Structural Evaluation for Generalized Feistel Structures and Applications to LBlock and TWINE

Huiling Zhang[1,2,3](✉) and Wenling Wu[1,2,3]

[1] TCA Laboratory, SKLCS, Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] University of Chinese Academy of Sciences, Beijing, China
{zhanghuiling,wwl}@tca.iscas.ac.cn

Abstract. The generalized Feistel structure (GFS) is the variant of Feistel structure with $m > 2$ branches. While the GFS is widely used, the security is not well studied. In this paper, we propose a generic algorithm for searching integral distinguishers. By applying the algorithm, we prove that the low bound for the length of integral distinguishers is m^2+m-1 and $2m+1$ for Type-1 GFS and Type-2 GFS, respectively. Meanwhile, we evaluate the security of the improved Type-1 and Type-2 GFSs when the size of each branch and the algebraic degree of F-functions are specified. Our results show that the distinguishers are affected by the parameters to various levels, which will provide valuable reference for designing GFS ciphers. Although our search algorithm is generic, it can improve integral distinguishers for specific ciphers. For instance, it constructs several 16-round integral distinguishers for LBlock and TWINE, which directly extends the numbers of attacked rounds.

Keywords: Generalized Feistel structure · Integral distinguisher · Algebraic degree · Division property · LBlock · TWINE

1 Introduction

Feistel structure is a basic symmetric cryptographic primitive, which provides many superior design features, for example, both the encryption and decryption algorithms can be achieved with a single scheme and the round function can be non-bijective. The generalized Feistel structure (GFS) introduced by Nyberg [4] is the variant of Feistel structure with $m > 2$ branches. Many GFSs exist in the literature so far. The most popular versions are Type-1 as in CAST-256 [1] and Type-2 as in CLEFIA [6]. They inherit the superior features from Feistel structure, moreover, have advantages of high parallelism, simple design and suitability for low cost implementation. Recently, lightweight cryptography has become a hot topic. Thus the GFS is an attractive structure for a lightweight symmetric key primitive such as a block cipher or a hash function.

A. Biryukov and V. Goyal (Eds.): INDOCRYPT 2015, LNCS 9462, pp. 218–237, 2015.
DOI: 10.1007/978-3-319-26617-6_12

In 2010, Suzaki et al. introduced the improved Type-2 GFS by replacing the cycle shift in Type-2 GFS with the optimal permutation [7]. More precisely, they proposed the maximum diffusion round (DR) which is the minimum number of rounds such that every output nibble depends on every input nibble. And then they found that the cycle shift does not provide optimum DR when $m \geq 6$. Hence, they exhaustively searched all the optimum permutations for $m \leq 16$, and gave a generic construction whose DR is close to the lower bound when m is a power of 2. In [12], Yanagihara and Iwata did the similar work for Type-1 GFS. They showed that better DR can also be obtained if one uses other permutations, moreover, an generic construction of optimum permutations for arbitrary m was devised. As shown in [5,7,12], the improved GFS has more secure than the standard GFS.

Integral attack was firstly proposed by Daemen et al. to evaluate the security of Square cipher [2], and then it was unified by Knudsen and Wagner in FSE 2002 [3]. The crucial part is the construction of the integral distinguisher, i.e., choosing a set of plaintexts such that the states after several rounds have a certain property, e.g., the XOR-sum of all states equals to 0 with probability 1. This property is called balanced in the following. The integral attack tends to be one of bottlenecks for the security of the GFS as shown in [5]. [7,12] evaluated the security of Type-1, Type-2 and their improved versions against integral attack. Their results show there exist m^2- and $2m$- round integral distinguishers for Type-1 GFS and Type-2 GFS, respectively. 2DR- or (2DR$-$1)-round integral distinguishers exist for the improved Type-2 GFS. However, the specific properties of the F-functions are not utilized in the evaluations, and F-functions are restricted to be bijective.

In EUROCRYPT 2015 [10], Todo proposed a new notion, named division property, which is a generalized integral property. Based on the division property, he introduced a path search algorithm to derive the integral distinguishers for Feistel or SPN ciphers. The algorithm has several desirable features, such as, it can take advantage of the algebraic degree of the F-functions, and it can effectively construct the integral distinguisher even though the F-functions are non-bijective. Therefore, generalizing and applying the algorithm to the GFS will be very meaningful.

Our Contributions. In this paper, we evaluate the security of the GFS against integral attack. We first study the propagation characteristic of the division property for the GFS. Due to the rapid expansion of the vectors in the division property, it is difficult to directly trace the propagation when $m \geq 14$ even if we perform it by computer. Therefore, a technique named early reduce technique is devised to simplify the procedure, which works by detecting and discarding "useless" vectors. Then we propose a generic algorithm of constructing integral distinguishers. By using our algorithm, we prove that integral distinguishers for Type-1 GFS and Type-2 GFS with bijective F-functions can be extended by $m-1$ and 1 rounds, respectively. And we show (m^2+m-2)- and $2m$- round distinguishers exist even though the F-functions are non-bijective.

For the improved GFS, our results indicate that distinguishers vary with several parameters, such as the number of branches, size of each branch, algebraic degree of F-functions, permutation layer and whether F-functions are bijective or not, which is not reflected from previous analysis. Hence, our results can provide valuable reference for designing GFS ciphers.

Although our search algorithm is generic, it can improve integral distinguishers for specific ciphers. We construct for the first time several 16-round integral distinguishers for LBlock and TWINE. The integral attacks can thus be applied to 23-round LBlock, 23-round TWINE-80 and 24-round TWINE-128.

Paper Outline. Section 2 describes the GFS we focus in this paper and gives a brief review on the division property. The path search algorithm and the improved integral distinguishers for the GFS are shown in Sect. 3. In Sect. 4, we apply the improvements to the integral attacks against LBlock and TWINE. Finally, we conclude this paper in Sect. 5.

2 Preliminaries

In this section, we introduce the generalized Feistel structure and review the definition of the division property.

2.1 Generalized Feistel Structure

A GFS divides its input into $m > 2$ *branches* of n bits each, where n is defined as the *branch size*. The round function can be separated into two successive layers, as done in [7], a F-function layer and a permutation layer. The F-function layer is made of F-functions whose inputs are some of the branches and whose outputs are added to the other branches. The permutation layer is a shuffle of m branches. In this paper, we focus on the generalized Type-1 GFS and the generalized Type-2 GFS as shown in Fig. 1. Note that they are Type-1 and Type-2 GFS, respectively, when the permutation layer is the left cycle shift.

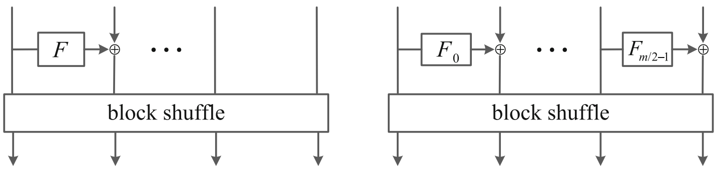


Fig. 1. Generalized Type-1 (left) and Type-2 (right) GFS

For convenience, we assume all F-functions are with algebraic degree d and $d < n$, which is reasonable for applicable ciphers. Let $P = \{p_0, p_1, \cdots, p_{m-1}\}$ denote the permutation layer moving i-th branch of the input to p_i-th branch (we number the branches from left to right, starting with 0 for the left-most branch), for example $P = \{3, 0, 1, 2\}$ for Type-1 GFS with 4 branches. Then, a GFS with parameters n,m,d and P can be defined as $[n, m, d, P]$-GFS.

