Delving Deep into Security Guarantees against Integral Distinguishers

with Applications to PRESENT, TWINE and LBLOCK

Shuo Peng[1,2,3], Jiahui He[1,2], Kai Hu[1,2,3*], Meiqin Wang[1,2,3]

[1]School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China.
[2]Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China.
[3]Quan Cheng Shandong Laboratory, Jinan, China.

*Corresponding author(s). E-mail(s): kai.hu@sdu.edu.cn;
Contributing authors: pengshuo@mail.sdu.edu.cn;
hejiahui2020@mail.sdu.edu.cn; mqwang@sdu.edu.cn;

Abstract

Integral attacks pose a significant threat to block cipher security, yet providing guarantees against such attacks for a target block cipher is difficult. At ASIACRYPT 2021, Hebborn, Lambin, Leander, and Todo proposed the integral-resistance property, which offers strong security guarantees for certain SPN and AND-RX block ciphers, assuming independent round keys. However, limitations remain: they proved a security bound for 13-round Present, while the longest known integral distinguisher covers only 9 rounds. Further, their method cannot tackle complex Feistel structures such as Twine and Lblock. A major challenge in their method is the difficulty of finding key monomials that lead to odd-number monomial trails. We observe that in the first and last parts of the target cipher, many *interfering monomials* exist that always produce *interfering trails*, which is a critical reason that makes it difficult to find odd-number monomial trails. Fortunately, we find that these interfering monomials are avoidable by a careful selection of the key monomials. Using this insight, we successfully prove the security of 11-round Present, improving the previous result by 2 rounds, and provide a partial analysis for 10-round Present. We also extend their integral-resistance property to general-Feistel-network (GFN) ciphers Twine and Lblock by proposing an equivalent key transformation method. Through acceleration strategies for identifying key monomials, we confirm, for the first time, that 20-round Twine

(out of 36 rounds) and Lblock (out of 32 rounds) are resistant to integral distinguishers. We believe our observations and strategies provide gains to Hebborn et al.'s security guarantees for block ciphers.

Keywords: Division property, Integral distinguisher, Integral-resistance property , Present , Twine, Lblock

Acknowledgements. We are grateful to the anonymous reviewers for their thorough review and insightful comments, which helped us strengthen the technical presentation and correctness of this work. This research is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the National Natural Science Foundation of China (Grant Nos. 62032014, U2336207, 62402283), Department of Science & Technology of Shandong Province (Grant No. SYS202201), NationalKey R&D Program of China (Grant No. 2023YFA1009500), Quan Cheng Laboratory (Grant Nos. QCLZD202301, QCLZD202306). Kai Hu is also supported by the Program of Qilu Young Scholars of Shandong University.

1 Introduction

Block ciphers are the most widely used primitives for encrypting data and serve as fundamental building blocks for advanced cryptographic primitives such as stream ciphers, hash functions, and message authentication codes. It is crucial to provide guarantees for the security of block ciphers when possible. Integral cryptanalysis [1, 2] is a powerful attack towards block ciphers, which has been well demonstrated by breaking some ciphers such as Misty1 [3, 4]. Integral attacks identify a subset of plaintexts that result in a constant value when their corresponding ciphertexts are summed. From an algebraic perspective, integral attacks on a cipher strongly relate to the Algebraic Normal Forms (ANFs). When some monomials are missing or coefficients of some monomials are linearly dependent in the ANFs of a block cipher, the block cipher is vulnerable to integral attacks. At ASIACRYPT 2021, Hebborn, Lambin, Leander, and Todo proposed a rigorous argument, called the *integral-resistance* property, which can demonstrate that a (round-reduced) block cipher does not have any integral distinguishers under the assumption of independent round keys [5]. Most recently, in [6], the security arguments for the case of adding keys by modular addition were proposed.

