Classic Full Plaintext Recovery Attacks on Low Round Generalized Feistel Networks

Yubing Zhu[1,2], Jianhong Shi[a,1,2], Yunteng Yang[1,2] and Yonghui Yang[1,2]

[1] Henan Key Laboratory of Quantum Information and Cryptography, Zhengzhou, Henan 450000, China
[2] Synergetic Innovation Center of Quantum Information and Quantum Physics, University of Science and Technology of China, Hefei 230026, China

Abstract

The Generalized Feistel Network (GFN) underpins widely standardized block ciphers, yet its resistance to full plaintext recovery remains largely unexplored. This paper extends the full-plaintext attack framework for standard Feistel ciphers to multi-branch scenarios, mainly makes the following 3 research contributions:

(1) Developed classic full plaintext recovery attacks on d-round Type-I GFN ($d \geq 3$) under CPA with $d+1$ encryption queries and $2d$-round Type-I GFN under CCA with d decryption queries. Query complexity depends on d, increasing branches can enhance anti-interference capability.

(2) Developed classic full plaintext recovery attack on 2-round Type-II GFN ($d \geq 4$) under CPA with 3 encryption queries and 3-round Type-II GFN under CCA with 1 encryption plus 2 decryption queries. Query complexity is independent of d, increasing branches does not improve resistance.

(3) Finded that all attacks treat round functions as black boxes, confirming that the weakness resides in the GFN topology rather than specific round-function designs. Strengthening S-boxes or diffusion matrices cannot mitigate it, only increasing rounds beyond the security threshold provides effective defense.

Keywords: Generalized Feistel Networks, Low Round, Full plaintext recovery attack

1 Introduction

Generalized Feistel Networks (GFNs), introduced by Zheng, Matsumoto and Imai [ZMI90], extend the classical two-branch Feistel structure by partitioning the internal state into $d \geq 3$ equal-length branches. GFN has three canonical variants, **Type-I**, **Type-II** and **Type-III**, which differ in the number and placement of round functions per round. Soon afterwards, the Suzaki-Minematsu generalization [SM10] further expanded the design space by replacing the fixed cyclic shift with an arbitrary permutation over branches, spurring systematic searches for optimal diffusion and security trade-offs. Because of the inherent suitability for low-area and low-power hardware implementation, GFNs have been widely adopted in practical lightweight block cipher designs, prominent examples include the ISO/IEC 29192-2 standard CLEFIA [SSAM07], HIGHT [HSK+06], Piccolo [SSMM11],

Emails: truth74751210@163.com (Yubing Zhu), sjh@qiclab.cn (Jianhong Shi), yangyunteng2026@163.com (Yunteng Yang), yyh@qiclab.cn (Yonghui Yang)
[a]This author is the corresponding author of this article

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-07-24.

TWINE [SMMK13], LBlock [WZ11] and WARP [BBI+21], as well as in hash function constructions such as the ISO standard Lesamnta-LW [ISO16] and Simpira [GM16].

The structural security of GFNs has been extensively investigated from both provable and concrete perspectives. On the provable side, Zheng [ZMI90] established the first round-complexity thresholds for pseudorandom permutation (PRP) security, $(2d-1)$ rounds for Type-I and $(d+1)$ rounds for Type-II and Type-III. Hoang and Rogaway [HR10] subsequently raised the tolerable adversarial queries to the beyond-birthday-bound regime ($q \sim N^{1-\varepsilon}$). These bounds were substantially sharpened by Shen, Guo and Wang [SGW20], who derived improved strong PRP (SPRP) guarantees via refined coupling arguments, proved $2n$-bit security for tweakable GFN instantiations for the first time. More recently, Chen, Guo and Li [CGL+26] developed AutoEGFN, the first computer-aided proof tool capable of systematically determining the (S)PRP round counts for a broad family of EGFN variants, including Type-I/II, Nyberg's GFN and Suzaki-Minematsu's GFN, yielding tighter bounds than previously known manual proofs. On a complementary front, Derbez and Euler [DE24] demonstrated that many GFN permutations are cryptographically equivalent up to re-labelling of inputs and outputs, reducing the $k!^2$ even-odd permutations on $2k$ branches to only $k!$ distinct equivalence classes.

Alongside these theoretical guarantees, a rich body of concrete cryptanalytic work has systematically exposed structural vulnerabilities. Blondeau and Minier [BM15] unified impossible differential, integral and zero-correlation distinguishers against Type-II GFNs using a round-function matrix representation, revealing that the key-words participating in zero-correlation attacks form a subset of those in impossible differential attacks. Shi, Liu and Li [SLL24] advanced impossible differential cryptanalysis by introducing a Boolean-function-based method for the middle-round contradiction step, yielding a 32-round key-recovery attack on WARP. On the meet-in-the-middle (MitM) front, Dong and Mao [DM17] initiated attacks on 3-branch GFNs, later generalized by Deng [DJL18, DJZ19] to $d \geq 4$-branch Type-I and Type-III GFNs, with data complexity $2^{n/2}$ and memory $2^{(d-1)n/d}$. Automated MitM search tools [HDQ+24] have also been ported from SPN to GFN-based hash functions, improving preimage attacks on Simpira-b and collision attacks on the ISO standard Lesamnta-LW. Collectively, these results confirm that GFNs exhibit exploitable structural regularities even when their internal round functions remain cryptographically strong.

Beyond classical adversaries, the emergence of quantum computing has introduced qualitatively new threats that exploit the periodic structures inherent in GFN branch-wiring patterns. Dong, Li and Wang [DLW19] pioneered this line of research by constructing polynomial-time quantum distinguishers for (d^2-d+2)-round Type-I GFN and $4d$-round Type-II GFN, followed by Grover-meet-Simon key-recovery attacks that outperform generic quantum brute force by factors of $2^{(nd^2+nd)/4}$ and $2^{3nd^2/2}$ respectively. This work triggered a cascade of extensions, Ni [NIDI19] demonstrated a $(d+1)$-round quantum distinguisher for Type-III GFN and a 17-round distinguisher for the full CAST-256 cipher; Dong, Dong and Wang [DDW20] exploited the strong-weak separability of branch outputs to construct quantum distinguishers for 4F-function and 2F-function GFN structures; You [YQZ+20] obtained a $(2d-2)$-round distinguisher and $(3d-2)$-round key-recovery attack for SMS4-like Type-II variants. These results underscore that the diffusion patterns which make GFNs attractive for lightweight implementation simultaneously generate periodicities that are efficiently detectable by Simon's period-finding algorithm, rendering classical round-count security heuristics largely irrelevant in the post-quantum setting.

Despite this extensive body of work on distinguishing attacks and key recovery, investigations into *full plaintext recovery* over generalized Feistel networks remain notably underdeveloped. A distinguishing attack merely detects non-random behavior, whereas a full plaintext recovery attack reconstructs the complete target plaintext from its ciphertext, obliviouly is a strictly stronger notion with direct practical implications. The recent work

of Guo [GWJ+26] took a decisive step in this direction by showing that both classical and quantum distinguishing attacks on low-round standard Feistel constructions can be systematically upgraded to full plaintext recovery attacks with essentially the same query complexity. Their framework leverages differential collision chains (classical) and improved truncated boomerang differentials to convert distinguishing observations into complete plaintext reconstructions. However, this paradigm has thus far been confined to the standard two-branch setting, its extension to the structurally richer multi-branch GFN families has remained unexplored.

Our contributions. In this paper, we extend the full plaintext recovery framework from standard Feistel structure to Type-I and Type-II GFN, achieving the first complete plaintext extraction results for these widely deployed structures.

- **Type-I GFN (Section 3):** We present a deterministic classical attack that fully recovers the target plaintext of a d-round Type-I GFN under CPA using only $d+1$ encryption queries, therefore extend it to $2d$-round Type-I GFN under CCA using d decryption queries with a punctured decryption oracle. Notably, the query complexity scales linearly with d, reflecting the slow diffusion of the Type-I structure.
- **Type-II GFN (Section 4):** We develop analogous attacks for Type-II GFN, a 2-round Type-II GFN under CPA requiring merely 3 encryption queries and a 3-round Type-II GFN under CCA with 1 encryption plus 2 decryption queries. Remarkably, the query complexity is independent of the number of branches d, meaning that increasing d does not improve resistance against these attacks.
- **Practical validation (Section 5):** We concretely instantiate our attacks on reduced-round versions of two real-world ciphers, CAST-256 (a 4-branch Type-I GFN, broken at 4 rounds under CPA and 8 rounds under CCA) and CLEFIA (a 4-branch Type-II GFN standardized in ISO/IEC 29192-2). All simulated recoveries succeed with probability 1 and match theoretical predictions.
- **Structural implications:** All attacks treat the internal round functions as ideal black boxes, meaning the vulnerability resides in the GFN wiring topology rather than in any specific instantiation of the F-functions. Consequently, strengthening S-boxes or diffusion matrices cannot mitigate these attacks, only increasing the number of rounds beyond the security threshold provides effective defense.

The remainder of the paper is organized as follows. Section 2 introduces notations, defines the three GFN types, formalizes the CPA and CCA attack models with the punctured decryption oracle and reviews Guo's full plaintext recovery framework on the standard two-branch Feistel cipher. Section 3 presents deterministic attacks on Type-I GFN. Section 4 extends the methodology to Type-II GFN. Section 5 validates the attacks through simulations on CAST-256 and CLEFIA. Section 6 concludes with a summary and directions for future work.

2 Preliminaries

2.1 Notation

Let $\{0,1\}^n$ denote the set of n-bit strings and $\oplus$ denote bitwise XOR. We consider generalized Feistel networks with an number of branches $d \geq 3$ and $n \ (mod\ d) = 0$. Throughout the paper, $k \equiv 0, 1, 2, \ldots, d-1 \pmod d$ for arbitrary branch indices, $j \equiv 0, 2, 4, \ldots, d-2 \pmod d$ is used for even-indexed branches. We denote F as the round function, in particular, the subscript of F may consist of a constant and k with a certain

value added to or subtracted from it, we should claim here that the constant does not affect the modulo d operation on k. For example, $d + k - 2 = 2d - 2$ when $k = 0$.

A plaintext and its ciphertext are written as d-branch vectors $\boldsymbol{P} = (P_0, \ldots, P_{d-1}) \in \{0,1\}^n$ and $\boldsymbol{C} = (C_0, \ldots, C_{d-1}) \in \{0,1\}^n$, where each branch $P_k, C_k \in \{0,1\}^{n/d}$. The target plaintext is denoted as $\boldsymbol{P^0}$ with its ciphertext $\boldsymbol{C^0} = E_{[r]}(\boldsymbol{P^0})$. Intermediate values produced at attack step i are denoted by $\boldsymbol{P^i}$ and $\boldsymbol{C^i}$. The r-round encryption and decryption oracles are denoted by $E_{[r]}$ and $\mathring{D}^{\boldsymbol{C^0}}_{[r]}$ respectively.

Difference vectors are defined by $\Delta^i = \boldsymbol{C^i} \oplus \boldsymbol{C^{i-1}} = (\Delta^i_0, \ldots, \Delta^i_{d-1}) \in \{0,1\}^n$, these differences can be divided into d parts according to the number of branches and we use $\Delta_k \in \{0,1\}^{n/d}$ to denote the difference of one single branch. $\boldsymbol{\Delta}$ denotes a non-zero difference vector injected on odd-indexed branches.

2.2 Type-I, Type-II, Type-III GFN

The input of n bits is equally partitioned into d branches, denoted as $(X_0, X_1, \ldots, X_{d-1})$, where each branch is n/d bits.

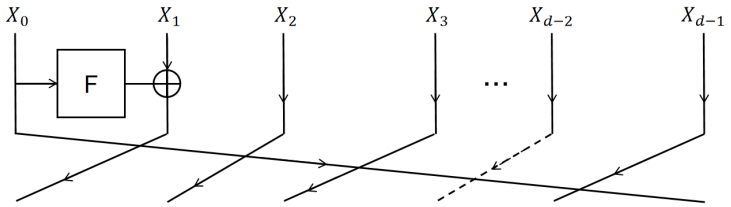


Figure 1: A single round transformation of Type-I GFN

In **Type-I GFN** [ZMI90], a single round function F is applied to one branch per round. Denoting the input state as $(X_0, X_1, \ldots, X_{d-1})$, one round transformation of Type-I GFN is defined as

$$(X_0, X_1, \ldots, X_{d-1}) \rightarrow (X_1 \oplus F(X_0), X_2, X_3, \ldots, X_{d-1}, X_0)$$

This construction employs only one F per round with a cyclic left shift by one position, resulting in slow diffusion—d rounds are required for every branch to influence every other branch. Representative ciphers include RC6 (with $d = 4$), CAST-256 and the inner structure of some lightweight designs.