2.2 Division Property

Some notations are first described for clarity. We use $\oplus$ and $+$ to distinct the XOR of $\mathbb{F}_2^n$ and the addition of $\mathbb{Z}$, and accordingly, $\bigoplus$ and $\sum$ represents XOR sum and addition sum, respectively. Denote the hamming weight of $u \in \mathbb{F}_2^n$ by $w(u)$ which is calculated as

$$w(u) = \sum_{0 \le i \le n-1} u[i],$$

where $u[i]$ is the i-th bit. Furthermore, denote the vectorial hamming weight of $U = (u_0, \cdots, u_{m-1}) \in (\mathbb{F}_2^n)^m$, $(w(u_0), \cdots, w(u_{m-1}))$, by $W(U)$. Let $K = (k_0, \cdots, k_{m-1})$ and $K' = (k'_0, \cdots, k'_{m-1})$ be the vectors in $\mathbb{Z}^m$. We define $K \succcurlyeq K'$ if $k_i \ge k'_i$ for $0 \le i \le m-1$, otherwise, $K \not\succcurlyeq K'$.

Subset $\mathbb{S}_{\boldsymbol{K}}^{\boldsymbol{n,m}}$. Let $\mathbb{S}_K^{n,m}$ be a subset of $(\mathbb{F}_2^n)^m$ for any vector $K = (k_0, \cdots, k_{m-1})$, where $0 \le k_i \le n$. The subset $\mathbb{S}_K^{n,m}$ is composed of all $U \in (\mathbb{F}_2^n)^m$ satisfying $W(U) \succcurlyeq K$, *i.e.*,

$$\mathbb{S}_K^{n,m} = \{U \in (\mathbb{F}_2^n)^m | W(U) \succcurlyeq K\}.$$

Bit Product Functions $\boldsymbol{\pi_u}$ and $\boldsymbol{\pi_U}$. Let $\pi_u : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function for $u \in \mathbb{F}_2^n$. For any $x \in \mathbb{F}_2^n$, $\pi_u(x)$ is the AND of $x[i]$ for i satisfying $u[i] = 1$. Namely, the bit product function π_u is defined as

$$\pi_u(x) = \prod_{u[i]=1} x[i].$$

Let $\pi_U : (\mathbb{F}_2^n)^m \to \mathbb{F}_2$ be a function for $U = (u_0, u_1, \cdots, u_{m-1}) \in (\mathbb{F}_2^n)^m$. For any $X = (x_0, x_1, \cdots, x_{m-1}) \in (\mathbb{F}_2^n)^m$, $\pi_U(X)$ is calculated as

$$\pi_U(X) = \prod_{i=0}^{m-1} \pi_{u_i}(x_i).$$

Definition 1 (Division Property). *[10] Let Λ be a multi-set whose elements take values in $(\mathbb{F}_2^n)^m$, and $K^{(j)}$ $(0 \le j \le q-1)$ are m-dimensional vectors whose elements take a value between* 0 *and* n. *When the multi-set Λ has the division property $\mathcal{D}^{n,m}_{K^{(0)},K^{(1)},\cdots,K^{(q-1)}}$, it fulfils the following conditions: the checksum, $\bigoplus_{X \in \Lambda} \pi_U(X)$, equals to* 0 *if $U \in \{V \in (\mathbb{F}_2^n)^m | W(V) \not\succcurlyeq K^{(0)}, \cdots, W(V) \not\succcurlyeq K^{(q-1)}\}$. Moreover, the checksum becomes unknown if there exist i satisfying $W(U) \succcurlyeq K^{(i)}$.*

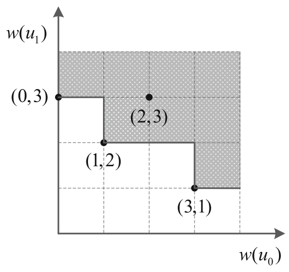


Fig. 2. Division Property $\mathcal{D}^{4,2}_{(0,3),(1,2),(2,3),(3,1)}$

We call U in the bit product function π_U as the *mask*, then view the division property from a vivid perspective as: it divides the set of all masks (i.e., $(\mathbb{F}_2^n)^m$) into two subsets, $\Gamma_?$ and Γ_0, where $\Gamma_?$ is the subset whose element results in an unknown checksum and Γ_0 is the subset whose element results in the zero-sum. Specifically, it has $\Gamma_? = \mathbb{S}^{n,m}_{K^{(0)}} \cup \cdots \cup \mathbb{S}^{n,m}_{K^{(q-1)}}$ and $\Gamma_0 = (\mathbb{F}_2^n)^m \backslash \Gamma_?$. Taking $\mathcal{D}^{4,2}_{(0,3),(1,2),(2,3),(3,1)}$ for an example, $\Gamma_?$ consists of all (u_0, u_1) located in the shadow area of Fig. 2. Note that this division property equals to $\mathcal{D}^{4,2}_{(0,3),(1,2),(3,1)}$, because they lead to the same division.

The division property is useful to construct integral distinguishers. The basic idea is that we choose a set of plaintexts satisfying certain division property and trace its propagation through $r+1$ encryption rounds until it has $\Gamma_0 \backslash \{(0^m)^n\} = \phi$. Since the cipher reduced to r rounds can be distinguished from a random permutation according to the checksum, a r-round integral distinguisher is thus constructed.

3 Improved Integral Distinguishers for GFS

In this section, we first study the propagation characteristic of the division property and construct an algorithm of searching integral distinguishers for the GFS. Meanwhile, a technique is proposed to optimize the program for wider applications. Finally, we apply the algorithm to evaluate the security of the GFS against integral attack.

3.1 Propagation Characteristic of the Division Property

The F-function layer of Type-2 GFS can be divided into three successive operations: "Type-2 copy", "Type-2 substitution" and "Type-2 compression" as depicted on Fig. 3, that is similar to [10] done for Feistel structure. We describe the propagation characteristics for these operations in Proposition 2–4 whose proofs are shown in Appendix A.

Proposition 1 (Type-2 Copy). *Let* $G : (\mathbb{F}_2^n)^m \to (\mathbb{F}_2^n)^{3m/2}$ *be the Type-2 copy, which accepts* $(x_0, \cdots, x_{m-1})$ *and produces* $(y_0, \cdots, y_{3m/2-1})$ *as* $(x_0, x_0, x_1,$

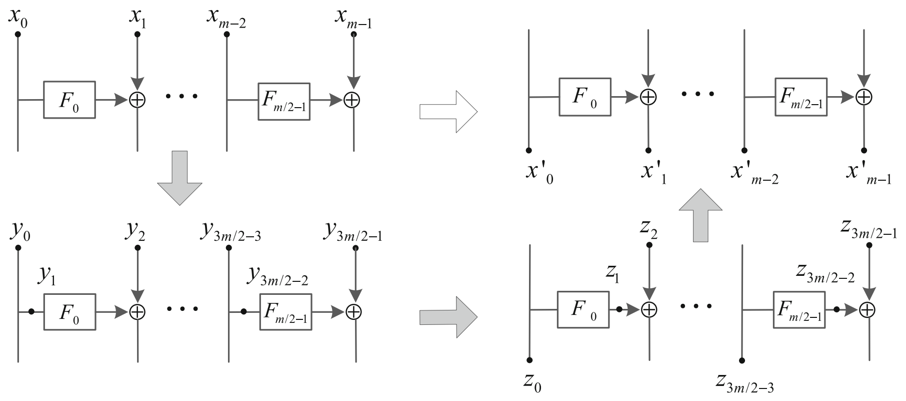


Fig. 3. Equivalent operations for GFS

$\cdots, x_{m-2}, x_{m-2}, x_{m-1})$. *If a multi-set of the inputs has division property* $\mathcal{D}_K^{n,m}$, *then the multi-set of the outputs has division property* $\mathcal{D}^{n,3m/2}_{K^{(0)},\cdots,K^{(q-1)}}$, *where* $\{K^{(0)},\cdots,K^{(q-1)}\}$ *is calculated as*

$$\{(i_0,(k_0-i_0),k_1,\cdots,i_{m/2-1},(k_{m-2}-i_{m/2-1}),k_{m-1}) | 0 \le i_j \le k_{2j}, 0 \le j < m/2\}.$$

Proposition 2 (Type-2 Substitution). *Let G be the Type-2 substitution, which accepts $(y_0,\cdots,y_{3m/2-1}) \in (\mathbb{F}_2^n)^m$ and produces $(z_0,\cdots,z_{3m/2-1}) \in (\mathbb{F}_2^n)^m$ as $(y_0, F_0(y_1), y_2, \cdots, y_{3m/2-3}, F_{m/2-1}(y_{3m/2-2}), y_{3m/2-1})$. If a multi-set of the inputs has division property $\mathcal{D}^{n,3m/2}_{K^{(0)},\cdots,K^{(q-1)}}$, then the multi-set of the outputs has division property $\mathcal{D}^{n,3m/2}_{K'^{(0)},\cdots,K'^{(q-1)}}$, where*

$$K'^{(j)} = \left(k_0^{(j)}, \left\lceil k_1^{(j)}/d \right\rceil, k_2^{(j)}, \cdots, k^{(j)}_{3m/2-3}, \left\lceil k^{(j)}_{3m/2-2}/d \right\rceil, k^{(j)}_{3m/2-1}\right).$$

Moreover, when the F-functions are bijective, we view $\lceil n/d \rceil$ as n.