The proof of integral-resistance property proposed in [5] requires that the block cipher has whitening keys at the beginning. For the remaining part of the cipher, it requires that the ANFs of all ciphertext bits or their linear combinations contain all monomials of degree $n-1$ (where n is the block size). Verifying whether the ANFs of ciphertext bits satisfy the integral-resistance property is challenging since the expressions of the ANFs are complex and not readily available. Fortunately, division properties proposed by Todo [7] have evolved into an effective and accurate method of probing the structure of Boolean functions. *The three-subset bit-based division property without unknown subsets*, implied by Wang et al. in [8] and Hao et al. in [9, 10], can accurately determine whether a monomial appears or disappears in an ANF that is not

directly accessible. Other views of the division properties, such as *parity sets* [11] and *monomial prediction* [12], can also provide precise predictions. Most recently, Beyne et al. [13] introduced the concept of *algebraic transition matrices* and its generalization, *ultrametric integral cryptanalysis* [14]. This latter approach lifts the theory of integral attacks to a field of characteristic zero and is able to describe more general divisibility properties than just divisibility by 2 (as it is used in classical integral cryptanalysis).

In [5], the authors proved the integral-resistance property for Present, Skinny-64, Craft, Gift-64, Simon, and Simeck by parity sets. However, their approach is still not perfect. For example, they provided the integral-resistance property for 13-round Present while the longest integral distinguisher of Present reaches only 9 rounds. Also, their method is limited within SPN and AND-RX ciphers. It is still unknown whether it is applicable to other types of ciphers such as the general Feistel network (GFN) ciphers. Since the security guarantees against integral attacks for block ciphers are crucial, it is interesting and important to improve this method further.

Contributions. The most important step in the proof [5] is to find key monomials with an odd-number of monomial trails connecting certain plaintext and ciphertext monomials. However, in many cases, the key monomials under check always lead to even-number trails. We study the reason for the appearance of even-number trails. When we select a key monomial for the first (resp. last) round which plays a critical role in the following search, it is very likely that two state monomials, say t_0 and t_1 will appear along with the key monomial we selected. We can choose one, e.g., t_0, as a candidate to construct the final key monomial; t_1 usually causes interference, i.e., the trails attributed to t_1 would cancel trails attributed to t_0. Thus, we call t_1 the *interfering monomial* of t_0 (dually, t_0 is also an interfering monomial of t_1). In [5], such a phenomenon was not considered.

Luckily, by imposing some constraints on the key monomials of intermediate rounds, we can easily control the appearance of t_0 and t_1[1]. Therefore, the interfering monomial will be avoided. The probability of finding a key monomial with odd-number of trails increases significantly. Applying this observation and strategy to Present, we easily proved the integral-resistance property for 12 rounds and 11 rounds (the previous results are stuck at 13 rounds [5]). For 10-round Present, a weaker result is provided where we prove that 63 out of 64 output bits do not give constant-sum. Specifically, for any subset of plaintexts, the sum of any of these 63 output bits (excluding the 0-th bit) is dependent on the key.

Furthermore, apart from SPN and AND-RX block ciphers, we extend the integral-resistance technique to GFN block ciphers Twine and Lblock. Since the GFN block cipher may lack a whitening key and the round keys are XORed only with half of the state, we introduce an equivalent key transformation method that generates an equivalent cipher, to which the integral-resistance property can be directly applied. The equivalent cipher can be regarded as a SPN structure with a Supersbox layer. Ensuring the equivalent cipher is resistant to integral attacks implies that the original block cipher is also resistant. Due to the complexity of the round function in the equivalent cipher, searching for the required monomial trails is challenging. Therefore,

[1] We do not restrict the number of interfering monomials to two—there may be more in general—but in our experiments, it is typically two.

we adopt a backtracking search strategy to accelerate the process. Besides, we further accelerate the search by leveraging the Supersbox structure. Finally, we manage to confirm that 20-round Twine (out of 36 rounds) and Lblock (out of 32 rounds) do not have any integral distinguishers.

Comparison with [15]. A concurrent work by Zeng and Tian [15] addresses the problem of proving the integral-resistance property for 5-round AES, where the complexity of the AES Sbox and linear layer makes it difficult to construct a full-rank integral-resistance matrix. Both their work and ours were developed independently and target different aspects of the problem. In their work, by carefully investigating the properties of the AES Sbox and its linear layer, they reduce the search space of division trails. In contrast, our work focuses on eliminating the influence of interfering monomials and on proving the integral-resistance property for ciphers with GFN structure. We present generic arguments showing that several other round-reduced block ciphers, including Present (with simple Sbox and linear layer), Twine, and Lblock (with GFN structure), satisfy the integral-resistance property.