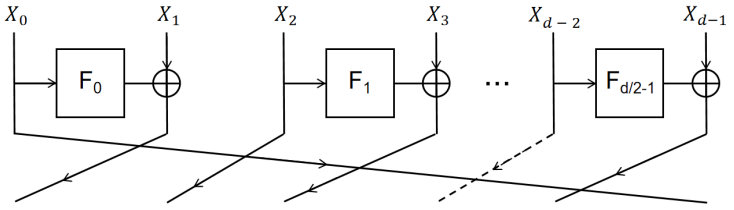


Figure 2: A single round transformation of Type-II GFN

As for **Type-II GFN** [ZMI90], the d branches are processed by $d/2$ independent round functions per round. For $l = 0, 1, \ldots, d/2 - 1$, each F_l takes its input from the even-index branch X_{2l} and its output is XORed into the next branch X_{2l+1}, followed by a cyclic left

shift. Single round transformation of Type-II GFN is defined as

$$(X_0, X_1, \ldots, X_{d-1})$$
$$\downarrow$$
$$(X_1 \oplus F_0(X_0), X_2, X_3 \oplus F_1(X_2), X_4, \ldots, X_{d-1} \oplus F_{d/2-1}(X_{d-2}), X_0)$$

Similarily, this construction applies a cyclic left shift after the F-function evaluations, resulting in a diffusion speed of d rounds. This structure has been widely adopted in practice, underpinning block ciphers such as CLEFIA, HIGHT, Piccolo and TWINE.

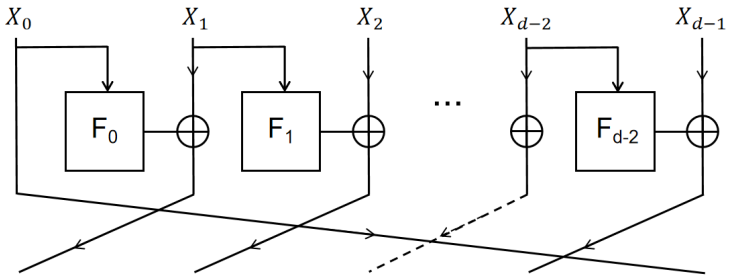


Figure 3: A single round transformation of Type-III GFN

Type-III GFN [ZMI90] employs $d-1$ round functions per round. For $l = 0, 1, \ldots, d-2$, each F_l takes its input from branch X_l and its output is XORed into the next branch X_{l+1}, followed by a cyclic left shift. One round of Type-III GFN is defined as

$$(X_0, X_1, \ldots, X_{d-1})$$
$$\downarrow$$
$$(X_1 \oplus F_0(X_0), X_2 \oplus F_1(X_1), X_3 \oplus F_2(X_2), \ldots, X_{d-1} \oplus F_{d-2}(X_{d-2}), X_0)$$

In Type-III, every non-final branch is updated in a single round. Although in practice Type-III appears less frequently, it still serve as an important comparator in structural cryptanalysis.

2.3 CPA and CCA models

All the attack schemes proposed in this paper solve for the target plaintext $\boldsymbol{P^0}$ under the condition of the ciphertexts are known, we mainly use two core attack models, classic chosen-plaintext attack (CPA) and classic chosen-ciphertext attack (CCA). These models are ranked from weak to strong in terms of the adversary's query capabilities, corresponding to diverse practical attack scenarios and security requirements respectively.

In the CPA model [KL14], adversaries can only adaptively submit arbitrary plaintexts to the encryption oracle E and obtain the corresponding ciphertexts, which is characterized by unidirectional query capability.

By contrast, in the CCA model [KL14], apart from encryption queries to E, adversaries are allowed to submit arbitrary ciphertexts (excluding the target ciphertext C^0) to the decryption oracle D to acquire decryption results, thus possessing bidirectional query capability.

To prevent trivial attacks where the adversary directly decrypts the challenge ciphertext, the CCA model employs a punctured decryption oracle $\mathring{D}^{\boldsymbol{C^0}}$ [KL14].

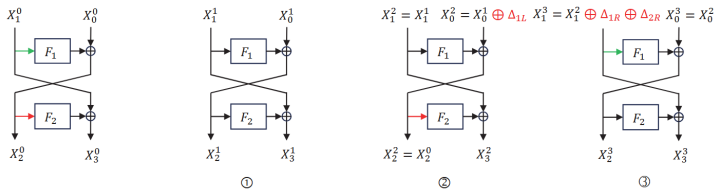
$$\mathring{D}^{\boldsymbol{C^0}}(x) = \begin{cases} \perp, & if\ x = \boldsymbol{C^0} \\ D(x), & otherwise \end{cases}$$

The symbol $\perp$ serves as an absorbing element, any operation involving $\perp$ (including XOR and truncation) strictly evaluates to $\perp$. In plaintext recovery experiments, $\mathring{D}^{\boldsymbol{C^0}}$ behaves identically to D except that it returns $\perp$ on input $\boldsymbol{C^0}$, thereby excluding trivial decryption of the target ciphertext.

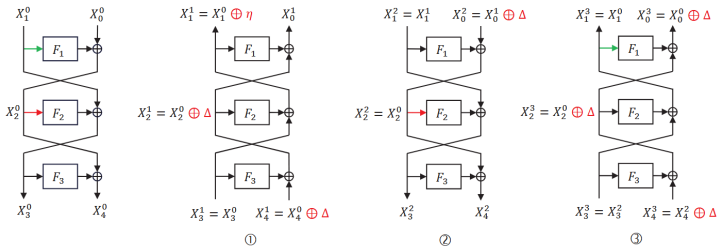
2.4 Full plaintext recovery attacks on standard Feistel structure

Recently, Guo [GWJ+26] showed that distinguishing attacks on low-round standard Feistel constructions can be systematically upgraded to full plaintext recovery with essentially the same query complexity. We review here the core techniques that underpin our attacks on GFN.

Classical recovery via differential collision chains. For the standard Feistel cipher with round transformation $R_i(X, Y) := (F_i(X) \oplus Y,\ X)$ and difference transformation $T_{(W,V)}(X, Y) := (X \oplus W,\ Y \oplus V)$, the classical attacks construct chains of collisions at round-function inputs via carefully chosen transformations T. We should claim here that X and Y are both n-bits long.



(1) The process of three classical encryption queries against $E_{[2]}$



(2) The process of $\mathring{D}^{C_0}_{[3]}$, $E_{[3]}$, and $\mathring{D}^{C_0}_{[3]}$

Figure 4: Classic full plaintext recovery attacks of standard Feistel structure [GWJ+26]

For 2-round Feistel under CPA, 3 encryption queries suffice

$$(X_1^0, X_0^0) = T_{(0^n,\Delta_{3L})} \circ T_{(\Delta_{1R}\oplus\Delta_{2R},0^n)} \circ T_{(0^n,\Delta_{1L})}(X_1^1, X_0^1)$$

where $T_{(0^n,\Delta_{1L})}$ creates an F_2-input collision, which cascades to an F_1-input collision via $T_{(\Delta_{1R}\oplus\Delta_{2R},0^n)}$.

For 3-round Feistel under CCA, 3 alternating D→E→D queries suffice

$$(X_1^0, X_0^0) = T_{(0^n,\Delta)} \circ \mathring{D}^{\boldsymbol{C^0}}_{[3]} \circ T_{(0^n,\Delta)} \circ E_{[3]} \circ T_{(0^n,\Delta)} \circ \mathring{D}^{\boldsymbol{C^0}}_{[3]} \circ T_{(0^n,\Delta)}(X_3^0, X_4^0)$$

The mechanism relies on the difference $\eta := F_2(X_2^0) \oplus F_2(X_2^0 \oplus \Delta)$, which is injected, cancelled, and re-injected across the three queries. Both attacks succeed with probability 1.

Improved truncated boomerang differential. The CCA attack pattern is an instantiation of the improved truncated boomerang differential [Wag99, GWJ+26], in this paper we mainly use this skill to realize the attack steps for Type-II GFN under the CCA model. For a cipher $E = E_2 \circ E_1$, the improved boomerang $(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$ with $\mathcal{D}_1 \subseteq \mathcal{D}_3$ holds with probability p if $D(C_3) \oplus D(C_4) \in \mathcal{D}_3$ whenever $P_1 \oplus P_2 \in \mathcal{D}_1$ and $C_1 \oplus C_3$, $C_2 \oplus C_4 \in \mathcal{D}_2$.

Theorem 1 (From boomerang differential to plaintext recovery)**.** *If E has an improved truncated boomerang differential $(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$ of probability p (with $0 \notin \mathcal{D}_2 \cup \mathcal{D}_3$), then 3 classical CCA queries yield a plaintext recovery attack with success probability $\geq p$: given $\boldsymbol{C^0}$, query $\mathring{D}^{\boldsymbol{C^0}}(\boldsymbol{C^0} \oplus \Delta_2) \to \boldsymbol{P^1}$, then $E(\boldsymbol{P^1} \oplus \Delta_1) \to \boldsymbol{C^2}$, then $\mathring{D}^{\boldsymbol{C^0}}(\boldsymbol{C^2} \oplus \Delta_2') \to \boldsymbol{P^3}$. The target satisfies $\boldsymbol{P^0} \in \{\boldsymbol{P^3} \oplus \Delta_3 \mid \Delta_3 \in \mathcal{D}_3\}$.*

3 Classic Full Plaintext Recovery Attack on Low-round Type-I GFN

In this section, we present the full plaintext recovery attacks on d-round Type-I GFN under the CPA model and $2d$-round Type-I GFN under the CCA model. Assume that the plaintext length is n, which is divided into d branches with each branch of length n/d. The plaintext space is $\boldsymbol{\mathcal{P}} = \{0,1\}^n$, the ciphertext space is $\boldsymbol{\mathcal{C}} = \{0,1\}^n$ and the intermediate state space $\boldsymbol{\mathcal{M}} = \{0,1\}^n$.

3.1 d-round Type-I GFN under CPA

Equally divided the plaintext into branches as $(P_0, P_1, \ldots, P_{d-1})$, so as the ciphertext $(C_0, C_1, \ldots, C_{d-1})$ obtained after d-round Type-I GFN encryption. The correspondence between the ciphertexts and plaintexts is given as

$$C_k = P_{k-1} \oplus F_{k-2}(C_{k-1}), \quad k \neq 2 \tag{1}$$

when $k = 2$, $C_2 = P_1 \oplus F_0(P_0)$.

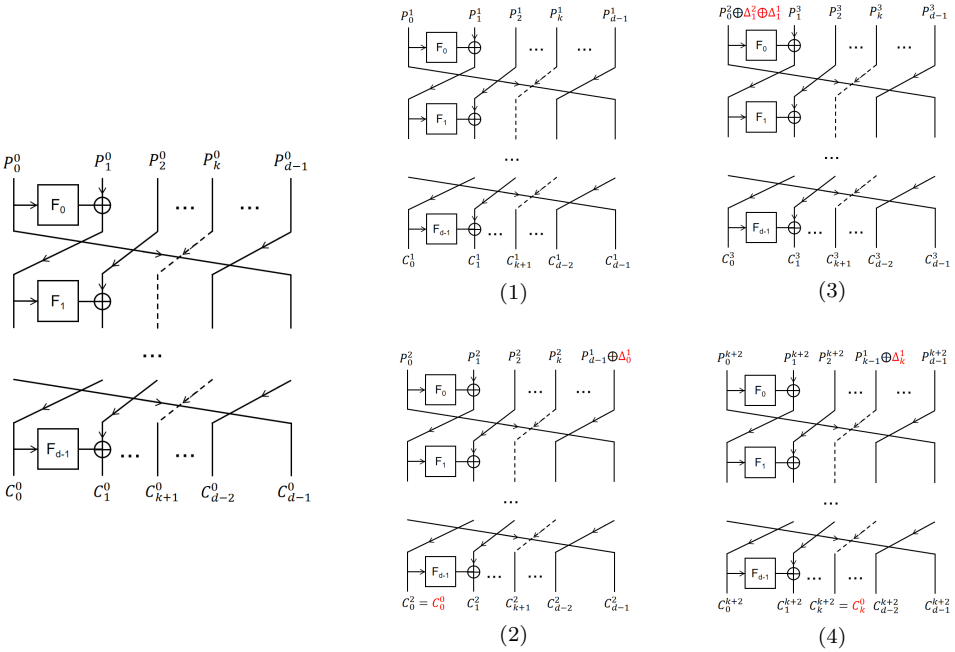


Figure 5: Classic plaintext recovery steps of d-round Type-I GFN under CPA

Attack steps. Given an arbitrary ciphertext $\boldsymbol{C^0} = E_{[d]}(\boldsymbol{P^0})$, we can recover the target plaintext $\boldsymbol{P^0}$ by $d+1$ encryption oracle $E_{[d]}$ queries with success probability 1 through the following steps. These attack steps are illustrated in Figure 5.

(1) Arbitrarily select a plaintext $\boldsymbol{P^1}$ different from the target plaintext, obtain the ciphertext $\boldsymbol{C^1}$ through the encryption oracle $E_{[d]}$, and calculate the difference $\Delta^1 = \boldsymbol{C^1} \oplus \boldsymbol{C^0} = (C_0^1, C_1^1, ..., C_{d-1}^1) \oplus (C_0^0, C_1^0, ..., C_{d-1}^0)$.