Proposition 3 (Type-2 Compression). *Let $G : (\mathbb{F}_2^n)^{3m/2} \to (\mathbb{F}_2^n)^m$ be the Type-2 compression, which accepts $(z_0,\cdots,z_{3m/2-1})$ and produces $(x'_0,\cdots, x'_{m-1})$ as $(z_0,(z_1 \oplus z_2),\cdots,z_{3m/2-3},(z_{3m/2-2} \oplus z_{3m/2-1}))$. If a multi-set of the inputs has division property $\mathcal{D}^{n,3m/2}_{K^{(0)},\cdots,K^{(q-1)}}$, then the multi-set of the outputs has division property $\mathcal{D}^{n,m}_{K'^{(0)},\cdots,K'^{(q-1)}}$, where*

$$K'^{(j)} = (k_0^{(j)}, (k_1^{(j)} + k_2^{(j)}), \cdots, k^{(j)}_{3m/2-3}, (k^{(j)}_{3m/2-2} + k^{(j)}_{3m/2-1})).$$

Following Proposition 2–4, the propagation characteristic for Type-2 GFS is easily achieved. For simplicity, we write the division property $\mathcal{D}^{n,m}_{K^{(0)},\cdots,K^{(q-1)}}$ as a set of vectors, $\{K^{(0)},\cdots,K^{(q-1)}\}$.

Theorem 1. *For $[n, m, d, P]$-Type-2 GFS, if a multi-set of its inputs has division property $\mathcal{D}_K^{n,m}$, then the multi-set of the outputs from one encryption round has division property $\{\sigma(i_0, \lceil (k_0 - i_0)/d \rceil + k_1, \cdots, i_{m/2-1}, \lceil (k_{m-2} - i_{m/2-1})/d \rceil + k_{m-1}) | 0 \leq i_j \leq k_{2j}, 0 \leq j < m/2\}$, where σ moves i-th component of the input to p_i-th component.*

In the similar manner, we get the propagation characteristic for Type-1 GFS.

Theorem 2. *For $[n, m, d, P]$-Type-1 GFS, if a multi-set of its inputs has division property $\mathcal{D}_K^{n,m}$, then the multi-set of the outputs from one encryption round has division property $\{\sigma(i, \lceil (k_0 - i)/d \rceil + k_1, k_2, k_3, \cdots, k_{m-1}) \mid 0 \leq i \leq k_0\}$, where σ moves i-th component of the input to p_i-th component.*

3.2 Path Search Algorithm for GFS

The most troublesome problem for the propagation of division property is the rapid expansion of the vectors, which makes the procedure time-consuming and costing mass memory. Therefore, we devise a technique to discard "useless" vectors early. After that, we propose the path search algorithm.

Early Reduce Technique. This technique is based on the following observation:

Observation 1. Let K and K' be two vectors which respectively propagates to $\{K^{(0)}, \cdots K^{(q-1)}\}$ and $\{K'^{(0)}, \cdots K'^{(q'-1)}\}$ *through the round function. If there exists a vector, $K'^{(j)} \in \{K'^{(0)}, \cdots K'^{(q'-1)}\}$, such that $K^{(i)} \succcurlyeq K'^{(j)}$ for each $K^{(i)} \in \{K^{(0)}, \cdots K^{(q-1)}\}$, $\Omega \cup \{K, K'\}$ and $\Omega \cup \{K'\}$ propagate to the same division property for any vector set Ω.*

Note that $K \succcurlyeq K'$ certainly satisfies with the condition, however, not limitation to it. An example is K = (1,3), K' = (4,0) for [4,2,3,{1,0}]-Type-2 GFS with the bijective F-function, which is actually Feistel structure. K propagates to {(4,0),(3,1)}, while K' propagates to {(4,0),(1,1),(0,4)}. It has (4,0)$\succcurlyeq$(4,0) and (3,1)$\succcurlyeq$(1,1), therefore, {(1,3),(0,4)} and {(0,4)} propagate to the same division property as depicted on Fig. 4.

The early reduce technique discards vector K if there exist $K' \in \Omega$ satisfying Observation 1. It can amazingly reduce the division property, meanwhile, it does not change the division property achieved through one round function. To show the effectiveness, we compare the numbers of vectors before and after applying the technique to TWINE in Table 1.

Table 1. Comparison of the numbers of vectors

Round		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Num.	Before	1	2	3	5	11	31	184	1967	22731	113440	124827	42756	7072	952	164	44
	After	1	2	3	5	10	30	110	841	3709	10560	8976	2139	415	71	36	8

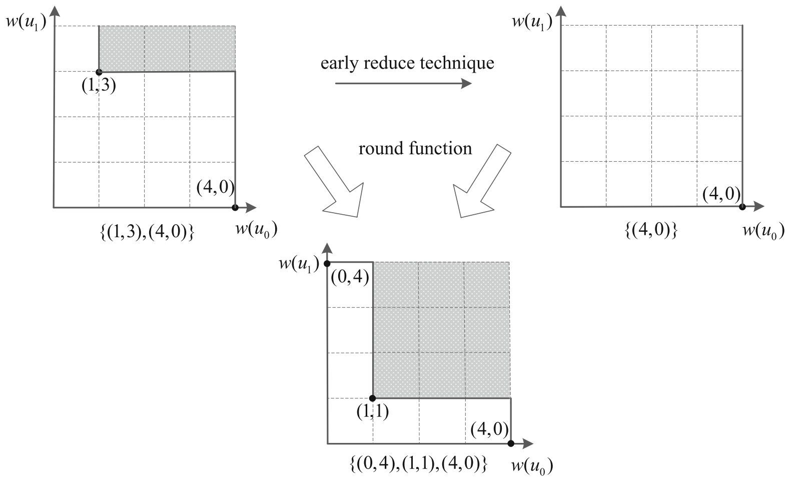


Fig. 4. An Example for early reduce technique

Path Search Algorithm. We then devise a generic algorithm of constructing integral distinguishers for the GFS, which is described in Algorithm 1. *RoundProp* propagates the division property through the round function by Theorem 1 (or 2). *TermCondition* judges whether the division property satisfies the terminal condition: if

$$1 < \max_{0 \le j \le q-1} \{ \sum_{0 \le i \le m-1} k_i^j \},$$

it returns true, otherwise, returns false. *AddDivision* adds vectors, $K_0^i, \cdots K_{q_i-1}^i$, to the vector set, Ω, and updates p which denotes the number of vectors in Ω. *SizeReduce* discards the vector $K \in \Omega$ if there exists vector $K' \in \Omega$ satisfying $K \succcurlyeq K'$, meanwhile, updates the value of p. *EarlyReduce* further reduces Ω by using the early reduce technique.