Outline. This paper is organized as follows: Section 2 provides a brief overview of notations and definitions. Section 3 outlines Hebborn et al.'s work. Section 4 gives our improved strategies and results of Present. Section 5 details how we ensure the integral-resistance property of Twine. Section 6 provides security guarantees against the integral attack of Lblock, and Section 7 concludes the paper. Appendices are provided at the end. We conduct our experiments on a server equipped with 36 Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz and 534GB of memory. Detailed results and the source code can be found at https://github.com/shuo-peng/refinedISAID.

2 Preliminaries

2.1 Notations and Definitions

We denote by $\mathbb{F}_2$ the finite field with two elements, and by $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. We use lowercase letters to represent bit vectors. Blackboard bold uppercase letters (e.g., $\mathbb{M}$) are used to represent sets of bit vectors. The Cartesian product $\mathbb{F}_2^n \times \mathbb{F}_2^m$ denotes the set of all ordered pairs (u, u') where $u \in \mathbb{F}_2^n$ and $u' \in \mathbb{F}_2^m$.

$u[i]$ denotes the i-th entry of the vector u, and $u[i:j]$ denotes the subvector of u from position i to position j. For an n-bit vector $u = (u[0], \ldots, u[n-1]) \in \mathbb{F}_2^n$, the Hamming weight of u is denoted by $\mathrm{wt}(u) = \sum_{i=0}^{n-1} u[i]$, where $u[i]$ is treated as an integer. We denote by 0^n and 1^n the all-zero and all-one vectors of length n, respectively. The vector e_i refers to the n-bit unit vector whose i-th bit is 1 and all other bits are 0, while $\bar{e}_i$ denotes the n-bit vector whose i-th bit is 0 and all other bits are 1.

For any two n-bit vectors u and u', we write $u \succeq u'$ if $u[i] \geq u'[i]$ for all i, and $u \preceq u'$ if $u[i] \leq u'[i]$ for all i. The inner product of u and u' is denoted by $\langle u, u' \rangle$ and defined as $\sum_{i=0}^{n-1} u[i]u'[i]$.

Boolean Function. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function whose ANF is

$$f(x) = f(x[0], x[1], \ldots, x[n-1]) = \sum_{u \in \mathbb{F}_2^n} a_u x^u = \sum_{u \in \mathbb{F}_2^n} a_u \prod_{i=0}^{n-1} x[i]^{u[i]},$$

where $a_u \in \mathbb{F}_2$, and

$$x^u = \prod_{i=0}^{n-1} x[i]^{u[i]} \text{ with } x[i]^{u[i]} = \begin{cases} x[i], & \text{if } u[i] = 1, \\ 1, & \text{if } u[i] = 0, \end{cases}$$

is called a monomial. If the coefficient of x^u in f is 1, i.e., x^u is contained by f, then we denote it by $x^u \to f$. Otherwise, we denote the absence of x^u in f by $x^u \nrightarrow f$.

Vectorial Boolean Function. Let $\boldsymbol{f} : \mathbb{F}_2^m \to \mathbb{F}_2^n$ be a vectorial Boolean function with $y = (y[0], y[1], \ldots, y[n-1]) = \boldsymbol{f}(x) = (f_0(x), f_1(x), \ldots, f_{n-1}(x))$. For $v \in \mathbb{F}_2^n$, we use y^v to denote the product of some coordinates of y :

$$y^v = \prod_{i=0}^{n-1} y[i]^{v[i]} = \prod_{i=0}^{n-1} f_i(x)^{v[i]},$$

which is a Boolean function of x.

