hopping model, then discuss its resistance against three typical attacks.

5.1 Algorithm Pseudorandom Permutation Security

The block cipher function CIPH: × {0,1}^{128} → {0,1}^{128} uses standard block cipher algorithms, making CIPH_K a secure PRF. FR-FPE uses r = 10 rounds. Let q be the total number of queries adversary A makes to its oracle, m_{min} be the minimum of min(u, v) across all queried formats N = (n, ···) (where u = n/2 , v = n - u), and radix_{min} be the minimum radix among queries.

Theorem 1 (FR-FPE SPRP Security). For any SPRP adversary A making at most q total queries to its oracle, there exist adversaries B_0, B_1 such that:

Adv_{FR}-FPE(A) ≤ q·Adv_b^{SPRP}(B_0) + (10·q)·Adv_{CIPH}K^{PRF}(B_1) / 2^{128} + (q, 10, radix_{min}^m_{min})

where B_0' s running time is similar to A' s, making at most q queries to CIPH and its inverse. B_1' s running time is similar to A' s, making at most 10q queries to CIPH_K.

Let A be an SPRP adversary against E = FR-FPE. The proof proceeds through a sequence of games, starting from the real world (Game G_0) and ending in an ideal world where the oracle is a random permutation (Game G_4), computing the difference in adversary' s success probability between consecutive games.

Game G_0: This is the real SPRP experiment. A key K←$ is selected. Adversary A interacts with oracles _{1}(N, T, X) = E_K,N,T(X) and _{1}^(-1)(N, T, Y) = D_K,N,T(Y). Pr[A^G_{0} 1] = Pr[SPRP_{FR}-FPE^{real}(A) 1].

Game G_{1}: This game is identical to G_{0} except that computing F = CIPH_K(P) is replaced with F = f(P), where f: {0,1}^{128} →0,1^{128} is

a truly random function selected at the game' s start. Internal computations CIPH_K(F Q) still use the original key K and block cipher CIPH.

To compute the difference |Pr[A^G_{0} 1] - Pr[A^G_{1} 1]|, we construct a SPRP adversary B_{0} against CIPH. B_{0} receives an oracle _{CIPH} that is either CIPH_K and its inverse (B_{0}'s real world) or a random permutation and its inverse (B_{0}' s ideal world).

When A makes queries, B_{0} computes initial vector P and queries oracle _{CIPH} to obtain F*, then simulates the remainder of FR-FPE operations for A using F.* If _{CIPH} is CIPH_K, A is in Game G_{0}' s environment. If _{CIPH} is , A is in Game G_{1}' s environment. B_{0}' s advantage in distinguishing the oracle is |Pr[A^G_{0} 1] - Pr[A^G_{1} 1]|, so:

|Pr[A^G_{0} 1] - Pr[A^G_{1} 1]| Adv_{CIPH}K^{SPRP}(B_{0})

Additionally, assuming no collisions in F* values. Let q_F be the number of distinct P inputs queried (q_F q). The collision probability in _{CIPH} outputs is Pr[Coll_F] q_F$^{\{2\}/(2\cdot 2}${128}) q$^{\{2\}/2}${129}. We add this collision probability to the total bound:

|Pr[A^G_{0} 1] - Pr[A^G_{1} 1]| q $\cdot$ Adv_{CIPH}K^{SPRP}(B_{0}) + q$^{\{2\}/2}${129}

Game G_{2}: This game is identical to G_{1} except that computing R = CIPH_K(F Q), where F = f(P), is replaced with R = g(F Q), where g: {0,1}^{128} $\rightarrow$0,1^{128} is an independent truly random function.