For Type-2 GFS, *EarlyReduce* is implemented as follows: we first create a list saving all vectors $K = (k_0, k_1)$ for each $K' = (k'_0, k'_1)$, which satisfies that there exists a vector $K'^{(j')} \in \{(i, \lceil (k'_0 - i)/d \rceil + k'_1) | 0 \le i \le k'_0\}$ such that $K^{(j)} \succcurlyeq K'^{(j')}$ for each $K^{(j)} \in \{(i, \lceil (k_0 - i)/d \rceil + k_1) | 0 \le i \le k_0\}$. Then, if (k_{2i}, k_{2i+1}) locals in the list of (k'_{2i}, k'_{2i+1}) for $0 \le i < m/2$, we discard $K = (k_0 \cdots k_{m-1})$ when $K' = (k'_0 \cdots k'_{m-1})$ is in the vector set Ω. Notice that, the function may change the result of *TermCondition*, therefore, we need to set a threshold to decide whether it will be performed. We suggest the threshold to be 20000 for $n = 4, m = 16$.

3.3 Improved Integral Distinguishers for GFS

We evaluate the security of $[n,m,d,P]$-Type-2 GFS and $[n,m,d,P]$-Type-1 GFS against integral attack by Algorithm 1. A low bound of the length of distinguishers

Algorithm 1. Path search algorithm for the GFS

```
**Input:** Parameters $n, m, d, P$ of the GFS and division property of the plaintext set $K = (k_0, k_1, \cdots, k_{m-1})$.
**Output:** The number of rounds of the integral distinguisher.
$0 \Rightarrow r$
$RoundProp(n, m, d, P, K) \Rightarrow \{K_0, K_1, \cdots K_{q-1}\}$
while $TermCondition(\{K_0, K_1, \cdots K_{q-1}\})$ do
  $r + 1 \Rightarrow r$
  $\emptyset \Rightarrow \Omega$, $0 \Rightarrow p$
  for $i = 0$ to $q - 1$ do
    $RoundProp(n, m, d, P, K_i) \Rightarrow \{K_0^i, K_1^i, \cdots K_{q_i-1}^i\}$
    $AddDivision(\{K_0^i, K_1^i, \cdots K_{q_i-1}^i\}) \Rightarrow (\Omega, p)$
    $SizeReduce(\Omega) \Rightarrow (\Omega, p)$
    if threshold $\leq p$ then
      $EarlyReduce(\Omega) \Rightarrow (\Omega, p)$
    end if
  end for
  $p \Rightarrow q$, $\Omega \Rightarrow \{K_0, K_1, \cdots K_{q-1}\}$
end while
return $r$
```

for Type-2 (or Type-1) GFS is first given, and then the lengths of the distinguishers for improved Type-2 (or Type-1) GFSs are specified.

Results on [*n, m, d, P*]-Type-2 GFS. We prove the following conclusion for Type-2 GFS.

Theorem 3. *For Type-2 GFS with $m \leq 16$ branches of size n, there always exist the integral distinguishers with at least $2m + 1$ rounds when F-functions are bijective, moreover, there exist the integral distinguishers with at least $2m$ rounds when F-functions are non-bijective.*

Proof. For simplicity, we prove the case when Type-2 GFS with $m = 4$ branches and bijective F-functions. The general case follows by a similar manner. Firstly, assume the degree of F-functions to be $n-1$ and $n \geq 4$. We start with the division property $K = \{(n-1, n, n, n)\}$ and trace its propagation by Algorithm 1, as shown in Table 2. Since the division property after 10 rounds will be {(0,0,0,1),(0,0,1,0), (0,1,0,0),(1,0,0,0)}, which reaches the terminal condition, integral distinguishers with 9 rounds are proved to be existed. Then, in the same way, we can prove the existence of 10-round distinguishers for $n = 3$. Due to the fact that the lower degree of F-functions, the longer distinguishers achieved by our path search algorithm, the results are actually low bounds.

For improved Type-2 GFSs, the shuffles do not show the regularity as the cycle shift, which leads to the absence of a similar conclusion. However, we search the integral distinguishers for each most common parameter. The results are summarized in Table 3 (m ≤8) and Table 6 (8< m ≤16, in Appendix C).

Table 2. Propagation of division property for Type-2 GFS.

Round	Division property
0	$\{(n\text{-}1,n,n,n)\}$
1	$\{(n,n,n,n\text{-}1)\}$
2	$\{(n,1,n,n),(n,n,n\text{-}1,n)\}$
3	$\{(2,n,n,1),(1,n,n,n),(n,n\text{-}1,n,n)\}$
4	$\{(n,1,2,2),(n,n,1,2),(n,n,n,1),(n\text{-}1,n,n,n)\}$
5	$\{(2,0,3,1),(2,2,2,1),(1,0,3,\mathrm{n}),(1,2,2,n),(n,1,2,n),(n,n,1,n)\}$
6	$\{(1,0,2,0),(1,3,1,0),(0,0,2,2),(0,3,1,2),(3,2,1,0),(2,2,1,2),(0,3,n,1)\}$
7	$\{(1,0,1,0),(1,2,0,0),(0,0,1,1),(0,2,0,1),(0,0,3,0),(0,2,2,0),(3,1,0,0),(2,1,0,3)\}$
8	$\{(1,1,0,0),(0,1,0,1),(0,0,1,0),(0,3,0,0),(2,0,0,0),(1,0,0,3)\}$
9	$\{(0,0,1,0),(0,1,0,0),(1,0,0,0),(0,0,0,2)\}$

Table 3. Integral distinguishers for improved type-2 GFS

m	Type	P	DR	IND [7]	IND for $[n,m,d,P]$-Type-2 GFS							
					$[n, d]$=[4,3]		$[n, d]$=[8,7]		$[n, d]$=[16,3]		$[n, d]$=[32,7]	
				bij	bij	nbij	bij	nbij	bij	nbij	bij	nbij
6	No. 1	$\{3, 0, 1, 4, 5, 2\}$	5	10	11	10	11	10	12	12	11	10
8	No. 1	$\{3, 0, 1, 4, 7, 2, 5, 6\}$	6	11	13	13	12	12	15	15	13	13
	No. 2	$\{3, 0, 7, 4, 5, 6, 1, 2\}$	6	11	13	12	12	12	13	13	12	12

Compared with the integral distinguishers in [7], our results extend the length of distinguishers by at least one round when the F-functions are bijective. Moreover, the distinguishers on Type-2 GFSs with non-bijective F-functions are constructed for the first time. Our results also indicate that the security of structures is sensitive to the parameters for different degrees. For example, No. 1 and No. 2 structures with $m = 8$ have the same length of distinguishers when n=4, d=3 and F-functions are bijective, however, No. 1 has longer distinguishers than No. 2 when n=16, d=3 and F-functions are bijective. This difference may help designers choosing the suitable structure to gain more security.

Results on $[n, m, d, P]$-Type-1 GFS. Similar to the proof of Theorem 3, we get the conclusion for Type-1 GFS.

Theorem 4. *For Type-1 GFS with $m \leq 16$ branches of size n, there always exist the integral distinguishers with at least $m^2 + m - 1$ rounds when F-functions are bijective, moreover, there exist the integral distinguishers with at least $m^2 + m - 2$ rounds when F-functions are non-bijective.*

We also search the integral distinguishers for improved Type-1 GFSs. The results are summarized in Table 4 ($m \leq 8$) and Table 5 ($8 < m \leq 16$, in Appendix B). An interesting observation is that our integral distinguishers have the same length with impossible differential distinguishers in [12] for all improved Type-1 GFSs when $[n, d]$ =[4, 3] or [8, 7]. Besides, the value of DR for the same m does not affect the length of integral distinguishers when the F-function is bijective.