2.2 Integral Attack

For a block cipher E_k with n-bit plaintexts and ciphertexts and m-bit keys, each output bit can be represented by a Boolean function of the plaintext x and the key k. When it comes to the integral attack, we are concerned about all linear combinations of the output bits (note that every single output bit is also included in this form), which can be obtained by computing an inner product of a linear mask $\beta \in \mathbb{F}_2^n$ and the output. The ANF of the linear combination of the output bits is denoted by

$$\langle \beta, E_k(x) \rangle = \sum_{u \in \mathbb{F}_2^n, v \in \mathbb{F}_2^m} \lambda_{u,v}^{\beta} x^u k^v = \sum_{u \in \mathbb{F}_2^n} p_u^{\beta}(k) x^u, \tag{1}$$

where $p_u^{\beta}(k)$ are functions $p_u^{\beta} : \mathbb{F}_2^m \to \mathbb{F}_2$ mapping keys to values in $\mathbb{F}_2$. The integral attack distinguishes a block cipher from a random permutation by examining the representation of Equation 1. If the coefficient $p_u^{\beta}(k)$ of a certain monomial x^u of Equation 1 is key-independent, the sum of the linear combination of ciphertexts over the plaintext set $\mathbb{M} = \{x : x \preceq u\}$ is a constant.

2.3 Monomial Prediction

The precise versions of the division properties have been extensively studied, including parity sets [11], the three-subset bit-based division property without unknown subsets [9], monomial prediction [12] and most recently algebraic transition matrices [13].

The monomial prediction technique offers another algebraic language for Todo's division properties [7, 16], which is used to determine whether a monomial appears in any product of the coordinate functions of a vectorial Boolean function $\boldsymbol{f}$. In this paper, we choose the monomial prediction language to explain how to ensure integral-resistance property.

Let $\boldsymbol{f}:\mathbb{F}_2^{n_0}\to\mathbb{F}_2^{n_r}$ be a composite vectorial Boolean function from a sequence of vectorial Boolean functions $\boldsymbol{f}^{(i)}:\mathbb{F}_2^{n_i}\to\mathbb{F}_2^{n_{i+1}}, 0\le i\le r-1$ whose ANFs are known, i.e.,

$$\boldsymbol{f}=\boldsymbol{f}^{(r-1)}\circ\boldsymbol{f}^{(r-2)}\circ\cdots\circ\boldsymbol{f}^{(0)}.$$

Let x_i denote the state at round i. Starting from a monomial of x_0, say ${x_0}^{u_0}$, all monomials of x_1 satisfying ${x_0}^{u_0}\to{x_1}^{u_1}$ can be derived; for every such ${x_1}^{u_1}$, we then find all ${x_2}^{u_2}$ satisfying ${x_1}^{u_1}\to{x_2}^{u_2}$; Such forward expansions continue until we arrive at the monomials of x_r. Each transition from ${x_0}^{u_0}$ to ${x_r}^{u_r}$ denoted by

$${x_0}^{u_0}\to{x_1}^{u_1}\to\cdots\to{x_r}^{u_r}$$

is called a monomial trail [12], denoted by ${x_0}^{u_0}\rightsquigarrow{x_r}^{u_r}$, which is also used to indicate the existence of at least one monomial trail from ${x_0}^{u_0}$ to ${x_r}^{u_r}$. The set of all trails from ${x_0}^{u_0}$ to ${x_r}^{u_r}$ is denoted by ${x_0}^{u_0}\bowtie{x_r}^{u_r}$. Whether ${x_0}^{u_0}\to{x_r}^{u_r}$ is determined by the size of ${x_0}^{u_0}\bowtie{x_r}^{u_r}$, represented as $|\,{x_0}^{u_0}\bowtie{x_r}^{u_r}\,|$. If there is no trail from ${x_0}^{u_0}$ to ${x_r}^{u_r}$, we say ${x_0}^{u_0}\not\rightsquigarrow{x_r}^{u_r}$ and accordingly $|{x_0}^{u_0}\bowtie{x_r}^{u_r}|=0$.

Lemma 1 ([12, 17]). *Let $\boldsymbol{f}=\boldsymbol{f}^{(r-1)}\circ\boldsymbol{f}^{(r-2)}\circ\cdots\circ\boldsymbol{f}^{(0)}$ defined as above. ${x_0}^{u_0}\to{x_r}^{u_r}$ if and only if*

$$|{x_0}^{u_0}\bowtie{x_r}^{u_r}|\equiv 1(\mathrm{mod}2).$$

Searching for the division properties or monomial trails is a time and memory-consuming task in practice, with complexity usually being an exponential function of the block size. To accelerate this process, Xiang et al. introduced Mixed Integer Linear Programming (MILP) models for conventional division properties at ASIACRYPT 2016 [18], which have since become the dominant tool in this area.