To compute |Pr[A^G_{1} 1] - Pr[A^G_{2} 1]|, we construct a PRF adversary B_{1} against CIPH. B_{1} receives an oracle _{PRF} that is either CIPH_K or a truly random function g. B_{1} simulates Game G_{1} or G_{2} for A. When A queries, B_{1} computes F = f(P) using its own random function f, then computes F Q and queries _{PRF} to obtain R. *It completes the 10-round Feistel simulation for A using R.* If _{PRF} is CIPH_K, A is in Game G_{1}. If _{PRF} is g, A is in Game G_{2}. B_{1} makes 10 $\cdot$ q total queries to its oracle. B_{1}' s distinguishing advantage is |Pr[A^G_{1} 1] - Pr[A^G_{2} 1]|, so:

|Pr[A^G_{1} 1] - Pr[A^G_{2} 1]| (10 $\cdot$ q) $\cdot$ Adv_{CIPH}K^{PRF}(B_{1})

Game G_{3}: This game runs an r = 10-round unbalanced Feistel network using independent random functions f and g, replacing CIPH_K. The oracle is (N, T, X) = E_f,g,N,T^{ideal}(X) and its inverse. This game is functionally identical to Game G_{2}.

Pr[A^G_{3} 1] = Pr[A^G_{2} 1]

Game G_{4}: This game replaces the ideal Feistel permutation E_f,g,N,T^{ideal} in Game G_{3} with independent truly random permutations _N,T and their inverses _N,T^(-1) for each (N, T), defined as:

Pr[A^G_{4} 1] = Pr[SPRP_{FR}-FPE^{ideal}(A) 1]

|Pr[A^G_{3} 1] - Pr[A^G_{4} 1]| is the maximum advantage difference for an SPRP adversary distinguishing an ideal Feistel cipher (using random functions f, g) from a truly random permutation. Let this information-theoretic advantage be (q, 10, N_{domain}), where N_{domain} radix_{min}^m_{min}. Based on Patarin' s [23] results, for any distinguisher making q plaintext/ciphertext queries, the distinguishing advantage from an ideal tweakable block cipher is bounded by (q, 10, N) O(q/N^{10-2}), which is negligible for radix_{min}^m_{min} 100,000.

|Pr[A^G_{3} 1] - Pr[A^G_{4} 1]| (q, 10, radix_{min}^m_{min}) = negl(k)

Conclusion: Using the triangle inequality to combine bounds:

Adv_{FR}-FPE(A) = |Pr[A^G_{0} 1] - Pr[A^G_{4} 1]|
 |Pr[A^G_{0} 1] - Pr[A^G_{1} 1]| + |Pr[A^G_{1} 1] - Pr[A^G_{2} 1]|
+ |Pr[A^G_{2} 1] - Pr[A^G_{3} 1]| + |Pr[A^G_{3} 1] - Pr[A^G_{4} 1]|
 (q·Adv_{CIPH}K^{SPRP}(B_{0}) + q^{2}/^{2}{129}) + ((10·q)·Adv_{CIPH}K^{PRF}(B_{1}))
+ negl(k)
 q·Adv_{CIPH}K^{SPRP}(B_{0}) + (10·q)·Adv_{CIPH}K^{PRF}(B_{1})
+ negl(k)

If CIPH_K is a secure pseudorandom function, then for any PPT adversary A making q queries, the advantage Adv_{FR}-FPE^{PRF}-CCA1(A) is negligible. Therefore, FR-FPE is secure and its behavior is computationally indistinguishable from a truly random permutation.

5.2 Attack Resistance Capability

Current attacks on FPE algorithms mainly include three types: small-domain attacks, slide attacks, and linear attacks. The following analyzes FR-FPE' s resistance capabilities.

(1) Small-Domain Attack Resistance
FR-FPE resists small-domain attacks through its 10-round Feistel network and the requirement radix^m_{min} > 10^6. The condition radix^m_{min} > 10^6 (where m_{min} = min(u, v)) ensures a sufficiently large operation domain. For 10-round Feistel networks, Patarin' s Coefficients H technique shows that this domain size (greater than one million) yields negligible advantage in distinguishing the construction from a random permutation. Due to the radix^m_{min} > 10^6 constraint, classic small-domain attacks like exhaustive search or complete permutation table construction are infeasible.