Table 4. Integral distinguishers for improved Type-1 GFS

m	P	DR	ID	IND [12]	IND for $[n,m,d,P]$-Type-1 GFS					
					$[n, d]$=[4,3]		$[n, d]$=[8,7]		$[n, d]$=[16,3]	
				bij	bij	nbij	bij	nbij	bij	nbij.
4	$\{2, 0, 3, 1\}$	10	19	16	19	18	19	18	23	23
5	$\{2, 0, 3, 4, 1\}$	17	29	25	29	28	29	28	34	34
	$\{2, 3, 1, 4, 0\}$	14	29	21	29	27	29	27	32	32
6	$\{2, 0, 3, 4, 5, 1\}$	26	41	36	41	40	41	40	47	47
7	$\{2, 0, 3, 4, 5, 6, 1\}$	37	55	49	55	54	55	54	62	62
	$\{2, 3, 4, 5, 1, 6, 0\}$	27	55	37	55	52	55	52	59	59
8	$\{2, 0, 3, 4, 5, 6, 7, 1\}$	50	71	64	71	70	71	70	79	79
	$\{2, 3, 4, 5, 1, 6, 7, 0\}$	38	71	50	71	68	71	68	77	77

4 Applications to LBlock and TWINE

Although our search algorithm is generic, it can improve integral distinguishers for specific ciphers. We construct several 16-round integral distinguishers for LBlock and TWINE, which directly leads to the extension of the numbers of attacked rounds for integral attack.

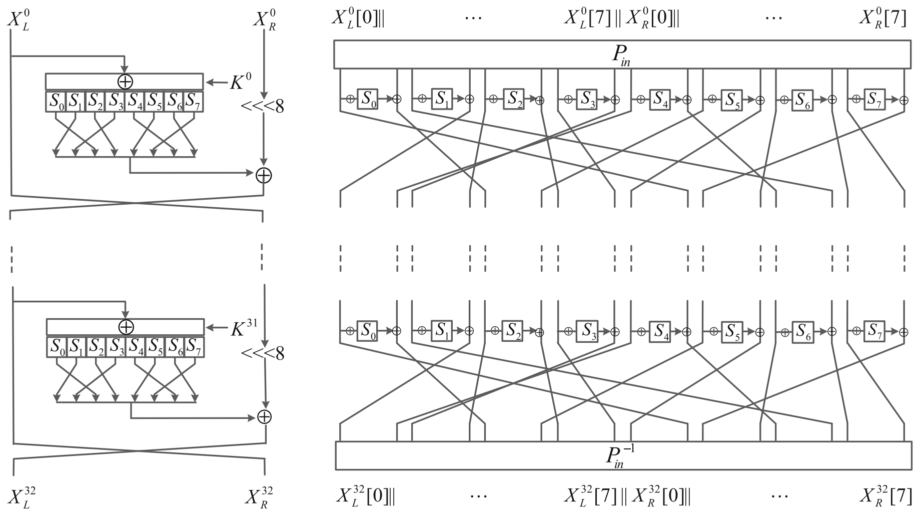
4.1 Integral Attack on LBlock

LBlock is a 32-round lightweight block cipher with 64-bit block and 80-bit master key. It adopts a Feistel structure with a twist: an 8-bit rotation is performed on the branch being XOR with the output of the Feistel function. The Feistel function is made of a key addition, a S-box layer and a nibble permutation. We denote $X_L^i||X_R^i$ the internal state which is the input to the i-th round (or the output from $(i-1)$-th round), and further describe 8 nibbles inside of X_L^i and X_R^i as $X_L^i = X_L^i[0]||X_L^i[1]\cdots||X_L^i[7]$ and $X_R^i = X_R^i[0]||X_R^i[1]\cdots||X_R^i[7]$, respectively. A plaintext is load into the state $X_L^0||X_R^0$ which is then processed as Fig. 5 (left), and finally, $X_L^{32}||X_R^{32}$ is produced as the ciphertext.

Keyschedule. The keyschedule generates 32 round keys from the master key. Firstly, the master key is loaded to a key register, denoted by $K = k_{79}k_{78}\cdots k_1k_0$. After that, extract leftmost 32 bits of current content of the register as round key K^0. And then update the key register as follow:

1. $K <<< 29$
2. $[k_{79}k_{78}k_{77}k_{76}] = S_9[k_{79}k_{78}k_{77}k_{76}]$, $[k_{75}k_{74}k_{73}k_{72}] = S_8[k_{75}k_{74}k_{73}k_{72}]$
3. $[k_{50}k_{49}k_{48}k_{47}k_{46}] \oplus [i]_2$
4. Output the left most 32 bits of the register as round key K^i

where S_8 and S_9 are 4-bit S-boxes, and $[i]_2$ is the binary form of i for $1 \le i \le 31$.



X_L^0
X_R^0
K^0
$<<<8$
S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
K^{31}
X_L^{32}
X_R^{32}
$X_L^0[0]\|$ ⋯ $X_L^0[7]\|X_R^0[0]\|$ ⋯ $X_R^0[7]$
P_{in}
P_{in}^{-1}
$X_L^{32}[0]\|$ ⋯ $X_L^{32}[7]\|X_R^{32}[0]\|$ ⋯ $X_R^{32}[7]$

Fig. 5. LBlock (left) and the equivalent representation (right)

Improved Integral Distinguishers. As shown in Fig. 5, LBlock is equal to a $[4, 16, 3, \{9, 4, 13, 0, 3, 6, 7, 2, 1, 12, 5, 8, 11, 14, 15, 10\}]$-Type-2 GFS cipher, except a shuffle and its inverse is applied to the plaintext and the ciphertext, respectively, where the shuffle P_{in} is $\{0, 2, 4, 6, 8, 10, 12, 14, 9, 13, 3, 7, 1, 5, 11, 15\}$. Therefore, we construct several 16-round integral distinguishers for LBlock by Algorithm 1, which improves the 15-round distinguisher proposed by designers in [11]. For example, choosing a set of 2^{63} plaintexts which are constant at one bit and are active at other 63 bits, then the state X_R^{16} is balanced.

Key Recovery. Appending 7 rounds to the integral distinguisher, we can attack 23-round LBlock with 2^{76} encryption, 2^{63} chosen plaintexts and 2^{67} bytes memory, which improved the previous best integral attack by one more round. We first give a high-level description of the key recovery.

1. Query 2^{63} plaintexts which are constant at one bit and are active at other bits.
2. Compute $\bigoplus(S(X_L^{16}[0] \oplus K^{16}[0]))$ by guessing 60-bit key.

3. Compute $\bigoplus(X_L^{17}[2])$ by guessing 40-bit key independently.
4. Find matches between two results, and get corresponding 74-bit key as key candidates.
5. For 2^{70} key candidates, we exhaustively search remaining 6-bit key to recover the master key.

Details of Step 2 is given in Appendix D. We obtain a list with 2^{60} entries which contains 64-bit information: $\bigoplus(S(X_L^{16}[0] \oplus K^{16}[0]))$ and corresponding 60-bit guessed key. This procedure costs $2^{69.8}$ 23-round encryptions. Due to the Feistel structure, Step 3 costs much less time to produce a list with 2^{40} entries, which contains 44-bit information: $\bigoplus X_L^{17}[2]$ and corresponding 40-bit guessed key $K^{22}[0,1,4,5,7]||K^{21}[0,2]||K^{20}[4,5]||K^{21}[7]_{(0)}||K^{20}[6]_{(2,3)}||K^{20}[7]_{(0)}$. In total, 78 bits key are guessed in key recovery as shown in Fig. 8, however, there exist only 74-bit significant key information, because 4-bit guessed in K^{18} can be deduced from remaining 74-bit key. Therefore, we obtain 2^{70} key candidates after Step 4. For 6-bit key is remained unknown as shown in Fig. 6, we guess it and exhaustively search the right master key combining with the key candidates.

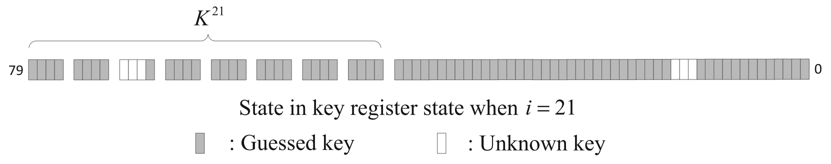


Fig. 6. Key state after key recovery

The time complexity of the attack is determined by exhaustively searching, that is $2^6 \times 2^{70} = 2^{76}$ 23-round encryptions. The data complexity is 2^{63} chosen plaintexts. Besides, we need $2^{60} \times 15 \times 4 \times 2^{-3} = 2^{61}$ bytes of memory to save 15 nibbles of ciphertext involved in the computation of $X_L^{16}[0]$.