A symmetric cipher can be decomposed into a sequence of basic operations, such as XOR, AND, and COPY. Therefore, it is sufficient to provide the propagation rules of the monomial prediction for these basic operations. To model the propagation of the monomial prediction for a Sbox operation, a common approach is to list all the possible (input, output) tuples according to the definition of the monomial prediction, which can be transformed into a set of linear inequalities [19–21] suitable for a MILP model. The concrete propagation rules and models of the monomial prediction are provided in Appendix A.

3 Integral-Resistance Property

In this section, we briefly review the integral-resistance property, proposed in [5]. For more details on the integral-resistance property, we refer the reader to [5].

3.1 Basic Idea of the Proof in [5]

Consider a block cipher $E_k(x) : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$, where x is an n-bit plaintext and k is an m-bit key. According to Equation 1, for a given linear mask $\beta \in \mathbb{F}_2^n$, the ANF of $\langle \beta, E_k(x) \rangle$ can be represented as $\sum_{u \in \mathbb{F}_2^n, v \in \mathbb{F}_2^m} \lambda_{u,v}^{\beta} x^u k^v$. Here, $\lambda_{u,v}^{\beta} \in \mathbb{F}_2$ is the coefficient of the monomial $x^u k^v$ in $\langle \beta, E_k(x) \rangle$. According to Lemma 1,

$$\lambda_{u,v}^{\beta} = |x^u k^v \bowtie \langle \beta, E_k(x) \rangle| \bmod 2.$$

As a first step, Hebborn et al. introduced the *integral-resistance matrix*, denoted by $\mathcal{I}(E)_{v_0,\cdots,v_{n^2-1}}$, which records the coefficients of all plaintext monomials of degree $n-1$ with selected key monomials $(k^{v_0}, k^{v_1}, \ldots, k^{v_{n^2-1}})$ across the n output bits of $E_k(x)$. Equivalently, these coefficients correspond to the ANFs of $\langle \beta, E_k(x) \rangle$ for all unit vectors β. The integral-resistance matrix $\mathcal{I}(E)_{v_0,\cdots,v_{n^2-1}}$ is defined as follows,

$$\begin{pmatrix} \lambda_{\bar{e}_0,v_0}^{e_0} & \lambda_{\bar{e}_0,v_0}^{e_1} & \cdots & \lambda_{\bar{e}_0,v_0}^{e_{n-1}} & \lambda_{\bar{e}_1,v_0}^{e_0} & \lambda_{\bar{e}_1,v_0}^{e_1} & \cdots & \lambda_{\bar{e}_i,v_0}^{e_\ell} & \cdots & \lambda_{\bar{e}_{n-1},v_0}^{e_{n-1}} \\ \lambda_{\bar{e}_0,v_1}^{e_0} & \lambda_{\bar{e}_0,v_1}^{e_1} & & \lambda_{\bar{e}_0,v_1}^{e_{n-1}} & \lambda_{\bar{e}_1,v_1}^{e_0} & \lambda_{\bar{e}_1,v_1}^{e_1} & & \lambda_{\bar{e}_i,v_1}^{e_\ell} & & \lambda_{\bar{e}_{n-1},v_1}^{e_{n-1}} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \lambda_{\bar{e}_0,v_{n^2-1}}^{e_0} & \lambda_{\bar{e}_0,v_{n^2-1}}^{e_1} & & \lambda_{\bar{e}_0,v_{n^2-1}}^{e_{n-1}} & \lambda_{\bar{e}_1,v_{n^2-1}}^{e_0} & \lambda_{\bar{e}_1,v_{n^2-1}}^{e_1} & & \lambda_{\bar{e}_i,v_{n^2-1}}^{e_\ell} & & \lambda_{\bar{e}_{n-1},v_{n^2-1}}^{e_{n-1}} \end{pmatrix}.$$

Each row of this matrix corresponds to a distinct key monomial k^v. Based on the integral-resistance matrix and the assumption of independent round keys, Hebborn et al. established the following integral-resistance property.