(2) Slide Attack Resistance
Slide attacks attempt to find plaintext/ciphertext pairs (X, C) and (X' , C') where X' is X encrypted through a few rounds and C' is C encrypted through the

same few rounds. By finding such "slide pairs," attackers transform attacks on the full r-round cipher into attacks on an (r-s)-round cipher, where s is the slide distance. FR-FPE resists slide attacks by introducing pre-computed vectors and round counters in round encryption design. Specifically, round counter i is incorporated in Q_i computation: Q = ([T_L]*{4}* */i/*{4}) [NUM(B)]_{12}, then CBC mode computes round ciphertext R = CIPH_K(F Q), involving the pre-encrypted value F in all Feistel rounds. This strongly associates critical parameters throughout the encryption process, effectively resisting slide attacks.

(3) Linear Attack Resistance

Cryptographic algebraic properties involve high-degree nonlinear equations among plaintext, ciphertext, and key bits. Matsui [24] first proposed linear cryptanalysis in 1993, finding effective linear approximations: m[i_{1},⋯,i] c[j_{1},⋯,j_b] = K[k_{1},⋯,k_c]. For random permutations, this holds with probability 1/2. Beyne [15] exploited round tweak parameters alternating between two values to identify high-probability linear paths in Feistel ciphers, building statistical distinguishers to differentiate real ciphers from random permutations. Using this distinguisher, message recovery attacks probabilistically infer partial plaintext content by analyzing message component behavior under different tweaks, reducing FF3-1 message recovery attack data complexity to O(N^2.5).

FR-FPE simultaneously involves both high-order and low-order tweak bytes in all Feistel rounds (high-order via CBC mode, low-order XORed with round number before plaintext concatenation), ensuring pseudorandom permutation properties and association with overall plaintext encryption, effectively resisting linear cryptanalysis. FR-FPE' s underlying block cipher (e.g., SM4) inherently resists linear cryptanalysis through its S-boxes and diffusion layers, making high-bias linear paths difficult to find. With 10 Feistel rounds, the Piling-up theorem causes linear biases to decrease exponentially through multiple rounds.

6 Experiments and Performance Analysis

This paper compares FR-FPE' s performance against NIST standard FF1 and TE-FPE from the Chinese Cryptography Industry Standardization Technical Committee' s GM/Y 5007-2024 research report. The underlying block cipher is the national standard SM4, the character set uses radix = 10 digits, and the test platform is an Intel i5 2.3GHz processor.

Table 2 shows encryption operations per second for three algorithms across different plaintext lengths.

Encryption Operations per Second for Three Algorithms at Different Plaintext Lengths

Plaintext Length	FR-FPE(SM4)	TE-FPE(SM4)	FF1(SM4)
8 characters	79,105 ops/s	58,051 ops/s	53,836 ops/s

Plaintext Length	FR-FPE(SM4)	TE-FPE(SM4)	FF1(SM4)
16 characters	69,632 ops/s	57,634 ops/s	44,865 ops/s
32 characters	39,905 ops/s	46,660 ops/s	31,024 ops/s
64 characters	21,244 ops/s	25,357 ops/s	18,534 ops/s

[Figure 2: see original paper] Comparison of Encryption Operations per Second for Three Algorithms at Different Plaintext Lengths

Both FR-FPE and FF1 use 10-round Feistel structures, but FR-FPE optimizes initial vector construction and encryption, reducing block cipher invocations by 45%. Therefore, FR-FPE is faster than FF1 for all character lengths. TE-FPE uses variable-round Feistel structures: for short plaintexts, it uses 10+ rounds (slower); for plaintexts longer than 10 characters, it reduces to 8 rounds (faster but with reduced security). As shown in Figure 2, for short plaintext encryption (8 characters), FR-FPE achieves average performance improvements of 50.7% over FF1 and 28.6% over TE-FPE. For long plaintext encryption (>8 characters), FR-FPE is 23.3% faster than FF1 but 15.1% slower than TE-FPE.