(2) Let $\boldsymbol{P^2} = (P_0^2, P_1^2, \ldots, P_{d-1}^2) = (P_0^1, P_1^1, \ldots, P_{d-1}^1 \oplus \Delta_0^1)$, the ciphertext $\boldsymbol{C^2}$ is obtained after d rounds of encryption, and compute the difference $\Delta^2 = \boldsymbol{C^2} \oplus \boldsymbol{C^1}$.

(3) Let $\boldsymbol{P^3} = (P_0^3, P_1^3, \ldots, P_{d-1}^3) = (P_0^2 \oplus \Delta_1^2 \oplus \Delta_1^1, P_1^2, \ldots, P_{d-1}^2)$. The ciphertext $\boldsymbol{C^3}$ is obtained after encryption, and then calculate the difference $\Delta^3 = \boldsymbol{C^3} \oplus \boldsymbol{C^2}$.

(4) Let $\boldsymbol{P^4} = (P_0^4, P_1^4, \ldots, P_{d-1}^4) = (P_0^1, P_1^1 \oplus \Delta_2^1, \ldots, P_{d-1}^1)$ and repeat for $d-2$ times while shifting backward the added difference terms each time. For example, next time we set $\boldsymbol{P^5} = (P_0^5, P_1^5, \ldots, P_{d-1}^5) = (P_0^1, P_1^1, P_2^1 \oplus \Delta_3^1, \ldots, P_{d-1}^1)$ and so on.

Proposition 1. *For any ciphertext $\boldsymbol{C^0} = E_{[3]}(\boldsymbol{P^0})$, we can always recover the target plaintext $\boldsymbol{P^0}$ with success probability 1 via $d+1$ encryption oracle $E_{[d]}$ queries*

$$\begin{aligned}\boldsymbol{P^0} =&(P_0^0, P_1^0, ..., P_k^0, ..., P_{d-1}^0)\\ =&(P_0^1 \oplus \Delta_1^2 \oplus \Delta_1^1, P_1^1 \oplus \Delta_2^3 \oplus \Delta_2^1, ..., P_k^1 \oplus C_{k+1}^{k+2} \oplus C_{k+1}^0, ..., P_{d-1}^1 \oplus C_0^{d+1} \oplus C_0^0)\\ =&(P_0^1 \oplus C_1^2 \oplus C_1^0, P_1^1 \oplus C_2^3 \oplus C_2^0, ..., P_k^1 \oplus C_{k+1}^{k+2} \oplus C_{k+1}^0, ..., P_{d-1}^1 \oplus C_0^{d+1} \oplus C_0^0)\end{aligned}$$

where $(P_0^1, P_1^1, ..., P_k^1, ..., P_{d-1}^1) = \boldsymbol{P^1}$ is an arbitrary plaintext we choose and

$$\begin{aligned}\Delta_k^i &= C_k^i \oplus C_k^{i-1}\\ \boldsymbol{P^2} &= (P_0^2, P_1^2, \ldots, P_{d-1}^2) = (P_0^1, P_1^1, \ldots, P_{d-1}^1 \oplus \Delta_0^1)\\ \boldsymbol{P^3} &= (P_0^3, P_1^3, \ldots, P_{d-1}^3) = (P_0^2 \oplus \Delta_1^2 \oplus \Delta_1^1, P_1^2, \ldots, P_{d-1}^2)\\ \boldsymbol{P^{k+2}} &= (P_0^1, ..., P_{k-1}^1 \oplus \Delta_k^1, ..., P_{d-1}^1)\end{aligned}$$

we should claim here that $i = 0, 1, 2, 3$. In addition, no modulo-d operation is required for superscripts, whereas subscripts shall be taken modulo d to guarantee the validity and authenticity of the corresponding plaintext branch.

Proof. In Step (2), we set $\boldsymbol{P^2} = (P_0^1, P_1^1, \ldots, P_{d-1}^1 \oplus \Delta_0^1)$, according to Equation 1, we can easily get

$$C_0^2 = P_{d-1}^1 \oplus \Delta_0^1 \oplus F_{d-2}(C_{d-1}^2)$$

Since C_{d-1} is determined by the plaintext $(P_0, P_1, ..., P_{d-2})$ except P_{d-1}, so that $C_{d-1}^2 = C_{d-1}^1$. Combining with above relation, we can learn

$$C_0^2 = C_0^1 \oplus \Delta_0^1 = C_0^0$$

Moreover, as $C_1 = P_0 \oplus F_{d-1}(C_0)$, we can obtain

$$C_1^2 \oplus C_1^0 = P_0^2 \oplus P_0^0$$

Therefore, we are able to recover $P_0^0 = P_0^1 \oplus \Delta_1^2 \oplus \Delta_1^1$. Next, in Step (3), we define $\boldsymbol{P^3} = (P_0^2 \oplus \Delta_1^2 \oplus \Delta_1^1, P_1^2, \ldots, P_{d-1}^2)$. Because $C_2 = P_1 \oplus F_0(P_0)$ and $P_0^3 = P_0^0$, hence

$$C_2^3 \oplus C_2^0 = P_1^3 \oplus P_1^0$$

Consequently, this allow us to recover $P_1^0 = P_1^1 \oplus \Delta_2^3 \oplus \Delta_2^1$. In Step (4), first we set $P_1^4 = P_1^1 \oplus \Delta_2^1$ and keep the rest unchanged comparing with $\boldsymbol{P^1}$, so we can get $C_2^4 = C_2^0$. Then, because $C_3 = P_2 \oplus F_1(C_2)$, so that

$$C_3^4 \oplus C_3^0 = P_2^4 \oplus P_2^0$$

This relation allows us to calculate $P_2^0 = P_2^4 \oplus C_3^4 \oplus C_3^0 = P_2^1 \oplus C_3^4 \oplus C_3^0$. Each time we let Let one component of the plaintext XOR corresponding components of the first difference Δ^1, then we can get $C_k^{k+2} = C_k^0$. So that

$$C_{k+1}^{k+2} \oplus C_{k+1}^0 = P_k^{k+2} \oplus P_k^0$$

Since all ciphertexts are known and $P_k^{k+2} = P_k^1$, we can obtain $P_k^0 = P_k^1 \oplus C_{k+1}^{k+2} \oplus C_{k+1}^0$ $(k \geq 2)$. So that after $d+1$ encryption oracle $E_{[2d]}$ queries, we can get the target plaintext $\boldsymbol{P^0} = (P_0^0, P_1^0, ..., P_k^0, ..., P_{d-1}^0) = (P_0^1 \oplus \Delta_1^2 \oplus \Delta_1^1, P_1^1 \oplus \Delta_2^3 \oplus \Delta_2^1, ..., P_k^1 \oplus C_{k+1}^{k+2} \oplus C_{k+1}^0, ..., P_{d-1}^1 \oplus C_0^{d+1} \oplus C_0^0)$ with success probability 1. □

3.2 2d-round Type-I GFN under CCA

Similarly, divide the plaintext into d equal-length branches as $(P_0, P_1, \ldots, P_{d-1})$, and the ciphertext $(C_0, C_1, \ldots, C_{d-1})$ is obtained after d-round Type-I GFN encryption. Specially, we define a intermediate state $\boldsymbol{M} = (M_0, M_1, ..., M_{d-1})$, it's obtained after d-round Type-I GFN and actually we don't known it's concrete values.

$$M_k = P_{k-1} \oplus F_{k-2}(M_{k-1}), \quad k \neq 2 \tag{2}$$

when $k = 2$, $M_2 = P_1 \oplus F_0(P_0)$. From the intermediate state $\boldsymbol{M}$, we can get the ciphertext $\boldsymbol{C}$

$$C_k = M_k \oplus F_{d+k-2}(C_{k-1}), \quad k \neq 2 \tag{3}$$

when $k = 2$, $C_2 = M_2 \oplus F_d(M_1)$. On account of the CCA model, we can query punctured decryption oracle, the inverse equation of Equation 2 and 3 are given as

$$M_k = C_k \oplus F_{d+k-2}(C_{k-1}), \quad k \neq 2$$

$$P_k = M_{k+1} \oplus F_{k-1}(M_k), \quad k \neq 2$$

when $k = 2$, $M_2 = C_2 \oplus F_d(M_1)$ and $P_2 = M_3 \oplus F_1(M_2)$.

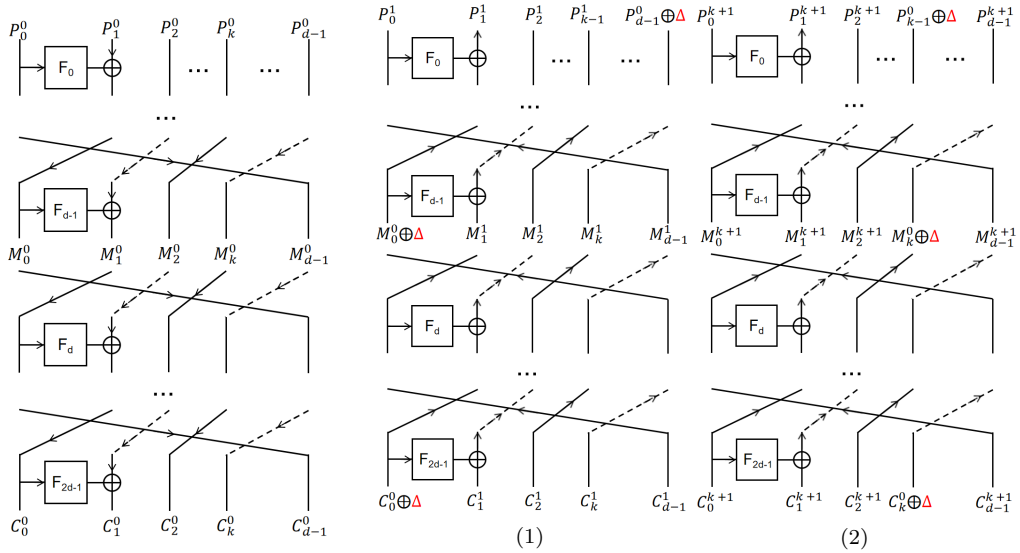


Figure 6: Classic plaintext recovery steps of $2d$-round Type-I GFN under CCA

Attack steps. Chosen an arbitrary ciphertext $\boldsymbol{C^0} = E_{[2d]}(\boldsymbol{P^0})$, we can recover the target plaintext $\boldsymbol{P^0}$ by d decryption oracle $\mathring{D}^{\boldsymbol{C^0}}_{[2d]}$ queries with success probability 1 through the following steps. These steps are illustrated in Figure 6

(1) Let $\boldsymbol{C^1} = (C_0^1, C_1^1, \ldots, C_{d-1}^1) = (C_0^0 \oplus \Delta, C_1^0, \ldots, C_{d-1}^0)$, query the punctured decryption oracle $\mathring{D}^{\boldsymbol{C^0}}_{[2d]}$ to obtain the plaintext $\boldsymbol{P^1}$. We must claim here that Δ is an arbitrary difference and $\Delta \neq 0$.

(2) Repeat Step (1) for d times, and shift backward the added difference terms each time. For example, next time we set $\boldsymbol{C^2} = (C_0^2, C_1^2, \ldots, C_{d-1}^2) = (C_0^0, C_1^0 \oplus \Delta, \ldots, C_{d-1}^0)$ and similarily caculate corresponding plaintext $\boldsymbol{P^2}$.

Proposition 2. *For any ciphertext $\boldsymbol{C^0} = E_{[3]}(\boldsymbol{P^0})$, we can always recover the target plaintext $\boldsymbol{P^0}$ with success probability 1 via d decryption oracle $\mathring{D}^{\boldsymbol{C^0}}_{[2d]}$ queries*

$$\begin{aligned}\boldsymbol{P^0} &= (P_0^0, P_1^0, ..., P_{k-1}^0, ..., P_{d-1}^0)\\ &= (P_0^2 \oplus \Delta, P_1^3 \oplus \Delta, ..., P_{k-1}^{k+1} \oplus \Delta, ..., P_{d-1}^1 \oplus \Delta)\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{P^{k+1}} &= \mathring{D}^{\boldsymbol{C^0}}_{[2d]}(\boldsymbol{C^{k+1}})\\ \boldsymbol{C^{k+1}} &= (C_0^0, C_1^0, ..., C_k^0 \oplus \Delta, ..., C_{d-1}^0)\\ \Delta &\neq 0\end{aligned}$$

It should be noted that when k is used as a superscript, it represents the attack step and no modulo d operation is required.