Theorem 1 (Integral-Resistance Property [5])**.** *Let $E_k(x)$ be a block cipher. If there exists some choice of $v_0, \ldots, v_{n^2-1}$ such that $\mathcal{I}(E)_{v_0,\ldots,v_{n^2-1}}$ has full rank, and k_0 is an independent whitening key, then $E_k(x \oplus k_0)$ satisfies the integral-resistance property. That is, for every non-empty subset $\mathbb{M} \subset \mathbb{F}_2^n$ and every nonzero output mask $\beta \in \mathbb{F}_2^n$, the sum*

$$\sum_{x \in \mathbb{M}} \langle \beta, E_k(x \oplus k_0) \rangle$$

is always key-dependent.

The proof of integral-resistance property by means of Theorem 1 requires the block cipher to have whitening keys and ensures that, for the remainder of the cipher, the ANFs of all ciphertext bits and their linear combinations contain all plaintext monomials of degree $n-1$. In the proof, the round keys are treated as independent variables, with a length of $r \times n$ where r is the number of rounds in the cipher. Given a key monomial k^v, the corresponding row in the integral-resistance matrix can be computed.

3.2 Strategies in [5] for Choosing Key Monomials

The integral-resistance matrix cannot be of full rank if the key monomials are not chosen carefully. To illustrate Hebborn et al.'s strategy of choosing key monomials, we use the block cipher Present (whose specification is illustrated in Appendix B) as an example. In the r-round Present, K_{r-1} and P_{r-1} in the last round will not affect our proof, so we will omit them. After removing the whitening key, and swapping the

positions between K_0 and P_0 [2], the r-round Present is then written as

$$\mathsf{E} = \underbrace{\mathsf{S}_{r-1} \circ \mathsf{K}_{r-2}}_{\mathsf{E}_2} \circ \underbrace{\mathsf{P}_{r-2} \circ \mathsf{S}_{r-2} \circ \mathsf{K}_{r-3} \circ \mathsf{P}_{r-3} \circ \cdots \circ \mathsf{S}_1 \circ \mathsf{P}_0}_{\mathsf{E}_1} \circ \underbrace{\mathsf{K}_0 \circ \mathsf{S}_0}_{\mathsf{E}_0}. \tag{2}$$

Round keys are viewed as independent variables, and we denote by k_i the key variables of round i. A key monomial at round i is written as ${k_i}^{w_i}$. If w_i are fixed for all rounds, the overall key monomial k^v is uniquely determined as

$$k^v = {k_0}^{w_0} {k_1}^{w_1} \cdots {k_{r-2}}^{w_{r-2}}.$$

$\mathsf{E}_0 = \mathsf{K}_0 \circ \mathsf{S}_0$ shown in Figure 1 can be represented by 16 parallel small functions composed of the Sbox followed by a 4-bit key addition. The i-th small function is denoted by $\mathsf{E}_{0,i} = \mathrm{Sbox}(\xi_i) \oplus \kappa_i$, where ξ_i and κ_i are the input and key corresponding to the i-th small function $\mathsf{E}_{0,i}$. Naturally, $\xi_i = x_0[4i : 4i+3]$ and $\kappa_i = k_0[4i : 4i+3]$. Using the property of $\mathsf{E}_{0,i}$, the integral-resistance matrix can be formed as a block diagonal matrix. Since this property is used many times in the following part of this paper, we summarize it into the following lemma.

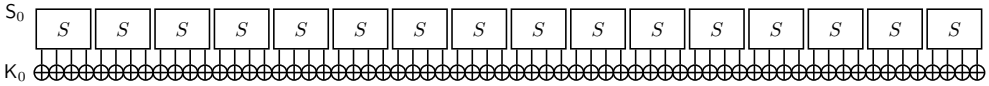


Fig. 1: An illustration of E_0 in Present

Lemma 2. *Let $\boldsymbol{f} : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a vector Boolean function defined as $\boldsymbol{f}(\xi, \kappa) = \mathrm{Sbox}(\xi) \oplus \kappa$, where Sbox is an invertible function and κ is a secret key. Let $v \neq 0^n$. Then ${\xi}^{1^n} \kappa^v$ does not appear in any output monomials.*