4.2 Integral Attack on TWINE

TWINE is a Type-2 GFS block cipher with 16 branches of 4 bits each. It supports two key lengths, 80-bit and 128-bit, which we write as TWINE-80 and TWINE-128, respectively. They only differ by the key-schedule and both have 36 rounds. The i-round of TWINE is depicted in Fig. 7, where X^i is the input which is also expressed by $X^i = X^i[0]||X^i[1] \cdots ||X^i[15]$ and the S-box S is a 4-bit permutation with algebraic degree 3. We denote the j-th nibble of i-th round key K^i by $K^i[j]$ for $0 \leq j \leq 7$.

Keyschedule. The keyschedule produces 36 round keys from the master key. Firstly, the key register is initialized to the master key, and then the key register are updated by a sparse GFS using only 2 S-box per updating procedure for

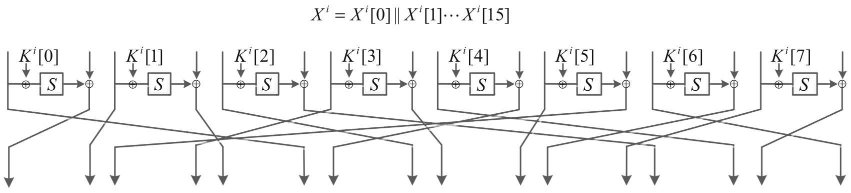


Fig. 7. Round function of TWINE

TWINE-80 and 3 for TWINE-128. Finally fixed 8 nibbles are extracted from the key register as the round key. For more details, please refer to [8].

In [8], the designers gave several 15-round integral distinguishers for TWINE. For example, considering a set of 2^{60} plaintexts which are constant for the left-most nibble (indexed by 0) and are active for other nibbles, the state after 15 rounds has 4 balanced nibbles indexed by 1, 3, 13, 15. Then, they launched the integral attack on 22-round TWINE-80 with the time, data and memory complexity being 2^{77} encryption, 2^{60} chosen plaintexts and 2^{70} bytes. The time complexity can be further reduced to $2^{68.4}$ encryption with the data complexity increased by a factor of 4. In a similar manner, 23-round TWINE-128 can be attacked with $2^{106.14}$ encryptions, $2^{62.81}$ chosen plaintexts and 2^{106} bytes memory. These results are both the best integral attacks up to now.

Improved Integral Distinguishers. We discover several 16-round integral distinguishers for TWINE by applying our path search algorithm. If we choose 2^{63} plaintexts which are constant at any one bit and are active at other 63 bits, the state after 16 encryption rounds is balanced for any nibble with odd index.

Key Recovery. Due to the keyschedule, 1-th nibble is the optimal choice for the attack considering the time complexity, therefore, we can attack 23-round TWINE-80 by following the key recovery procedure in [8] directly. Note that a structure for our distinguisher contains 2^{63} plaintexts instead of 2^{60}. Using one structure, the time, data and memory complexities of the attack are thus 2^{77} encryption, 2^{63} chosen plaintexts and 2^{70} bytes, respectively. Similarly, we can attack 24-round TWINE-128 with 2^{124} encryption, 2^{63} chosen plaintexts and 2^{106} bytes.

5 Conclusion

In this paper, we first studied the propagation characteristic of the division property for the GFS, and then proposed a generic algorithm of searching the integral distinguishers. Meanwhile, we devised the early reduce technique, which is useful to optimize the time and memory complexities. By using our algorithm, we evaluated the security of the GFS. The results show that the length of integral distinguishers can be extended by at least $m - 1$ and 1 rounds for Type-1 and

Type-2 GFS with m branches, respectively. For improved Type-1 and Type-2 GFSs, distinguishers depend on the specific parameters of the structure, such as m, branch size, algebraic degree of F-functions, permutation layer and whether F-functions are bijective or not. Finally, the algorithm was applied to LBlock and TWINE. We constructed several 16-round integral distinguishers, which lead to the integral attacks on 23-round LBlock, 23-round TWINE-80 and 24-round TWINE-128.

Acknowledgments. We would like to thank the anonymous reviewers for their useful comments and suggestions. The research presented in this paper is supported by the National Basic Research Program of China (No. 2013CB338002) and National Natural Science Foundation of China (No. 61272476, No. 61232009 and No. 61202420).

A Proofs for Proposition 2-4

Proposition 2

Proof. We give the proof for the case of $m = 4$, which can then simply be transferred to the general case. Denote Λ and Λ' as the multi-set of inputs and outputs, respectively. The checksum of Λ' for $U = (u_0, \cdots, u_5)$ has

$$\begin{aligned}
&\bigoplus_{Y\in\Lambda'} \pi_U(Y)\\
&= \bigoplus_{X\in\Lambda} \pi_{(u_0,\ldots,u_5)}(x_0, x_0, x_1, x_2, x_2, x_3)\\
&= \bigoplus_{X\in\Lambda} \pi_{u_0}(x_0) \times \pi_{u_1}(x_0) \times \pi_{u_2}(x_1) \times \pi_{u_3}(x_2) \times \pi_{u_4}(x_2) \times \pi_{u_5}(x_3)\\
&= \bigoplus_{X\in\Lambda} \pi_{u_0\vee u_1}(x_0) \times \pi_{u_2}(x_1) \times \pi_{u_3\vee u_4}(x_2) \times \pi_{u_5}(x_3)\\
&= \bigoplus_{X\in\Lambda} \pi_{(u_0\vee u_1, u_2, u_3\vee u_4, u_5)}(X)
\end{aligned}$$

where $\vee$ is OR. When $(w(u_0 \vee u_1), w(u_2), w(u_3 \vee u_4), w(u_5)) \not\succeq K$, the result is always 0. Its sufficient condition is $(w(u_0) + w(u_1), w(u_2), w(u_3) + w(u_4), w(u_5)) \not\succeq K$. Therefore, the division property of Λ' is $\{(i_0, k_0 - i_0, k_1, i_1, k_2 - i_1, k_3) | 0 \le i_0 \le k_0, 0 \le i_1 \le k_2\}$.

Proposition 3

Proof. The proposition describes a special case of Rule 1 in [9]. Readers can refer to [9] for the details of the proof.

Proposition 4

Proof. We prove the case when $m = 4$. The general case follows by a similar manner. Denote Λ and Λ' as the multi-set of inputs and the multi-set of outputs,

Table 5. Integral distinguishers for improved Type-1 GFS with $8 < m \leq 16$

m	P	DR	ID	IND [12]	Our IND			
					$[n, d]$=[4, 3]		$[n, d]$=[8, 7]	
				bij	bij	nbij	bij	nbij.
9	$\{2,3,1,4,5,6,7,8,0\}$	58	89	73	89	87	89	87
	$\{2,3,4,5,6,7,1,8,0\}$	44	89	57	89	85	89	85
10	$\{2,3,4,5,1,6,7,8,9,0\}$	66	109	82	109	106	109	106
11	$\{2,3,1,4,5,6,7,8,9,10,0\}$	92	131	111	131	129	131	129
	$\{2,3,4,5,1,6,7,8,9,10,0\}$	83	131	101	131	128	131	128
	$\{2,3,4,5,6,7,1,8,9,10,0\}$	74	131	91	131	127	131	127
	$\{2,3,4,5,6,7,8,9,1,10,0\}$	65	131	81	131	126	131	126
12	$\{2,3,4,5,6,7,8,9,1,10,11,0\}$	82	155	100	155	150	155	150
13	$\{2,3,1,4,5,6,7,8,9,10,11,12,0\}$	134	181	157	181	179	181	179
	$\{2,3,4,5,1,6,7,8,9,10,11,12,0\}$	123	181	145	181	178	181	178
	$\{2,3,4,5,6,7,1,8,9,10,11,12,0\}$	112	181	133	181	177	181	177
	$\{2,3,4,5,6,7,8,9,1,10,11,12,0\}$	101	181	121	181	176	181	176
	$\{2,3,4,5,6,7,8,9,10,11,1,12,0\}$	90	181	109	181	175	181	175
14	$\{2,3,4,5,1,6,7,8,9,10,11,12,13,0\}$	146	209	170	209	206	209	206
	$\{2,3,4,5,6,7,8,9,1,10,11,12,13,0\}$	122	209	144	209	204	209	204
15	$\{2,3,1,4,5,6,7,8,9,10,11,12,13,14,0\}$	184	239	211	239	236	239	236
	$\{2,3,4,5,6,7,1,8,9,10,11,12,13,14,0\}$	158	239	183	239	234	239	234
	$\{2,3,4,5,6,7,8,9,10,11,12,13,1,14,0\}$	119	239	141	239	232	239	232
16	$\{2,3,4,5,1,6,7,8,9,10,11,12,13,14,15,0\}$	198	271	211	271	268	271	268
	$\{2,3,4,5,6,7,8,9,1,10,11,12,13,14,15,0\}$	170	271	183	271	266	271	266
	$\{2,3,4,5,6,7,8,9,10,11,12,13,1,14,15,0\}$	142	271	141	271	264	271	264