Table 3 and Figure 3 show encryption throughput rates for three algorithms across different plaintext lengths.

Throughput Rates for Three Algorithms Encrypting Different Plaintext Lengths

Plaintext Length	FR-FPE(SM4)	TE-FPE(SM4)	FF1(SM4)
8 characters	2.76 Mbps	1.99 Mbps	1.89 Mbps
16 characters	3.62 Mbps	2.31 Mbps	2.28 Mbps
32 characters	4.28 Mbps	4.80 Mbps	3.36 Mbps
64 characters	4.97 Mbps	5.62 Mbps	4.24 Mbps

[Figure 3: see original paper] Comparison of Throughput Rates for Three Algorithms at Different Plaintext Lengths

Consistent with theoretical performance analysis, for short plaintexts (8 characters), FR-FPE' s throughput is 48.3% higher than TE-FPE and 53.1% higher than FF1 on average. For long plaintexts (>8 characters), FR-FPE' s throughput is 9.3% lower than TE-FPE but 21.7% higher than FF1 on average.

Conclusion

Addressing the challenges that existing FPE standards struggle to defend against slide and linear cryptanalysis attacks while suffering from low efficiency, this paper proposes the Finite Radix Format-Preserving Encryption (FR-FPE) algorithm, supporting encryption of plaintexts up to 192 bits and tweak parameters up to 96 bits. Through its CBC-mode-based parameter vector encryption mechanism and structured tweak parameter fully-associated encryption model,

FR-FPE effectively resists slide and linear cryptanalysis attacks. Its lightweight deterministic encryption structure significantly reduces underlying block cipher invocations. Based on the Game-Hopping model, we prove FR-FPE achieves Strong Pseudorandom Permutation (SPRP) security. Experiments demonstrate that under identical plaintext and tweak parameter lengths, FR-FPE achieves 41.56% higher encryption operations per second and 32.79% higher data throughput compared to FF1.

References

[1] National Bureau of Standards. FIPS PUB 74, Guidelines for Implementing and Using the NBS Data Encryption Standard, 1981.
[2] Black J, Rogaway P. Ciphers with arbitrary finite domains. In: Preneel B, ed. Proc. of the Topics in Cryptology CT-RSA 2002. LNCS 2271, San Jose: Springer-Verlag, 2002. 114-130.
[3] Dworkin M. 800-38G Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption[J]. 2013.
[4] ANSI X9.124-1-2023, Symmetric Key Cryptography For the Financial Services Industry –Format Preserving Encryption- Part 1: Definitions and Model.
[5] Jung-Keun Lee, Bonwook Koo, Dongyoung Roh, et al. Format-Preserving Encryption Algorithms Using Families of Tweakable Blockciphers[C] International Conference on Information Security and Cryptology. 2014.
[6] Jang W, Lee S Y. Partial image encryption using format-preserving encryption in image processing systems for Internet of things environment[J]. International Journal of Distributed Sensor Networks, 2020, 16(3):155014772091477.
[7] Dukyoung Kim, Hyunji Kim, Kyungbae Jang, Seyoung Yoon and Hwajeong Seo. Deep-Learning-Based Neural Distinguisher for Format-Preserving Encryption Schemes FF1 and FF3[J]. Electronics, 2024, Vol.13(7): 1196.
[8] Majeed, Mohammed Abdul; Sulaiman, Rossilawati; Shukur, Zarina. New Text Steganography Technique Based on Part-of-Speech Tagging and Format-Preserving Encryption[J]. KSII Transactions on Internet and Information Systems, 2024, Vol.18(1): 170-191.
[9] Heehwan Kim; Sungjune Park; Daeseon Choi. Secure and Reversible Face De-Identification With Format-Preserving Encryption[J]. IEEE Access, 2025, Vol.13: 116130-116142.
[10] 刘哲理, 贾春福, 李经纬. 保留格式加密技术研究 [J]. 软件学报, 2012, 23(1):19.
[11] Vidhya, S. Enhancing Cloud Security for Structured Data: An AES-GCM Based Format-Preserving Encryption Approach[J]. Communications in Computer and Information Science, 2025, Vol.2429: 196-205.
[12] Hoang, V.T., Miller, D., Trieu, N. (2019). Attacks only Get Better: How to Break FF3 on Large Domains. In: Ishai, Y., Rijmen, V. (eds) Advances in Cryptology -EUROCRYPT 2019. EUROCRYPT 2019. Lecture Notes in Computer Science(), vol 11477. Springer, Cham.
[13] Liskov M, Rivest RL, Wagner D. Tweakable block ciphers. In: Advances in Cryptology –CRYPTO 2002. LNCS 2442, Santa Barbara: Springer-Verlag,