Proof. In Step (1), we set $\boldsymbol{C^1} = (C_0^0 \oplus \Delta, C_1^0, ..., C_{d-1}^0)$. Combining with $M_0 = C_0 \oplus F_{d-2}(C_{d-1})$, it's easy for us to know

$$M_0^1 = M_0^0 \oplus \Delta$$

Besides, since the ciphertext C_{d-2} and C_{d-1} are unchanged compare to $\boldsymbol{C^0}$, from $M_{d-1} = C_{d-1} \oplus F_{2d-3}(C_{d-2})$, we can also get

$$M_{d-1}^1 = M_{d-1}^0$$

Then, from $P_{d-1} = M_0 \oplus F_{d-2}(M_{d-1})$, so that

$$P_{d-1}^0 = P_{d-1}^1 \oplus \Delta$$

Similarily, in Step (2), we let $C_k^{k+1} = C_k^0 \oplus \Delta$ and keep the rest unchanged. By analogous arguments, we arrive at

$$M_k^{k+1} = M_k^0 \oplus \Delta$$

Combining with the relation $P_k = M_{k+1} \oplus F_{k-1}(M_k)$, it should note that the term inside the round function is P_0 when $k = 2$. Since the term inside the round function F keep unchanged due to the ciphertext we choose, so that

$$P_{k-1}^0 = P_{k-1}^{k+1} \oplus \Delta$$

So that after d decryption oracle $\mathring{D}^{\boldsymbol{C^0}}_{[2d]}$ queries, we can get the target plaintext $\boldsymbol{P^0} = (P_0^0, P_1^0, ..., P_{k-1}^0, ..., P_{d-1}^0) = (P_0^2 \oplus \Delta, P_1^3 \oplus \Delta, ..., P_{k-1}^{k+1} \oplus \Delta, ..., P_{d-1}^1 \oplus \Delta)$ with success probability 1. □

4 Classic Full Plaintext Recovery Attack on Low-round Type-II GFN

In this section, we present the full plaintext recovery attack on 2-round Type-II GFN under the CPA model, as well as the full plaintext recovery attack on 3-round Type-II GFN under the CCA model. Similiary, we let the plaintext length be n, which is divided into d branches with each branch of length n/d. The plaintext space is $\boldsymbol{\mathcal{P}} = \{0,1\}^n$, the ciphertext space is $\boldsymbol{\mathcal{C}} = \{0,1\}^n$.

4.1 2-round Type-II GFN under CPA

Suppose the plaintext is divided into $(P_0, P_1, \ldots, P_{d-1}) \in \{0,1\}^n$, so as the ciphertext $(C_0, C_1, \ldots, C_{d-1}) \in \{0,1\}^n$ is obtained after 2-round Type-II GFN. The correspondence between ciphertexts and plaintexts is given as

$$C_k = \begin{cases} P_{k+1} \oplus F_{k/2}(P_k), & k \text{ is even} \\ P_{k+1} \oplus F_{d/2+(k-1)/2}(P_k \oplus F_{(k-1)/2}(P_{k-1})), & k \text{ is odd} \end{cases} \tag{4}$$

The second formula can be further simplified to $C_k = P_{k+1} \oplus F_{d/2+(k-1)/2}(C_{k-1})$.

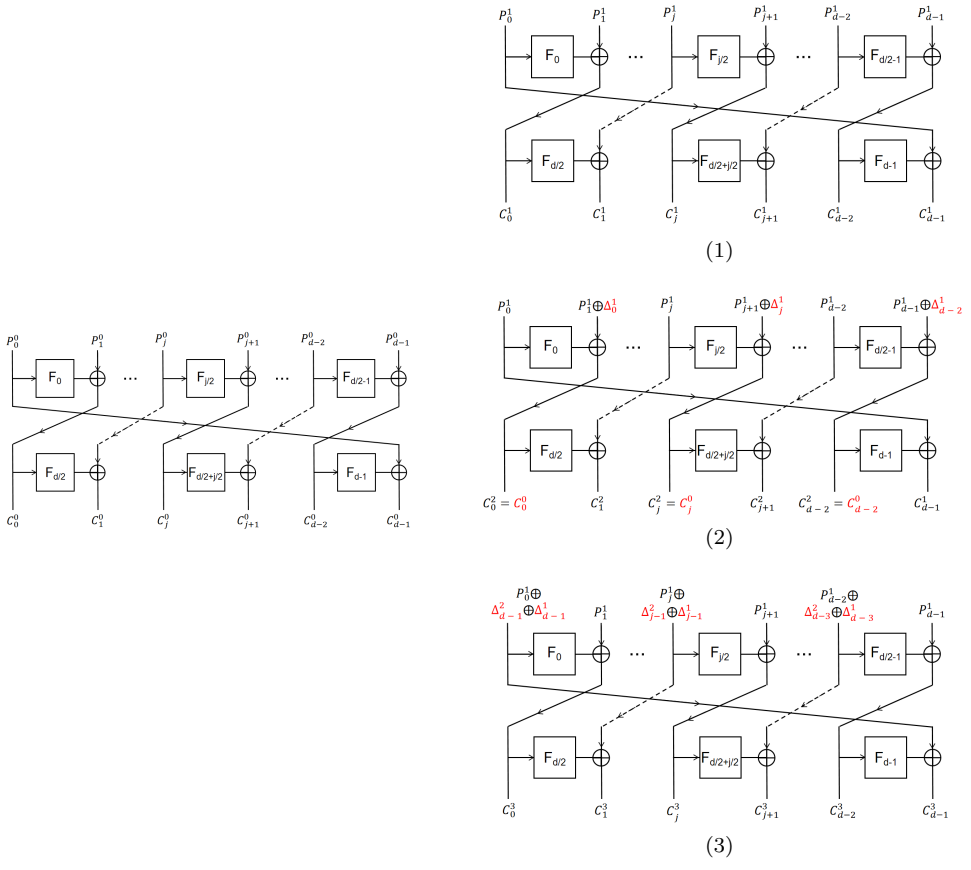


Figure 7: Classic full plaintext recovery steps of 2-round Type-II GFN under CPA

Attack steps. Given an arbitrary ciphertext $\boldsymbol{C}^0 = (C_0^0, C_1^0, \ldots, C_{d-1}^0) = E_{[2]}(\boldsymbol{P^0})$. Since $\boldsymbol{C}^0$ is known, we can fully recover the target plaintext $\boldsymbol{P}^0$ with success probability 1 via 3 encryption oracle E_2 queries. The attack steps are illustrated in Figure 7.

(1) Select a plaintext $\boldsymbol{P^1}$ arbitrarily, query the encryption oracle to obtain the corresponding ciphertext $\boldsymbol{C^1}$, and compute the difference $\Delta^1 = \boldsymbol{C^1} \oplus \boldsymbol{C^0} = (C_0^1, C_1^1, \ldots, C_{d-1}^1) \oplus (C_0^0, C_1^0, \ldots, C_{d-1}^0)$,

(2) Let $\boldsymbol{P^2} = (P_0^2, P_1^2, \ldots, P_{d-1}^2) = (P_0^1, P_1^1 \oplus \Delta_0^1, \ldots, P_j^1, P_{j+1}^1 \oplus \Delta_j^1, \ldots, P_{d-1}^1 \oplus \Delta_{d-2}^1)$. Then, query the encryption oracle to acquire the corresponding ciphertext $\boldsymbol{C^2}$, and calculate the difference $\Delta^2 = \boldsymbol{C^2} \oplus \boldsymbol{C^1}$

(3) Let $\boldsymbol{P^3} = (P_0^3, P_1^3, \ldots, P_j^3, P_{j+1}^3, \ldots, P_{d-2}^3, P_{d-1}^3) = (P_0^2 \oplus \Delta_{d-1}^2 \oplus \Delta_{d-1}^1, P_1^2, \ldots, P_j^2 \oplus \Delta_{j-1}^2 \oplus \Delta_{j-1}^1, P_{j+1}^2, \ldots, P_{d-2}^2 \oplus \Delta_{d-3}^2 \oplus \Delta_{d-3}^1, P_{d-1}^2)$. Then, query the encryption oracle to obtain the corresponding ciphertext and calculate $\Delta^3 = \boldsymbol{C^3} \oplus \boldsymbol{C^2}$.

Proposition 3. *For any ciphertext $\boldsymbol{C^0} = E_{[3]}(\boldsymbol{P^0})$, we can always recover the target plaintext $\boldsymbol{P^0}$ with success probability 1 via 3 encryption oracle $E_{[2]}$ queries*

$$\begin{aligned}\boldsymbol{P^0} &= (P_0^0, P_1^0, ..., P_j^0, P_{j+1}^0, ...) \\ &= (P_0^1 \oplus \Delta_{d-1}^2 \oplus \Delta_{d-1}^1, P_1^1 \oplus \Delta_0^3 \oplus \Delta_0^1, ..., P_j^1 \oplus \Delta_{j-1}^2 \oplus \Delta_{j-1}^1, P_{j+1}^1 \oplus \Delta_j^3 \oplus \Delta_j^1, ...)\end{aligned}$$

where $\boldsymbol{P^1} = (P_0^1, P_1^1, ..., P_{d-1}^1)$ is an arbitrary plaintext we choose and

$$\begin{aligned}\Delta_k^i &= C_k^i \oplus C_k^{i-1} \\ \boldsymbol{P^2} &= (P_0^1, P_1^1 \oplus \Delta_0^1, \ldots, P_j^1, P_{j+1}^1 \oplus \Delta_j^1, \ldots, P_{d-1}^1 \oplus \Delta_{d-2}^1) \\ \boldsymbol{P^3} &= (P_0^2 \oplus \Delta_{d-1}^2 \oplus \Delta_{d-1}^1, P_1^2, \ldots, P_j^2 \oplus \Delta_{j-1}^2 \oplus \Delta_{j-1}^1, P_{j+1}^2, \ldots, P_{d-1}^2)\end{aligned}$$

Proof. In Step (2), we set all odd-position plaintexts satisfying $P_{j+1}^2 = P_{j+1}^1 \oplus \Delta_j^1$, while keeping all even-indexed plaintexts unchanged. Substituting into Equation 4, we can derive that $C_j^2 = P_{j+1}^1 \oplus \Delta_j^1 \oplus F_{j/2}(P_j^1)$. Furthermore, we can deduce that $C_j^2 \oplus C_j^1 = \Delta_j^1$. Moreover, since $C_j^1 \oplus C_j^0 = \Delta_j^1$. Thus, we can obtain

$$C_j^2 = C_j^0$$

Subsequently, in Steps (3), on the basis of the plaintexts obtained in Step (2), we set $P_j^3 = P_j^2 \oplus \Delta_{j-1}^2 \oplus \Delta_{j-1}^1$, and keep the rest unchanged. Since $C_{j-1} = P_j \oplus F_{d/2+(j-2)/2}(C_{j-2})$, j is an even number, $j-1$ is definitely odd and $j-2$ is also even. Hence, $F_{d/2+(j-2)/2}(C_{j-2}^2) = F_{d/2+(j-2)/2}(C_{j-2}^0)$. Then, we can derive the relation $C_{j-1}^2 \oplus C_{j-1}^0 = P_j^2 \oplus P_j^0$. Moreover, since $C_{j-1}^2 \oplus C_{j-1}^0 = \Delta_{j-1}^2 \oplus \Delta_{j-1}^1$, we have

$$P_j^0 = P_j^2 \oplus \Delta_{j-1}^2 \oplus \Delta_{j-1}^1 = P_j^3$$

According to $C_j = P_{j+1} \oplus F_{j/2}(P_j)$ and $P_j^0 = P_j^3$, we can obtain $P_{j+1}^0 \oplus P_{j+1}^3 = C_j^3 \oplus C_j^0 = C_j^3 \oplus C_j^2 = \Delta_j^3$, so

$$P_{j+1}^0 = P_{j+1}^3 \oplus \Delta_j^3$$

So that after 3 encryption oracle $E_{[2]}$ queries, we can get the target plaintext $\boldsymbol{P^0} = (P_0^1 \oplus \Delta_{d-1}^2 \oplus \Delta_{d-1}^1, P_1^1 \oplus \Delta_0^1 \oplus \Delta_0^3, ..., P_j^1 \oplus \Delta_{j-1}^2 \oplus \Delta_{j-1}^1, P_{j+1}^1 \oplus \Delta_j^1 \oplus \Delta_j^3, ..., P_{d-1}^1 \oplus \Delta_{d-2}^1 \oplus \Delta_{d-2}^3)$ with success probability 1. □

4.2 3-round Type-II GFN under CCA

Suppose the plaintext is divided into $(P_0, P_1, \ldots, P_{d-1})$, the ciphertext $(C_0, C_1, \ldots, C_{d-1})$ and the intermediate state $m_{j/2}$ are obtained after 3-round Type-II GFN. $m_{j/2}$ is given as

$$m_{j/2} = P_{j+1} \oplus F_{j/2}(P_j) \tag{5}$$

The corresponding ciphertext can be expressed as

$$C_k = \begin{cases} P_{k+2} \oplus F_{(d+k)/2}(m_{k/2}), & k \ is \ even \\ m_{(k+1)/2} \oplus F_{d+(k-1)/2}(P_{k+1} \oplus F_{(d+k-1)/2}(m_{(k-1)/2})), & k \ is \ odd \end{cases} \tag{6}$$

Similarly, when k is odd, we can obtain the relation $C_k = m_{(k+1)/2} \oplus F_{d+(k-1)/2}(C_{k-1})$. If we possess the ciphertext and can query 3-round decryption oracle $\mathring{D}^{\mathbf{C^0}}_{[3]}$, the intermediate state can also be expressed as

$$m_{j/2} = C_{j-1} \oplus F_{d+j/2-1}(C_{j-2})$$

The corresponding plaintext can be expressed as

$$P_k = \begin{cases} C_{k-2} \oplus F_{d/2+(k-2)/2}(m_{(k-2)/2}), & k \ is \ even \\ m_{(k-1)/2} \oplus F_{(k-1)/2}(C_{k-3} \oplus F_{d/2+(k-3)/2}(m_{(k-3)/2})), & k \ is \ odd \end{cases}$$

when k is odd, we can easily deduce that $P_k = m_{(k-1)/2} \oplus F_{(k-1)/2}(P_{k-1})$.