respectively. The checksum of Λ' for $U = (u_0, \cdots, u_3)$ has

$$\begin{aligned}
&\bigoplus_{X' \in \Lambda'} \pi_{(u_0,u_1,u_2,u_3)}(X') \\
&= \bigoplus_{Z \in \Lambda} \pi_U((z_0, z_1 \oplus z_2, z_3, z_4 \oplus z_5)) \\
&= \bigoplus_{Z \in \Lambda} \pi_{u_0}(z_0) \times \pi_{u_1}(z_1 \oplus z_2) \times \pi_{u_2}(z_3) \times \pi_{u_3}(z_4 \oplus z_5) \\
&= \bigoplus_{Z \in \Lambda} \left(\pi_{u_0}(z_0) \times (\bigoplus_{c_1 \prec u_1} \pi_{u_1}(z_1) \times \pi_{u_1 \oplus c_1}(z_2)) \times \pi_{u_2}(z_3) \times (\bigoplus_{c_2 \prec u_3} \pi_{u_3}(z_4) \times \pi_{u_3 \oplus c_2}(z_5)) \right) \\
&= \bigoplus_{Z \in \Lambda} \bigoplus_{c_1 \prec u_1} \bigoplus_{c_2 \prec u_3} \left(\pi_{(u_0, u_1, u_1 \oplus c_1, u_2, u_3, u_3 \oplus c_2)}(Z) \right)
\end{aligned}$$

where $c \prec u$ denotes the elements of F_2^n satisfying c AND u equals to c. Obviously, it has $w(c) + w(u \oplus c) = w(u)$ if $c \prec u$. When $(w(u_0), w(u_1), w(u_1) - w(c_1), w(u_2), w(u_3), w(u_3) - w(c_2)) \nsucceq K$ for any $c_1 \prec u_1$ and any $c_2 \prec u_3$, the result is always 0. Thereafter, the division property of Λ' is $\{K'^{(j)} = (k_0^j, (k_1^j + k_2^j), k_3^j, (k_4^j + k_5^j)) | 0 \leq j \leq q - 1\}$.

B Results on Improved Type-1 GFS for $8 < m \leq 16$

Table 5 shows integral distinguishers for improved Type-1 GFS when the number of branches is more than 8.

C Results on Improved Type-2 GFS for $8 < m \leq 16$

Table 6 shows integral distinguishers for improved Type-2 GFS when the number of branches is more than 8.

Table 6. Integral distinguishers for improved Type-2 GFS with $8 < m \leq 16$

m	Type	P	DR	IND [7]	Our IND			
					$[n, d]$=[4,3]		$[n, d]$=[8, 7]	
				bij	bij	nbij	bij	nbij.
10	No. 1	{5, 0, 7, 2, 9, 6, 3, 8, 1, 4}	7	13	14	14	14	13
	No. 2	{3, 0, 1, 4, 7, 2, 5, 8, 9, 6}	7	13	15	14	14	14
	No. 3	{3, 0, 7, 4, 1, 6, 5, 8, 9, 2}	7	13	14	13	13	13
12	No. 1	{3,0,7,2,9,4,11,8,5,10,1,6}	8	15	17	16	16	16
	No. 2	{3,0,7,2,11,4,1,8,5,10,9,6}	8	16	17	17	17	16
	No. 3	{7,0,9,2,11,4,1,8,5,10,3,6}	8	15	16	15	16	15
	No. 4	{5,0,9,2,1,6,11,4,3,10,7,8}	8	15	17	17	16	16
14	No. 1	{1,2,9,4,3,6,13,8,7,10,11,12,5,0}	8	15	17	16	16	16
	No. 2	{1,2,9,4,13,6,7,8,5,10,3,12,11,0}	8	15	16	15	16	15
	No. 14	{1,2,11,4,13,6,7,8,5,12,9,10,3,0}	8	15	16	16	16	16
	No. 16	{5,2,9,4,1,6,13,10,11,8,7,0,3,12}	8	15/16	17	17	17	16
	No. 20	{7,2,1,4,9,6,5,10,3,12,13,0,11,8}	8	15	16	16	16	15
16	No. 1	{1,2,9,4,15,6,5,8,13,10,7,14,11,12,3,0}	8	16	17	16	17	16
	No. 7	{1,2,11,4,3,6,7,8,15,12,5,14,9,0,13,10}	8	15	17	16	16	16
	No. 10	{7,2,13,4,11,8,3,6,15,0,9,10,1,14,5,12}	8	15	16	15	16	15

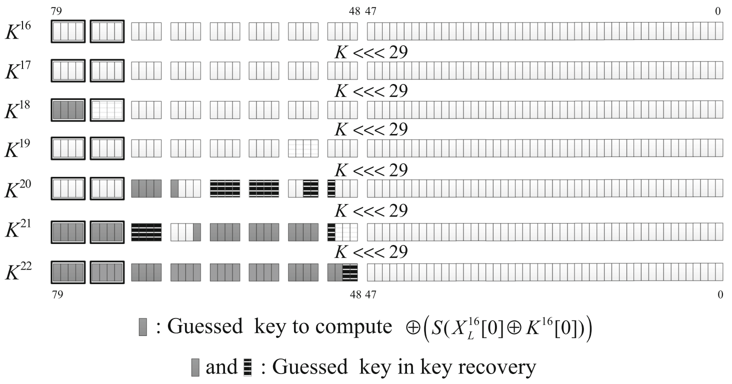
D Details of the Attack on LBlock

We need to guess 60-bit key to compute the value of $\bigoplus(S(X_L^{16}[0] \oplus K^{16}[0]))$ according to the keyschedule. These guessed keys are marked by gray cubes in Fig. 8, and the procedure is as follows:

1. Query 2^{63} plaintexts which are constant at one bit and are active at other bits.
2. Count whether each 15-nibble value $X_L^{23}[0, 1, 2, 3, 4, 6, 7]||X_R^{23}[0, 1, 2, 3, 4, 5, 6, 7]$ appears even or odd times, and pick the values which appear odd times.
3. Guess $K^{22}[3]$, and then compute $X_R^{22}[3]$. Compress the data into 2^{56} texts of the value of $X_L^{23}[0, 2, 3, 4, 6, 7]||X_R^{23}[0, 1, 2, 4, 5, 6, 7]||X_R^{22}[3]$ appearing odd times.