2002. 31-46.
[14] Amon, O., Dunkelman, O., Keller, N., Ronen, E., Shamir, A. (2021). Three Third Generation Attacks on the Format Preserving Encryption Scheme FF3. In: Canteaut, A., Standaert, FX. (eds) Advances in Cryptology –EUROCRYPT 2021. EUROCRYPT 2021. Lecture Notes in Computer Science(), vol 12697. Springer, Cham.
[15] Beyne, T. (2021). Linear Cryptanalysis of FF3-1 and FEA. In: Malkin, T., Peikert, C. (eds) Advances in Cryptology –CRYPTO 2021. CRYPTO 2021. Lecture Notes in Computer Science(), vol 12825. Springer, Cham.
[16] Spies T. Format preserving encryption. Unpublished Voltage White Paper. 2008.
[17] Bellare M, Ristenpart T, Rogaway P, et al. Format-Preserving Encryption [C]. International Workshop on Selected Areas in Cryptography. Springer, Berlin, Heidelberg, 2009.
[18] Bellare M, Rogaway P, Spies T. The FFX Mode of Operation for Format-Preserving Encryption Draft 1.1[J]. Unpublished Nist Proposal, 2010, 136(9):633.
[19] 李经纬, 贾春福, 刘哲理, 等. 基于 k-分割 Feistel 网络的 FPE 方案 [J]. 通信学报, 2012, 33(4):7.
[20] Bellare, Mihir, et al. "Message-Recovery Attacks on Feistel-Based Format Preserving Encryption." Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery, 2016, pp. 444-455.
[21] Durak, F.B., Vaudenay, S. (2017). Breaking the FF3 Format-Preserving Encryption Standard over Small Domains. In: Katz, J., Shacham, H. (eds) Advances in Cryptology –CRYPTO 2017. CRYPTO 2017. Lecture Notes in Computer Science(), vol 10402. Springer, Cham.
[22] Tim Beyne. Linear Cryptanalysis of FF3-1 and FEA[C]//Advances in Cryptology –CRYPTO 2021. 2021.
[23] Patarin, J. (2004). Security of Random Feistel Schemes with 5 or More Rounds. In: Franklin, M. Advances in Cryptology –CRYPTO 2004. CRYPTO 2004. Lecture Notes in Computer Science, vol 3152. Springer, Berlin, Heidelberg.
[24] M. Matsui, Linear cryptanalysis method for DES cipher, in Advances in Cryptology—Eurocrypt' 93, ed. by T. Helleseth. LNCS, vol. 765 (Springer, Berlin, 1993), pp. 386–397.
[25] Hoang, Viet Tung, et al. "The Curse of Small Domains: New Attacks on Format-Preserving Encryption." Advances in Cryptology –CRYPTO 2018, vol. 10991, Springer, 2018, pp. 221-251.

(Corresponding author: Xiaofeng Wang, E-mail: xf_{wang}@nudt.edu.cn)

Note: Figure translations are in progress. See original paper for figures.