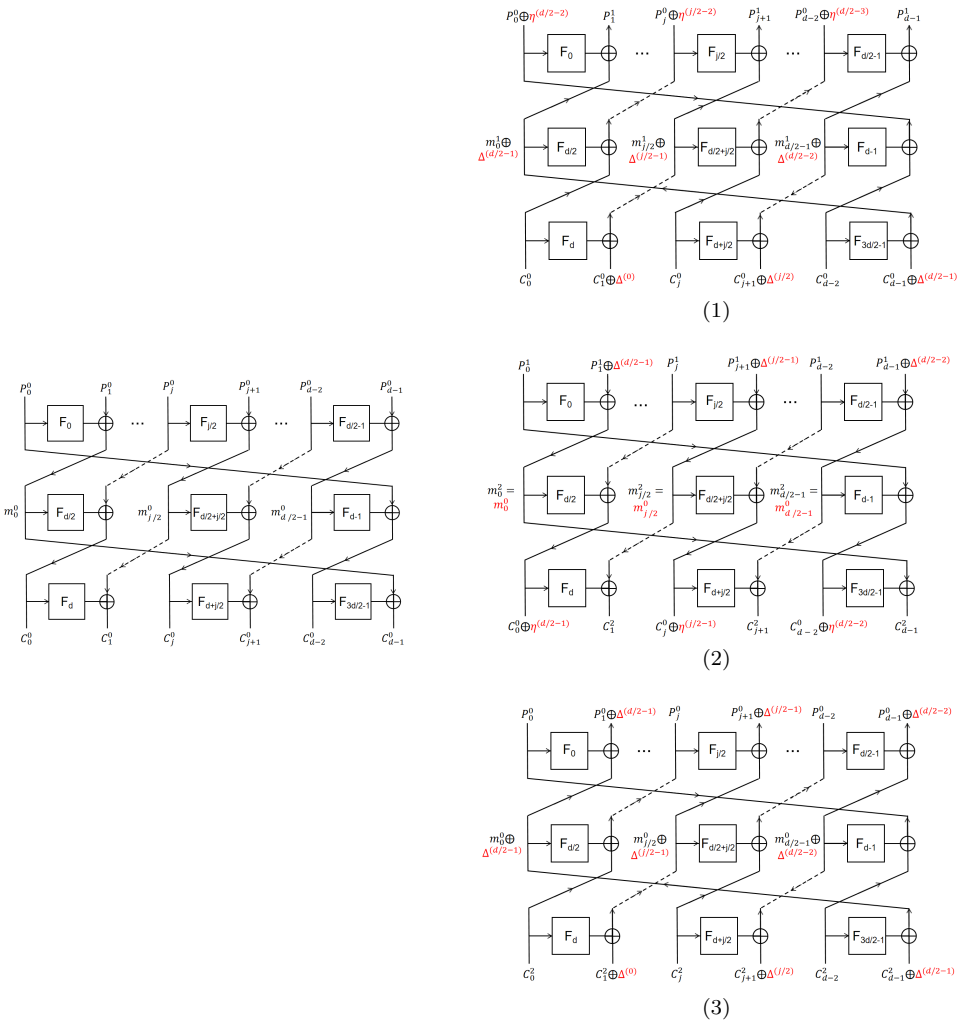


Figure 8: Classic full plaintext recovery steps of 3-round Type-II GFN under CCA

Attack steps. Choose a ciphertext $\boldsymbol{C^0} = E_{[3]}(\boldsymbol{P^0})$ arbitrarily, the target plaintext $\boldsymbol{P^0}$ can be fully recovered via 1 encryption oracle $E_{[3]}$ query and 2 decryption oracle $\mathring{D}_{[3]}^{\boldsymbol{C^0}}$ queries with success probability 1. The attack steps are illustrated in Figure 8.

(1) Let the ciphertext $\boldsymbol{C^1} = (C_0^1, C_1^1, \ldots, C_{d-1}^1) = (C_0^0, C_1^0 \oplus \Delta^{(0)}, \ldots, C_j^0, C_{j+1}^0 \oplus \Delta^{(j/2)}, \ldots, C_{d-1}^0 \oplus \Delta^{(d/2-1)})$ and query the decryption oracle $\mathring{D}_{[3]}^{\boldsymbol{C^0}}$ to obtain the plaintext $\boldsymbol{P^1}$. It should be noted that $\boldsymbol{\Delta} = (\Delta^{(0)}, \Delta^{(1)}, \ldots, \Delta^{(d/2-1)}) \neq \boldsymbol{0}$.

(2) Let the plaintext $\boldsymbol{P^2} = (P_0^2, P_1^2, \ldots, P_{d-1}^2) = (P_0^1, P_1^1 \oplus \Delta^{(d/2-1)}, \ldots, P_j^1, P_{j+1}^1 \oplus \Delta^{(j/2-1)}, \ldots, P_{d-1}^1 \oplus \Delta^{(d/2-2)})$. Then, query the encryption oracle $E_{[3]}$ to obtain the ciphertext $\boldsymbol{C^2}$.

(3) Let the ciphertext $\boldsymbol{C^3} = (C_0^3, C_1^3, \ldots, C_{d-1}^3) = (C_0^2, C_1^2 \oplus \Delta^{(0)}, \ldots, C_j^2, C_{j+1}^2 \oplus \Delta^{(j/2)}, \ldots, C_{d-1}^2 \oplus \Delta^{(d/2-1)})$. Query the decryption oracle to obtain the plaintext $\boldsymbol{P^3}$.

Proposition 4. *For any ciphertext $\boldsymbol{C^0} = E_{[3]}(\boldsymbol{P^0})$, we can always recover the target plaintext $\boldsymbol{P^0}$ with success probability 1 via* 1 *encryption oracle $E_{[3]}$ query and* 2 *decryption oracle $\mathring{D}_{[3]}^{\boldsymbol{C^0}}$ queries*

$$\begin{aligned}\boldsymbol{P^0} &= (P_0^0, P_1^0, \ldots, P_{d-1}^0)\\ &= (P_0^3, P_1^3 \oplus \Delta^{(d/2-1)}, \ldots, P_j^3, P_{j+1}^3 \oplus \Delta^{(j/2-1)}, \ldots, P_{d-1}^3 \oplus \Delta^{(d/2-2)})\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{C^1} &= (C_0^0, C_1^0 \oplus \Delta^{(0)}, \ldots, C_j^0, C_{j+1}^0 \oplus \Delta^{(j/2)}, \ldots, C_{d-1}^0 \oplus \Delta^{(d/2-1)})\\ \boldsymbol{P^1} &= \mathring{D}_{[3]}^{\boldsymbol{C^0}}(\boldsymbol{C^1})\\ \boldsymbol{P^2} &= (P_0^1, P_1^1 \oplus \Delta^{(d/2-1)}, \ldots, P_j^1, P_{j+1}^1 \oplus \Delta^{(j/2-1)}, \ldots, P_{d-1}^1 \oplus \Delta^{(d/2-2)})\\ \boldsymbol{C^2} &= E_{[3]}(\boldsymbol{P^2})\\ \boldsymbol{C^3} &= (C_0^2, C_1^2 \oplus \Delta^{(0)}, \ldots, C_j^2, C_{j+1}^2 \oplus \Delta^{(j/2)}, \ldots, C_{d-1}^2 \oplus \Delta^{(d/2-1)})\end{aligned}$$

Proof. After adding differences $\boldsymbol{\Delta}$ according to the rule introduced in Step (1), combining with $m_{j/2+1} = C_{j+1} \oplus F_{d+j/2}(C_j)$, we can deduce that

$$m_{j/2+1}^1 \oplus m_{j/2+1}^0 = C_{j+1}^1 \oplus C_{j+1}^0$$

Since $C_{j+1}^1 \oplus C_{j+1}^0 = \Delta^{(j/2)}$, we have

$$m_{j/2}^1 = m_{j/2}^0 \oplus \Delta^{(j/2-1)}$$

Moreover, from $P_{j+2} = C_j \oplus F_{d/2+j/2}(m_{j/2})$, we can obtain

$$P_{j+2}^1 \oplus P_{j+2}^0 = F_{d/2+j/2}(m_{j/2}^0 \oplus \Delta^{(j/2-1)}) \oplus F_{d/2+j/2}(m_{j/2}^0)$$

where $\eta^{(j/2-1)} = F_{d/2+j/2}(m_{j/2}^0 \oplus \Delta^{(j/2-1)}) \oplus F_{d/2+j/2}(m_{j/2}^0)$.

In Step (2), we set the plaintext

$$P^2 = (P_0^1, P_1^1 \oplus \Delta^{(d/2-1)}, \ldots, P_j^1, P_{j+1}^1 \oplus \Delta^{(j/2-1)}, \ldots, P_{d-1}^1 \oplus \Delta^{(d/2-2)})$$

Combining with Equation 5, we can get

$$m_{j/2}^2 = m_{j/2}^1 \oplus \Delta^{(j/2-1)}$$

Obviously, it's known as

$$m_{j/2}^2 = m_{j/2}^0$$

What's more, from $C_j = P_{j+2} \oplus F_{(d+j)/2}(m_{j/2})$, we can deduce that

$$C_j^2 = C_j^0 \oplus \eta^{(j/2-1)}$$

Subsequently, in Step (3) we set the ciphertext

$$\boldsymbol{C^3} = (C_0^2, C_1^2 \oplus \Delta^{(0)}, \ldots, C_j^2, C_{j+1}^2 \oplus \Delta^{(j/2)}, \ldots, C_{d-1}^2 \oplus \Delta^{(d/2-1)})$$

According to $m_{j/2-1} = C_{j-3} \oplus F_{d+j/2-2}(C_{j-4})$, we can get

$$m_{j/2-1}^3 = m_{j/2-1}^0 \oplus \Delta^{(j/2-2)}$$

Combining with $P_j^3 = C_{j-2}^3 \oplus F_{d/2+j/2-1}(m_{j/2-1}^3)$ and $C_{j-2}^3 = C_{j-2}^2 = C_{j-2}^0 \oplus \eta^{j/2-2}$, we have

$$\begin{aligned} P_j^3 &= C_{j-2}^2 \oplus F_{d/2+j/2-1}(m_{j/2-1}^0 \oplus \Delta^{(j/2-2)}) \\ &= C_{j-2}^0 \oplus \eta^{(j/2-2)} \oplus F_{d/2+j/2-1}(m_{j/2-1}^0 \oplus \Delta^{(j/2-2)}) \end{aligned}$$

Since $\eta^{(j/2-2)} = F_{d/2+j/2-1}(m_{j/2-1}^0 \oplus \Delta^{(j/2-2)}) \oplus F_{d/2+j/2-1}(m_{j/2-1}^0)$, we can obtain

$$P_j^3 = C_{j-2}^0 \oplus F_{d/2+j/2-1}(m_{j/2-1}^0) = P_j^0$$

Moreover, from $P_{j+1} = m_{j/2} \oplus F_{j/2}(P_j)$, we can obtain

$$P_{j+1}^3 = P_{j+1}^0 \oplus \Delta^{(j/2-1)}$$

As long as the difference $\boldsymbol{\Delta}$ is non-zero, after 1 encryption query and 2 decryption queries, we can always obtain the target plaintext $\boldsymbol{P^0} = (P_0^3, P_1^3 \oplus \Delta^{(d/2-1)}, ..., P_j^3, P_{j+1}^3 \oplus \Delta^{(j/2-1)}, ..., P_{d-1}^3 \oplus \Delta^{(d/2-2)})$ with probability 1. □

5 Applications to Practical GFN's Cryptographic Schemes

From section 3 and 4, it is easy to find that our attack schemes regard the round function as a black box. The key to this full plaintext recovery attack lies in that after the target plaintext is encrypted, for the ciphertexts of specific branches, there still exist some plaintext branches that do not undergo round function F operations and can be used to characterize the relationship between plaintexts and ciphertexts. If only the linear transformation part of the generalized Feistel network is modified, we merely permute the ciphertexts of different branches, but the differential propagation relationship between plaintexts and ciphertexts still holds. From this perspective, our attack scheme is actually applicable to most existing variants of Type-I and Type-II generalized Feistel networks, which possesses strong universality.

In this section, we investigate the applicability of the full plaintext recovery attacks proposed in this work to some practical cryptographic schemes based on GFN, with the main focus on analyzing CAST-256 and CLEFIA.

5.1 Full Plaintext Recovery Attacks on CAST-256

CAST-256 [AG99] is a 128-bit block cipher designed by Adams and Tavares, which was selected as a candidate in the first round of the AES competition. The cipher adopts a 4-branch Type-I GFN structure. It supports 128, 160, 192, 224 and 256-bit key lengths, with 48 rounds in all cases.