4. Guess $K^{22}[5]$, and then compute $X_R^{22}[6]$. Compress the data into 2^{52} texts of the value of $X_L^{23}[0,2,3,6,7]||X_R^{23}[0,1,2,4,6,7]||X_R^{22}[3,6]$ appearing odd times.
5. Guess $K^{22}[1]$, and then compute $X_R^{22}[2]$. Compress the data into 2^{48} texts of the value of $X_L^{23}[2,3,6,7]||X_R^{23}[0,2,4,6,7]||X_R^{22}[3,6,2]$ appearing odd times.
6. Guess $K^{21}[6]$, and then compute $X_R^{21}[1]$. Compress the data into 2^{44} texts of the value of $X_L^{23}[2,3,6,7]||X_R^{23}[0,2,4,6]||X_R^{22}[3,2]||X_R^{21}[1]$ appearing odd times.
7. Guess $K^{22}[4]$, and then compute $X_R^{22}[0]$. Compress the data into 2^{44} texts of the value of $X_L^{23}[2,3,7]||X_R^{23}[0,2,4,6]||X_R^{22}[0,2,3]||X_R^{21}[1]$ appearing odd times.
8. Guess $K^{22}[6]$, and then compute $X_R^{22}[1]$. Compress the data into 2^{44} texts of the value of $X_L^{23}[2,3]||X_R^{23}[0,2,4,6]||X_R^{22}[0,1,2,3]||X_R^{21}[1]$ appearing odd times.
9. Guess $K^{22}[0]$, and then compute $X_R^{22}[4]$. Compress the data into 2^{44} texts of the value of $X_L^{23}[3]||X_R^{23}[0,2,4,6]||X_R^{22}[0,1,2,3,4]||X_R^{21}[1]$ appearing odd times.
10. Guess $K^{21}[4]$, and then compute $X_R^{21}[0]$. Compress the data into 2^{40} texts of the value of $X_L^{23}[3]||X_R^{23}[0,2,4]||X_R^{22}[0,1,2,3]||X_R^{21}[0,1]$ appearing odd times.
11. Due to the keyschedule, $K^{20}[0]$ is determined by rightmost two bits in $K^{22}[5]$ and leftmost two bits in $K^{22}[6]$, which are all guessed. We can directly compute $X_R^{20}[4]$ and compress the data into 2^{36} texts of the value of $X_L^{23}[3]$ $||X_R^{23}[0,2,4]||X_R^{22}[0,1,3]||X_R^{21}[1]||X_R^{20}[4]$ appearing odd times.
12. Due to the keyschedule, $K^{20}[1]$ is determined by rightmost two bits in $K^{22}[6]$ and leftmost two bits in $K^{22}[7]$. We only need guess the leftmost two bits in $K^{22}[7]$. Compute $X_R^{20}[2]$ and compress the data into 2^{36} texts of the value of $X_L^{23}[3]||X_R^{23}[0,2,4]$ $||X_R^{22}[0,1,3]||X_R^{20}[2,4]$ appearing odd times.
13. Guess $K^{21}[1]$ and compute $X_R^{21}[2]$ and compress the data into 2^{32} texts of the value of $X_L^{23}[3]||X_R^{23}[2,4]||X_R^{22}[0,3]||X_R^{21}[2]||X_R^{20}[2,4]$ appearing odd times.
14. Guess $K^{20}[2]$ and compute $X_R^{20}[5]$ and compress the data into 2^{28} texts of the value of $X_L^{23}[3]||X_R^{23}[2,4]||X_R^{22}[0]||X_R^{20}[2,4,5]$ appearing odd times.
15. Guess $K^{21}[0]$ and compute $X_R^{21}[4]$ and compress the data into 2^{28} texts of the value of $X_L^{23}[3]||X_R^{23}[2,4]||X_R^{21}[4]||X_R^{20}[2,4,5]$ appearing odd times.
16. Due to the keyschedule, $K^{19}[5]$ is determined by $K^{22}[3]$ and $K^{22}[4]$. Compute $X_R^{19}[6]$ and compress the data into 2^{24} texts of the value of $X_L^{23}[3]||X_R^{23}$ $[2,4]||$ $X_R^{20}[2,4]$ $||X_R^{19}[6]$ appearing odd times.
17. Guess $K^{22}[2]$ and then compute $X_R^{22}[5]$ and compress the data into 2^{20} texts of the value of $X_R^{22}[5]||X_R^{23}[4]||X_R^{20}[2,4]||X_R^{19}[6]$ appearing odd times.
18. Guess $K^{21}[5]$ and then compute $X_R^{21}[6]$ and compress the data into 2^{16} texts of the value of $X_R^{21}[6]||X_R^{20}[2,4]||X_R^{19}[6]$ appearing odd times.
19. Due to the keyschedule, $K^{19}[4]$ is determined by $K^{22}[2]$ and $K^{22}[3]$. Compute $X_R^{19}[0]$ and compress the data into 2^{12} texts of the value of $X_R^{20}[2]||X_R^{19}$ $[0,6]$ appearing odd times.

20. Guess $K^{18}[0]$ and compute $X_R^{18}[4]$ and compress the data into 2^8 texts of the value of $X_R^{19}[6]||X_R^{18}[4]$ appearing odd times.
21. Due to the keyschedule, $K^{17}[4]$ is determined by the rightmost three bits of $K^{20}[2]$ and the leftmost bit of $K^{20}[3]$. Guess the leftmost bit of $K^{20}[3]$, compute $X_R^{17}[0]$ and compress the data into 2^4 texts of the value of $X_R^{17}[0]$ appearing odd times.
22. Due to the keyschedule, $K^{16}[0]$ is determined by the rightmost bit of $K^{21}[3]$ and the leftmost three bits of $K^{21}[4]$. Guessing the rightmost bit of $K^{21}[3]$, and then we compute the sum $\bigoplus(S(X_L^{16}[0] \oplus K^{16}[0]))$.



79
48 47
0
K^{16}
K^{17}
K^{18}
K^{19}
K^{20}
K^{21}
K^{22}
$K <<< 29$
▮ : Guessed key to compute $\oplus(S(X_L^{16}[0] \oplus K^{16}[0]))$
▮ and ≣ : Guessed key in key recovery

Fig. 8. Key state for key recovery

Complexity for Computing $\bigoplus(S(X_L^{16}[0] \oplus K^{16}[0]))$. The complexity for each step is estimated as a product of the previous date size and the total number of guessed bits. In total,

$$\begin{aligned}
&2^4 \times 2^{60} + 2^8 \times 2^{56} + 2^{12} \times 2^{52} + 2^{16} \times 2^{48} + 2^{20} \times 2^{44} \\
&+2^{24} \times 2^{44} + 2^{28} \times 2^{44} + 2^{32} \times 2^{44} + 2^{32} \times 2^{40} + 2^{34} \times 2^{36} \\
&+2^{38} \times 2^{36} + 2^{42} \times 2^{32} + 2^{46} \times 2^{28} + 2^{46} \times 2^{28} + 2^{50} \times 2^{24} \\
&+2^{54} \times 2^{20} + 2^{54} \times 2^{16} + 2^{58} \times 2^{12} + 2^{59} \times 2^8 + 2^{40} \times 2^4 \\
&= 2^{77.3}.
\end{aligned}$$

That is $2^{77.3} \times \frac{1}{8} \times \frac{1}{23} \approx 2^{69.8}$ 23-round encryptions. After step 22, we obtain a list with 2^{60} entries which contains 64-bit information: $\bigoplus(S(X_L^{16}[0] \oplus K^{16}[0]))$ and corresponding 60-bit guessed key.

References

1. Adams, C.: The CAST-256 encryption algorithm. In: AES proposal (1998)
2. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)

3. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, p. 112. Springer, Heidelberg (2002)
4. Nyberg, K.: Generalized Feistel networks. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 91–104. Springer, Heidelberg (1996)
5. Shibutani, K.: On the diffusion of generalized Feistel structures regarding differential and linear cryptanalysis. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 211–228. Springer, Heidelberg (2011)
6. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA (extended abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
7. Suzaki, T., Minematsu, K.: Improving the generalized Feistel. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 19–39. Springer, Heidelberg (2010)
8. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: a lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer, Heidelberg (2013)
9. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 413–432. Springer, Heidelberg (2015)
10. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015)
11. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)
12. Yanagihara, S., Iwata, T.: On permutation layer of Type 1, Source-Heavy, and Target-Heavy generalized Feistel structures. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 98–117. Springer, Heidelberg (2011)