The 128-bit internal state is partitioned into four 32-bit branches (P_0, P_1, P_2, P_3). The 48 rounds are organized into 12 groups of quad-rounds, each utilizing four distinct round functions drawn from the CAST design family, which combine modular addition/subtraction, XOR, and 8×32 S-box lookups with key-dependent rotation.

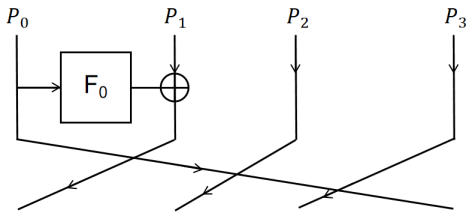


Figure 9: A single round transformation of CAST-256

In each round, only one round function F operates on the leftmost branch P_0 using the corresponding round key and the output is XORed into the adjacent branch P_1, followed by a left cyclic shift of all four branches. This is precisely the $d = 4$ case of the Type-I GFN transformation specified in Section 3.

In this section, we conduct experiments on 4-round and 8-round CAST-256 to verify the applicability and correctness of the full plaintext recovery attack proposed in Section 3. In the simulation, SHA-256 [Nat15] operating in the counter mode is adopted to construct the round function F. As a keyed pseudorandom function, SHA-256 performs operations with round keys concatenated with 32-bit input branches after appending the round function index.

The target plaintext is randomly generated as

$$\boldsymbol{P^0} = (\texttt{0x3b5f8c2a}, \texttt{0xd97e1406}, \texttt{0x68a3bf71}, \texttt{0xec249d53})$$

The 128-bit master key is fixed as

$$K = \texttt{0x8f1a3c5d7e2b9406a8c1d3e5f7284b6a}$$

For the r-round experiments ($r \in \{4, 8\}$), r round keys are derived from the master key K via a SHA-256-based key expansion method and listed in Table 1.

Table 1: Round keys for the 8-round CAST-256

Round i	$\boldsymbol{RK_{r-1}}$
1	0x15ea7644
2	0x36fa2248
3	0xb60bb19d
4	0xbd6c63a0
5	0xa575f6d4
6	0x464f9bec
7	0x46725975
8	0x770bc1c6

For 4-round CAST-256, only the first 4 rows are used. For 8-round CAST-256, all 8

rows are adopted. Then, the target ciphertexts can be computed as

$$\boldsymbol{C^0} = \begin{cases} E_{[4]}(\boldsymbol{P^0}) = (\texttt{0xedb06ae7}, \texttt{0x2f10ba7d}, \texttt{0xd69ae8fb}, \texttt{0x98824693}), & r = 4 \\ E_{[8]}(\boldsymbol{P^0}) = (\texttt{0x4f099ec5}, \texttt{0xe09beb61}, \texttt{0x4e7bd281}, \texttt{0xf385543d}), & r = 8 \end{cases}$$

Example 1. (**4-round CAST-256 under CPA**)

According to Proposition 1 in Section 3.1, a 4-round CAST-256 requires $d + 1 = 5$ encryption oracle $E_{[4]}$ queries to fully recover the target plaintext $\boldsymbol{P^0}$. The detailed attack steps are as follows.

(1) Selecting an arbitrary plaintext $\boldsymbol{P^1} \neq \boldsymbol{P^0}$, then querying the encryption oracle $E_{[4]}$

$$\boldsymbol{P^1} = (\texttt{0x7c14a93e}, \texttt{0x2f8d5b71}, \texttt{0xa406c8d5}, \texttt{0x1e973f28})$$
$$\boldsymbol{C^1} = E_{[4]}(\boldsymbol{P^1}) = (\texttt{0xbb950afb}, \texttt{0xa95936fd}, \texttt{0x34a3c1e7}, \texttt{0x417f55e6})$$

The ciphertext difference $\Delta^1 = \boldsymbol{C^1} \oplus \boldsymbol{C^0}$ is computed as

$$\Delta^1 = (\texttt{0x5625601c}, \texttt{0x86498c80}, \texttt{0xe239291c}, \texttt{0xd9fd1375})$$

(2) Constructing $\boldsymbol{P^2}$ by XORing Δ_0^1 into the last branch P_3^1, while keeping P_0^1, P_1^1, P_2^1 unchanged compared with $\boldsymbol{P^1}$

$$P_0^2 = P_0^1 = \texttt{0x7c14a93e}$$
$$P_1^2 = P_1^1 = \texttt{0x2f8d5b71}$$
$$P_2^2 = P_2^1 = \texttt{0xa406c8d5}$$
$$P_3^2 = P_3^1 \oplus \Delta_0^1 = \texttt{0x1e973f28} \oplus \texttt{0x5625601c} = \texttt{0x48b25f34}$$

Querying the encryption oracle to obtain $\boldsymbol{C^2}$

$$\boldsymbol{C^2} = (\texttt{0xedb06ae7}, \texttt{0x685b9f69}, \texttt{0x34a3c1e7}, \texttt{0x417f55e6})$$

One can observe that $C_0^2 = C_0^0 = \texttt{0xedb06ae7}$, which confirms the theoretical result $C_0^2 = C_0^0$ established in Proposition 1. The difference $\Delta^2 = \boldsymbol{C^2} \oplus \boldsymbol{C^1}$ is

$$\Delta^2 = (\texttt{0x5625601c}, \texttt{0xc102a994}, \texttt{0x00000000}, \texttt{0x00000000})$$

(3) Constructing $\boldsymbol{P^3}$ by XORing $\Delta_1^2 \oplus \Delta_1^1$ into P_0^2, while keeping the other branches unchanged compared with $\boldsymbol{P^2}$

$$P_0^3 = P_0^2 \oplus \Delta_1^2 \oplus \Delta_1^1 = \texttt{0x7c14a93e} \oplus \texttt{0xc102a994} \oplus \texttt{0x86498c80} = \texttt{0x3b5f8c2a}$$
$$P_1^3 = P_1^2 = \texttt{0x2f8d5b71}$$
$$P_2^3 = P_2^2 = \texttt{0xa406c8d5}$$
$$P_3^3 = P_3^2 = \texttt{0x48b25f34}$$

Querying the encryption oracle to obtain $\boldsymbol{C^3}$

$$\boldsymbol{C^3} = (\texttt{0x350ec9a6}, \texttt{0x79866af1}, \texttt{0x2069a78c}, \texttt{0x08794656})$$

The difference $\Delta^3 = \boldsymbol{C^3} \oplus \boldsymbol{C^2}$ is

$$\Delta^3 = (\texttt{0xd8bea341}, \texttt{0x11ddf598}, \texttt{0x14ca666b}, \texttt{0x490613b0})$$

By Proposition 1, the first two branches P_0^0 and P_1^0 are recovered as

$$\begin{aligned} P_0^0 &= P_0^3 = \texttt{0x3b5f8c2a} \\ P_1^0 &= P_1^3 \oplus \Delta_2^3 \oplus \Delta_2^1 \\ &= \texttt{0x2f8d5b71} \oplus \texttt{0x14ca666b} \oplus \texttt{0xe239291c} \\ &= \texttt{0xd97e1406} \end{aligned}$$

(4) Constructing $\boldsymbol{P^4}$ by XORing Δ_2^1 into P_1^1

$$\begin{aligned} P_0^4 &= P_0^1 = \texttt{0x7c14a93e} \\ P_1^4 &= P_1^1 \oplus \Delta_2^1 = \texttt{0x2f8d5b71} \oplus \texttt{0xe239291c} = \texttt{0xcdb4726d} \\ P_2^4 &= P_2^1 = \texttt{0xa406c8d5} \\ P_3^4 &= P_3^1 = \texttt{0x1e973f28} \end{aligned}$$

Querying the encryption oracle to obtain $\boldsymbol{C^4}$

$$\boldsymbol{C^4} = (\texttt{0x37176822}, \texttt{0x7cc7c88e}, \texttt{0xd69ae8fb}, \texttt{0x54273137})$$

The third branch P_2^0 is recovered as

$$\begin{aligned} P_2^0 &= P_2^4 \oplus C_3^4 \oplus C_3^0 \\ &= \texttt{0xa406c8d5} \oplus \texttt{0x54273137} \oplus \texttt{0x98824693} \\ &= \texttt{0x68a3bf71} \end{aligned}$$

(5) Repeating the above step with the difference shifted backward, construct $\boldsymbol{P^5}$ by XORing Δ_3^1 into P_2^1

$$\begin{aligned} P_0^5 &= P_0^1 = \texttt{0x7c14a93e} \\ P_1^5 &= P_1^1 = \texttt{0x2f8d5b71} \\ P_2^5 &= P_2^1 \oplus \Delta_3^1 = \texttt{0xa406c8d5} \oplus \texttt{0xd9fd1375} = \texttt{0x7dfbdba0} \\ P_3^5 &= P_3^1 = \texttt{0x1e973f28} \end{aligned}$$

Querying the encryption oracle to obtain $\boldsymbol{C^5}$

$$\boldsymbol{C^5} = (\texttt{0x1f03c89c}, \texttt{0xccb75cc7}, \texttt{0x34a3c1e7}, \texttt{0x98824693})$$

The last branch P_3^0 is recovered as

$$\begin{aligned} P_3^0 &= P_3^5 \oplus C_0^5 \oplus C_0^0 \\ &= \texttt{0x1e973f28} \oplus \texttt{0x1f03c89c} \oplus \texttt{0xedb06ae7} \\ &= \texttt{0xec249d53} \end{aligned}$$

After these attack steps, we can recover the target plaintext by

$$\begin{aligned} P_0^0 &= P_0^3 = \texttt{0x3b5f8c2a} \\ P_1^0 &= P_1^3 \oplus \Delta_2^3 \oplus \Delta_2^1 = \texttt{0xd97e1406} \\ P_2^0 &= P_2^4 \oplus C_3^4 \oplus C_3^0 = \texttt{0x68a3bf71} \\ P_3^0 &= P_3^5 \oplus C_0^5 \oplus C_0^0 = \texttt{0xec249d53} \end{aligned}$$

Example 2. (**8-round CAST-256 under CCA**)

According to Proposition 2 in Section 3.2, an 8-round CAST-256 ($2d = 8$ with $d = 4$) requires $d = 4$ decryption oracle queries under the CCA model. We select the non-zero difference Δ as

$$\Delta = \texttt{0x00000001}$$

Any non-zero 32-bit constant is applicable, the value 1 with the minimal Hamming weight is adopted for simplicity.

(1) Constructing $\boldsymbol{C^1}$ by XORing Δ into the first ciphertext branch C_0^0, while keeping the remaining branches unchanged

$$\begin{aligned}
C_0^1 &= C_0^0 \oplus \Delta = \texttt{0x4f099ec5} \oplus \texttt{0x00000001} = \texttt{0x4f099ec4} \\
C_1^1 &= C_1^0 = \texttt{0xe09beb61} \\
C_2^1 &= C_2^0 = \texttt{0x4e7bd281} \\
C_3^1 &= C_3^0 = \texttt{0xf385543d}
\end{aligned}$$

Querying the punctured decryption oracle $\mathring{D}_{[8]}^{\boldsymbol{C^0}}$ to obtain

$$\boldsymbol{P^1} = (\texttt{0xce6a4773}, \texttt{0xf7f1b135}, \texttt{0xc2734000}, \texttt{0xec249d52})$$

By Proposition 2, the last branch P_3^0 is recovered as

$$P_3^0 = P_3^1 \oplus \Delta = \texttt{0xec249d52} \oplus \texttt{0x00000001} = \texttt{0xec249d53}$$

(2) Constructing $\boldsymbol{C^2}$ by shifting the difference backward to C_1^0

$$\begin{aligned}
C_0^2 &= C_0^0 = \texttt{0x4f099ec5} \\
C_1^2 &= C_1^0 \oplus \Delta = \texttt{0xe09beb61} \oplus \texttt{0x00000001} = \texttt{0xe09beb60} \\
C_2^2 &= C_2^0 = \texttt{0x4e7bd281} \\
C_3^2 &= C_3^0 = \texttt{0xf385543d}
\end{aligned}$$

Querying the decryption oracle yields

$$\boldsymbol{P^2} = (\texttt{0x3b5f8c2b}, \texttt{0x97a5c3ee}, \texttt{0xa13a9afb}, \texttt{0xec249d53})$$

The first branch P_0^0 can be recovered as

$$P_0^0 = P_0^2 \oplus \Delta = \texttt{0x3b5f8c2b} \oplus \texttt{0x00000001} = \texttt{0x3b5f8c2a}$$

(3) Shifting the difference to C_2^0

$$\begin{aligned}
C_0^3 &= C_0^0 = \texttt{0x4f099ec5} \\
C_1^3 &= C_1^0 = \texttt{0xe09beb61} \\
C_2^3 &= C_2^0 \oplus \Delta = \texttt{0x4e7bd281} \oplus \texttt{0x00000001} = \texttt{0x4e7bd280} \\
C_3^3 &= C_3^0 = \texttt{0xf385543d}
\end{aligned}$$

The decryption oracle returns

$$\boldsymbol{P^3} = (\texttt{0x3b5f8c2a}, \texttt{0xd97e1407}, \texttt{0xad26645b}, \texttt{0xcd795223})$$

The second branch P_1^0 is recovered as

$$P_1^0 = P_1^3 \oplus \Delta = \texttt{0xd97e1407} \oplus \texttt{0x00000001} = \texttt{0xd97e1406}$$

(4) Finally, shifting the difference to C_3^0

$$C_0^4 = C_0^0 = \texttt{0x4f099ec5}$$
$$C_1^4 = C_1^0 = \texttt{0xe09beb61}$$
$$C_2^4 = C_2^0 = \texttt{0x4e7bd281}$$
$$C_3^4 = C_3^0 \oplus \Delta = \texttt{0xf385543d} \oplus \texttt{0x00000001} = \texttt{0xf385543c}$$

The final decryption query yields

$$\boldsymbol{P^4} = (\texttt{0xad5a9f71}, \texttt{0xac1e66fb}, \texttt{0x68a3bf70}, \texttt{0xc8b4de0a})$$

From which the third branch P_2^0 is recovered

$$P_2^0 = P_2^4 \oplus \Delta = \texttt{0x68a3bf70} \oplus \texttt{0x00000001} = \texttt{0x68a3bf71}$$

After these four steps, the complete target plaintext is recovered by

$$P_0^0 = P_0^2 \oplus \Delta = \texttt{0x3b5f8c2a}$$
$$P_1^0 = P_1^3 \oplus \Delta = \texttt{0xd97e1406}$$
$$P_2^0 = P_2^4 \oplus \Delta = \texttt{0x68a3bf71}$$
$$P_3^0 = P_3^1 \oplus \Delta = \texttt{0xec249d53}$$

The above simulation confirms that the theoretical analysis of Section 3 yields implementable attacks on reduced round CAST-256. Under the CPA model, the full plaintext can be recovered in $d + 1 = 5$ encryption queries for the 4-round variant, while $d = 4$ decryption queries suffice for the 8-round variant under the CCA model. Although the full round CAST-256 remains secure against these structural attacks, this results demonstrate that low-round Type-I GFN instances with $d = 4$ branches are vulnerable to full plaintext recovery with a polynomial number of oracle queries.

5.2 Full Plaintext Recovery Attacks on Low-round CLEFIA

CLEFIA [SSAM07] is a 128-bit block cipher standardized in ISO/IEC 29192-2 for lightweight cryptography. It adopts a 4-branch Type-II GFN, where the internal state is split into four 32-bit branches (P_0, P_1, P_2, P_3).

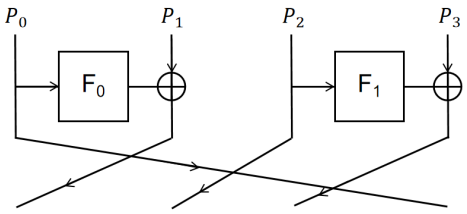


Figure 10: A single round transformation of CLEFIA

In each encryption round, two round functions are executed on even-indexed branches, whose outputs are XORed into the neighboring odd-indexed branches, followed by a left cyclic shift operation. This structure exactly corresponds to the case $d = 4$ of the Type-II transformation given in Section 4.

Both round functions F_0 and F_1 are constructed with double SPN structures, which take a 32-bit plaintext input and a round key to generate a 32-bit output. Specifically, distinct functions F_{2i} and F_{2i+1} are utilized in the i-th round. CLEFIA supports key lengths of

128, 192 and 256 bits, corresponding to 18 rounds, 20 rounds and 20 rounds respectively. All round keys are generated from the master key through a 12-round GFN-based key expansion algorithm.

In this section, we conduct experiments on 2-round and 3-round CLEFIA to verify the applicability and correctness of the full plaintext recovery attack proposed in Section 4. In the simulation, we still adopt SHA-256 operating in the counter mode to construct the round function F.

The target plaintext is randomly generated as

$$\boldsymbol{P^0} = (0\text{x}236\text{b}624\text{d}, 0\text{xfb}92821\text{a}, 0\text{x}253\text{a}1\text{c}77, 0\text{x}9\text{acdd}4\text{e}4)$$

The 128-bit master key is fixed as

$$K = 0\text{x}06\text{cc}02\text{a}00\text{e}523\text{f}0\text{c}2\text{efc}1\text{fbf}42\text{faa}49\text{a}$$

For the r-round experiments ($r \in \{2, 3\}$), $2r$ round-keys are generated from the master key K via a SHA-256-based key expansion method and listed in Table 2.

Table 2: Round keys for 3-round CLEFIA

Round r	$\boldsymbol{RK_{2r-2}}$	$\boldsymbol{RK_{2r-1}}$
1	0xd141468b	0x90ce6680
2	0xa8fb1797	0x99df1cbd
3	0xf37492ae	0x4c95004e

For 2-round CLEFIA, only the first 2 rows of Table 2 are used. For 3-round CLEFIA, all 3 rows are used. Then, we can calculate the target ciphertext $\boldsymbol{C^0}$

$$\boldsymbol{C^0} = \begin{cases} E_{[2]}(\boldsymbol{P^0}) = (0\text{xb}327\text{f}5\text{df}, 0\text{x}1\text{fa}0\text{d}862, 0\text{x}4\text{e}46\text{a}1\text{fc}, 0\text{x}5\text{bddd}6\text{b}5), & r = 2 \\ E_{[3]}(\boldsymbol{P^0}) = (0\text{x}1\text{fa}0\text{d}862, 0\text{x}0\text{a}3\text{b}390\text{e}, 0\text{x}5\text{bddd}6\text{b}5, 0\text{x}6\text{fa}5\text{b}796), & r = 3 \end{cases}$$

Example 3. (**2-round CLEFIA under CPA**)

(1) According to Proposition 3, firstly select an arbitrary plaintext $P^1 \neq P^0$ and querying the encryption oracle $E_{[2]}$

$$\boldsymbol{P^1} = (0\text{xa}187\text{bb}3\text{d}, 0\text{x}35\text{a}77991, 0\text{x}528101\text{e}2, 0\text{x}751\text{b}3595)$$
$$\boldsymbol{C^1} = E_{[2]}(\boldsymbol{P^1}) = (0\text{xa}4\text{fbf}5\text{ca}, 0\text{x}3\text{b}7788\text{c}7, 0\text{xd}5594\text{f}06, 0\text{xce}1734\text{e}6)$$

The ciphertext difference $\Delta_1 = \boldsymbol{C^1} \oplus \boldsymbol{C^0}$ is computed as

$$\Delta^1 = (0\text{x}17\text{dc}0015, 0\text{x}24\text{d}750\text{a}5, 0\text{x}9\text{b}1\text{feefa}, 0\text{x}95\text{cae}253)$$

The values $\Delta_0^1 = 0\text{x}17\text{dc}0015$ and $\Delta_2^1 = 0\text{x}9\text{b}1\text{feefa}$, which correspond to the even-indexed ciphertext branches, are injected into the odd-indexed plaintext branches in the subsequent step.

(2) Constructing $\boldsymbol{P^2}$ by XORing Δ^1 into the odd-indexed branches P_1^1 and P_3^1, while keeping the even-indexed branches P_0^1 and P_2^1 consistent with those of $\boldsymbol{P^1}$.

$$P_0^2 = P_0^1 = 0\text{xa}187\text{bb}3\text{d}$$
$$P_1^2 = P_1^1 \oplus \Delta_0^1 = 0\text{x}35\text{a}77991 \oplus 0\text{x}17\text{dc}0015 = 0\text{x}227\text{b}7984$$
$$P_2^2 = P_2^1 = 0\text{x}528101\text{e}2$$
$$P_3^2 = P_3^1 \oplus \Delta_2^1 = 0\text{x}751\text{b}3595 \oplus 0\text{x}9\text{b}1\text{feefa} = 0\text{xee}04\text{db}6\text{f}$$

Querying the encryption oracle to obtain

$$\boldsymbol{C^2} = (0\text{xb327f5df}, 0\text{x681bc5f7}, 0\text{x4e46a1fc}, 0\text{xd9310fc5}).$$

It's easy to notice that $C_0^2 = C_0^0 = 0\text{xb327f5df}$ and $C_2^2 = C_2^0 = 0\text{x4e46a1fc}$. The difference $\Delta^2 = \boldsymbol{C^2} \oplus \boldsymbol{C^1}$ is calculated as

$$\Delta^2 = (0\text{x17dc0015}, 0\text{x536c4d30}, 0\text{x9b1feefa}, 0\text{x17263b23})$$

Among them, $\Delta_1^2 = 0\text{x536c4d30}$ and $\Delta_3^2 = 0\text{x17263b23}$ are adopted for the subsequent operation.

(3) Constructing $\boldsymbol{P^3}$ by XORing Δ^2 into the even-indexed branches P_0^2 and P_2^2, while keeping the odd-indexed branches P_1^2 and P_3^2 unchanged as those of $\boldsymbol{P^2}$.

$$\begin{aligned}
P_0^3 &= P_0^2 \oplus \Delta_3^2 \oplus \Delta_3^1 = 0\text{xa187bb3d} \oplus 0\text{x17263b23} \oplus 0\text{x95cae253} = 0\text{x236b624d}\\
P_1^3 &= P_1^2 = 0\text{x227b7984}\\
P_2^3 &= P_2^2 \oplus \Delta_1^2 \oplus \Delta_1^1 = 0\text{x528101e2} \oplus 0\text{x536c4d30} \oplus 0\text{x24d750a5} = 0\text{x253a1c77}\\
P_3^3 &= P_3^2 = 0\text{xee04db6f}
\end{aligned}$$

Querying the encryption oracle $E_{[2]}(\boldsymbol{P^3})$ obtains

$$\boldsymbol{C^3} = (0\text{x6ace0e41}, 0\text{xd8ebb2d5}, 0\text{x3a8fae77}, 0\text{xa6f87fc4}).$$

The difference $\Delta^3 = \boldsymbol{C^3} \oplus \boldsymbol{C^2}$ is computed as

$$\Delta^3 = (0\text{xd9e9fb9e}, 0\text{xb0f07722}, 0\text{x74c90f8b}, 0\text{x7fc97001}).$$

After these attack steps, the plaintext $\boldsymbol{P^0}$ is recovered as follows

$$\begin{aligned}
P_0^0 &= P_0^3 = 0\text{x236b624d}\\
P_1^0 &= P_1^3 \oplus \Delta_0^3 = 0\text{x227b7984} \oplus 0\text{xd9e9fb9e} = 0\text{xfb92821a}\\
P_2^0 &= P_2^3 = 0\text{x253a1c77}\\
P_3^0 &= P_3^3 \oplus \Delta_2^3 = 0\text{xee04db6f} \oplus 0\text{x74c90f8b} = 0\text{x9acdd4e4}
\end{aligned}$$

Example 4. (**3-round CLEFIA under CCA**)

Under the chosen-ciphertext attack model, the adversary can further access the 3-round decryption oracle $\mathring{D}_{[3]}^{\boldsymbol{C^0}}$. This attack requires 1 encryption query and 2 decryption queries. The non-zero difference vector $\boldsymbol{\Delta}$ is selected as

$$\boldsymbol{\Delta} = (\Delta^{(0)}, \Delta^{(1)}) = (0\text{x00000001}, 0\text{x00000001})$$

Any non-zero 32-bit constant is applicable. Here, the value 1 with the minimal Hamming weight is adopted for simplicity.

(1) According to Proposition 4, firstly construct $\boldsymbol{C^1}$ by superimposing $\boldsymbol{\Delta}$ on the odd-indexed ciphertext branches C_1^0 and C_3^0, while keep the even-indexed branches C_0^0 and C_2^0 unchanged

$$\begin{aligned}
C_0^1 &= C_0^0 = 0\text{x1fa0d862}\\
C_1^1 &= C_1^0 \oplus \Delta^{(0)} = 0\text{x0a3b390e} \oplus 0\text{x00000001} = 0\text{x0a3b390f}\\
C_2^1 &= C_2^0 = 0\text{x5bddd6b5}\\
C_3^1 &= C_3^0 \oplus \Delta^{(1)} = 0\text{x6fa5b796} \oplus 0\text{x00000001} = 0\text{x6fa5b797}
\end{aligned}$$

Querying the decryption oracle obtains

$$\boldsymbol{P^1} = D_{[3]}(\boldsymbol{C^1}) = (\text{0x0c9773d7}, \text{0x913f08a4}, \text{0xff8176cb}, \text{0x9a81ecec})$$

(2) Constructing $\boldsymbol{P^2}$ by XORing $\boldsymbol{\Delta}$ into the odd-indexed plaintext branches of $\boldsymbol{P^1}$

$$\begin{aligned}
P_0^2 &= P_0^1 = \text{0x0c9773d7} \\
P_1^2 &= P_1^1 \oplus \Delta^{(1)} = \text{0x913f08a4} \oplus \text{0x00000001} = \text{0x913f08a5} \\
P_2^2 &= P_2^1 = \text{0xff8176cb} \\
P_3^2 &= P_3^1 \oplus \Delta^{(0)} = \text{0x9a81ecec} \oplus \text{0x00000001} = \text{0x9a81eced}
\end{aligned}$$

Querying the encryption oracle obtains

$$\boldsymbol{C^2} = E_{[3]}(\boldsymbol{P^2}) = (\text{0xc51bb2de}, \text{0x6fc450ab}, \text{0x7421c72f}, \text{0x50045618})$$

(3) Constructing $\boldsymbol{C^3}$ by XORing the original differences $\Delta^{(0)}$ and $\Delta^{(1)}$ into the odd-indexed branches of $\boldsymbol{C^2}$

$$\begin{aligned}
C_0^3 &= C_0^2 = \text{0xc51bb2de} \\
C_1^3 &= C_1^2 \oplus \Delta^{(0)} = \text{0x6fc450ab} \oplus \text{0x00000001} = \text{0x6fc450aa} \\
C_2^3 &= C_2^2 = \text{0x7421c72f} \\
C_3^3 &= C_3^2 \oplus \Delta^{(1)} = \text{0x50045618} \oplus \text{0x00000001} = \text{0x50045619}
\end{aligned}$$

The decryption oracle yields

$$\boldsymbol{P^3} = (\text{0x236b624d}, \text{0xfb92821b}, \text{0x253a1c77}, \text{0x9acdd4e5})$$

After these attack steps, we can find that

$$\begin{aligned}
P_0^0 &= P_0^3 = \text{0x236b624d} \\
P_1^0 &= P_1^3 \oplus \Delta^{(1)} = \text{0xfb92821b} \oplus \text{0x00000001} = \text{0xfb92821a} \\
P_2^0 &= P_2^3 = \text{0x253a1c77} \\
P_3^0 &= P_3^3 \oplus \Delta^{(0)} = \text{0x9acdd4e5} \oplus \text{0x00000001} = \text{0x9acdd4e4}
\end{aligned}$$

This simulation confirms that the theoretical analysis of Sections 4 produces practical, implementable attacks when the number of rounds is low enough. Although the full round CLEFIA remains secure against these specific attacks, we find a new way to recovery target plaintext with polynomial number of queries as long as we can resolve the issue of differential diffusion.

6 Summary and Discussion

In this paper, we have systematically investigated full plaintext recovery attacks on generalized Feistel networks with $d \geq 3$ branches, extending the recent paradigm of Guo beyond standard Feistel structure to Type-I and Type-II GFN. Under the CPA model, we presented a deterministic attack that fully recovers the target plaintext of a d-round Type-I GFN with only $d+1$ encryption queries and a 2-round Type-II GFN with 3 encryption queries. Under the CCA model, with a punctured decryption oracle, we further showed that $2d$-round Type-I GFN succumbs to d decryption queries and 3-round Type-II GFN can be broken with 1 encryption plus 2 decryption queries. All attacks succeed with probability 1 and treat the internal round functions as ideal black boxes, meaning the results apply

to any instantiation regardless of the cryptographic strength of the F-functions. The practicality of these theoretical constructions was validated through concrete simulations on reduced-round instances of two real-world block ciphers: CAST-256 (a 4-branch Type-I GFN, broken at 4 rounds under CPA and 8 rounds under CCA) and CLEFIA (a 4-branch Type-II GFN standardized in ISO/IEC 29192-2). These case studies demonstrate that the structural vulnerability exposed by our attacks is not merely a theoretical curiosity, but translates into directly implementable procedures when the round count is limited.

Two natural directions emerge from this work. On the classical side, extending the full plaintext recovery methodology to Type-III Generalized Feistel Networks represents an important open problem, as many lightweight designs employ Type-III structures whose resistance to such structural attacks remains unexplored. On the quantum side, a promising avenue is to integrate the existing body of quantum distinguishing results on GFNs into the full plaintext recovery framework established in this paper. Prior work has already demonstrated that Simon's period-finding algorithm can construct polynomial-time quantum distinguishers for substantially higher round counts than what classical methods permit, with results reaching $(d^2 - d + 2)$-round Type-I GFN and $4d$-round Type-II GFN by exploiting the periodic structures inherent in the branch-wiring patterns of multi-branch Feistel networks. If these quantum distinguishers can be systematically upgraded to full plaintext recovery attacks in the quantum CPA/CCA setting, following a paradigm analogous to the classical one developed here, the recoverable round thresholds would increase dramatically, potentially intersecting with the design parameters of practical, full-round ciphers such as CLEFIA (18/20/22 rounds) and CAST-256 (48 rounds). Achieving either of these extensions would mark a significant escalation from "distinguishing attacks of theoretical interest" to "plaintext recovery attacks with practical implications," thereby recalibrating our understanding of both the classical and post-quantum security margins of GFN-based lightweight cryptography.

References

[AG99] Carlisle Adams and Jeff Gilchrist. The CAST-256 encryption algorithm. RFC 2612, Internet Engineering Task Force, September 1999. First published as AES candidate, 1998.

[BBI+21] Subhadeep Banik, Zhenzhen Bao, Takanori Isobe, Hiroyasu Kubo, Fukang Liu, Kazuhiko Minematsu, Kosei Sakamoto, Nao Shibata, and Maki Shigeri. WARP: Revisiting GFN for lightweight 128-bit block cipher. In *Selected Areas in Cryptography – SAC 2020*, volume 12804 of *Lecture Notes in Computer Science*, pages 535–564. Springer, 2021. doi:10.1007/978-3-030-81652-0_21.

[BM15] Céline Blondeau and Marine Minier. Efficient impossible differential, integral and zero-correlation cryptanalysis of type-II GFN. In *Fast Software Encryption – FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 92–113, Istanbul, Turkey, 2015. Springer. doi:10.1007/978-3-662-48116-5_5.

[CGL+26] Yuchao Chen, Chun Guo, Muzhou Li, Shuo Peng, Hao Lei, Guang Zeng, and Meiqin Wang. Computer-aided proof for extended generalized Feistel networks. *IACR Trans. Symmetric Cryptol.*, 2026(1), 2026.

[DDW20] Xiaoyu Dong, Boning Dong, and Xiaoyun Wang. Quantum attacks on some Feistel block ciphers. *Des. Codes Cryptogr.*, 88(6):1179–1203, 2020. doi:10.1007/s10623-020-00741-y.

[DE24] Patrick Derbez and Marie Euler. Equivalence of generalised Feistel networks. *IACR Trans. Symmetric Cryptol.*, 2024(1):412–440, 2024. doi:10.46586/tosc.v2024.i1.412-440.

[DJL18] Yuanhao Deng, Chenhui Jin, and Rongjia Li. Meet in the middle attack on type-1 Feistel construction. In *Information Security and Cryptology – Inscrypt 2017*, volume 10726 of *Lecture Notes in Computer Science*, pages 427–444, Beijing, China, 2018. Springer. Presented at Inscrypt 2017. doi:10.1007/978-3-319-75160-3_25.

[DJZ19] Yuanhao Deng, Chenhui Jin, and Jieqing Zhao. Meet-in-the-middle attack on type-3 Feistel structure. *J. Cryptologic Res.*, 6(1):27–36, 2019. doi:10.13868/j.cnki.jcr.000280.

[DLW19] Xiaoyu Dong, Zhengping Li, and Xiaoyun Wang. Quantum cryptanalysis on some generalized Feistel schemes. *Sci. China Inf. Sci.*, 62(2):022501, 2019. doi:10.1007/s11432-017-9436-7.

[DM17] Le Dong and Yuming Mao. Meet-in-the-middle attacks on three-branch type-I generalized Feistel network. IACR Cryptology ePrint Archive, Report 2017/1071, 2017. URL: https://eprint.iacr.org/2017/1071.

[GM16] Shay Gueron and Nicky Mouha. Simpira v2: A family of efficient permutations using the AES round function. In *Advances in Cryptology – ASIACRYPT 2016*, volume 10031 of *Lecture Notes in Computer Science*, pages 95–125, Hanoi, Vietnam, 2016. Springer. doi:10.1007/978-3-662-53887-6_4.

[GWJ+26] Tingting Guo, Peng Wang, Jiwu Jing, Shuping Mao, and Gang Liu. Classical and quantum full plaintext recovery for low-round Feistel-type designs. In *Advances in Cryptology – CRYPTO 2026*, Lecture Notes in Computer Science, Santa Barbara, CA, USA, 2026. Springer. Full version at IACR ePrint 2026/1040.

[HDQ+24] Nian Hou, Xiaoyu Dong, Tairong Qin, Wentao Zhang, and Xiaoyun Wang. Automated meet-in-the-middle attack goes to Feistel. In *Information Security and Cryptology – Inscrypt 2023*, Lecture Notes in Computer Science, Beijing, China, 2024. Springer. Full version at IACR ePrint 2023/1359.

[HR10] Viet Tung Hoang and Phillip Rogaway. On generalized Feistel networks. In *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 614–631, Santa Barbara, CA, USA, 2010. Springer. Full version at IACR ePrint 2010/301. doi:10.1007/978-3-642-14623-7_33.

[HSK+06] Deukjo Hong, Jin-Hee Sung, Seokhie Kim, Sangjin Lim, Sangjin Lee, Bon-Soo Koo, Changhoon Lee, Dukjae Chang, Jongtae Lee, Kitae Kim, Daesung Chee, Eunbo Park, Jongdeuk Kim, and Donghoon Lee. HIGHT: A new block cipher suitable for low-resource cryptographic applications. In *Cryptographic Hardware and Embedded Systems – CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59, Yokohama, Japan, 2006. Springer. doi:10.1007/11894063_4.

[ISO16] ISO/IEC JTC 1/SC 27. ISO/IEC 29192-5:2016 — information technology — security techniques — lightweight cryptography — part 5: Hash-functions. International Standard, 2016. Includes the Lesamnta-LW hash function.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, Boca Raton, FL, 2nd edition, 2014.

[Nat15] National Institute of Standards and Technology. FIPS pub 180-4: Secure hash standard (SHS). Federal Information Processing Standards Publication, 2015. URL: https://csrc.nist.gov/publications/detail/fips/180/4/final.

[NIDI19] Boyu Ni, Gembu Ito, Xiaoyu Dong, and Tetsu Iwata. Quantum attacks against type-1 generalized Feistel ciphers and applications to CAST-256. In *Progress in Cryptology – INDOCRYPT 2019*, volume 11898 of *Lecture Notes in Computer Science*, pages 433–455, Hyderabad, India, 2019. Springer. doi:10.1007/978-3-030-35423-7_22.

[SGW20] Xiao Shen, Jian Guo, and Meiqin Wang. Improved security bounds for generalized Feistel networks. *IACR Trans. Symmetric Cryptol.*, 2020(1):425–457, 2020. doi:10.13154/tosc.v2020.i1.425-457.

[SLL24] Jianhong Shi, Jiazhu Liu, and Rongjia Li. Impossible differential analysis of type-II GFN and its application to WARP. *Chin. J. Electron.*, 33(1):80–89, 2024. doi:10.23919/cje.2022.00.132.

[SM10] Tomoyasu Suzaki and Kazuhiko Minematsu. Improving the generalized Feistel. In *Fast Software Encryption – FSE 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 19–39, Seoul, Korea, 2010. Springer. doi:10.1007/978-3-642-13858-4_2.

[SMMK13] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Takeshi Kobara. TWINE: A lightweight block cipher for multiple-platform security. In *Selected Areas in Cryptography – SAC 2012*, volume 7707 of *Lecture Notes in Computer Science*, pages 292–308, Windsor, ON, Canada, 2013. Springer. Presented at SAC 2012. doi:10.1007/978-3-642-35999-6_18.

[SSAM07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, and Shiho Moriai. The 128-bit blockcipher CLEFIA (extended abstract). In *Fast Software Encryption – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195, Luxembourg, 2007. Springer. doi:10.1007/978-3-540-74619-5_12.

[SSMM11] Kyoji Shibutani, Taizo Shirai, Atsushi Mitsuda, and Shiho Moriai. Piccolo: An ultra-lightweight blockcipher. In *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 342–357, Nara, Japan, 2011. Springer. doi:10.1007/978-3-642-23916-9_23.

[Wag99] David Wagner. The boomerang attack. In *Fast Software Encryption – FSE '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170, Rome, Italy, 1999. Springer. doi:10.1007/3-540-48519-8_12.

[WZ11] Wenling Wu and Lei Zhang. LBlock: A lightweight block cipher. In *Applied Cryptography and Network Security – ACNS 2011*, volume 6715 of *Lecture Notes in Computer Science*, pages 327–344, Nerja, Spain, 2011. Springer. doi:10.1007/978-3-642-21554-4_19.

[YQZ+20] Qidi You, Xin Qian, Xuan Zhou, Ye Yuan, and Zhaoyang Wu. Research on quantum cryptanalysis on SMS4-like structure and NBC algorithm. *J. Cryptologic Res.*, 7(6):864–874, 2020. doi:10.13868/j.cnki.jcr.000412.

[ZMI90] Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. On the construction of block ciphers provably secure and not relying on any unproven hypotheses. In *Advances in Cryptology — CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 461–480, Heidelberg, 1990. Springer.