Proof. Let $y = \mathrm{Sbox}(\xi)$. Since Sbox is invertible, only ${y}^{1^n} = \prod_{i=0}^{n-1} y[i]$ can contain the monomial ${\xi}^{1^n}$. The function $\boldsymbol{f}$ is defined as $\boldsymbol{f}(\xi, \kappa) = y \oplus \kappa$. Due to the XOR rule of monomial trails(Appendix A), the monomial of $\boldsymbol{f}$ can only contain either the key or the state variable, never both. For each coordinate, $f_i(x, \kappa) = y[i] \oplus \kappa[i]$, whose ANF contains no product term involving both $y[i]$ and $\kappa[i]$. Hence, no monomial of the form ${y}^{1^n} \kappa^v$ with $v \neq 0^n$ can appear in any coordinate of $\boldsymbol{f}$ or in any product of coordinates. Therefore, no monomial of the form ${x}^{1^n} \kappa^v$ appears in the monomial of $\boldsymbol{f}$. □

According to Lemma 2, if the key monomial associated with $\mathsf{E}_{0,i}$ is nonzero, then all $\lambda^{e_\ell}_{\bar{e}_{i'}, v_j}$ (where $0 \leq i' \leq 63$ and $i' \notin \{4i, 4i+1, 4i+2, 4i+3\}$) are zero, since ${x_0}^{\bar{e}_{i'}}$ contains the full product monomial ${\xi_i}^{1^4}$. The corresponding constraint of the key monomial is as follows,

$$w_0[4i] + w_0[4i+1] + w_0[4i+2] + w_0[4i+3] \geq 1. \tag{3}$$

[2] This swap helps clarify the strategy and does not affect the proof, since P_0 is merely a bit permutation. If a key monomial is obtained after swapping P_0 and K_0, the corresponding valid key monomial in the original setting can be easily derived by applying the bit permutation rule.

A similar approach can be used for $\mathsf{E}_{2,\ell}$, where choosing a monomial of a key used only in one Sbox of S_{r-1} allows certain entries in the $\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}}$ to be determined as 0. The constraint of the key monomial is as follows,

$$w_{r-2}[4\ell] + w_{r-2}[4\ell+1] + w_{r-2}[4\ell+2] + w_{r-2}[4\ell+3] \geq 1. \tag{4}$$

Finally, for each i, ℓ, by selecting 16 key monomials k^v that satisfy the constraints Equation 3 and Equation 4, and by permuting the columns, $\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}}$ will be a block diagonal matrix, as follows,

$$\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}} = \begin{pmatrix} \boldsymbol{A_{0,0}} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{A_{0,1}} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{A_{15,15}} \end{pmatrix}.$$

Each block $A_{i,\ell}$ is determined by an $\mathsf{E}_{0,i}$ and $\mathsf{E}_{2,\ell}$ as follows,

$$\boldsymbol{A_{i,\ell}} = \begin{pmatrix} \lambda^{(4\ell)}_{\bar{e}_{4i},v_j} & \cdots & \lambda^{(4\ell+3)}_{\bar{e}_{4i},v_j} & \cdots & \lambda^{(4\ell)}_{\bar{e}_{4i+3},v_j} & \cdots & \lambda^{(4\ell+3)}_{\bar{e}_{4i+3},v_j} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \lambda^{(4\ell)}_{\bar{e}_{4i},v_{j+15}} & \cdots & \lambda^{(4\ell+3)}_{\bar{e}_{4i},v_{j+15}} & \cdots & \lambda^{(4\ell)}_{\bar{e}_{4i+3},v_{j+15}} & \cdots & \cdots \lambda^{(4\ell+3)}_{\bar{e}_{4i+3},v_{j+15}} \end{pmatrix}.$$

It is obvious that if all blocks $\boldsymbol{A}_{i,\ell}$ are of full rank, the integral-resistance matrix is also of full rank. In [5], the construction of a full-rank matrix $A_{i,\ell}$ proceeds through the following steps:

1. Compute the monomial trails for the four input monomials of degree $n-1$ of $\mathsf{E}_{0,i}$, and identify the corresponding key and output monomials;
2. Compute the monomial trails for four output monomials of degree 1 from $\mathsf{E}_{2,\ell}$, and identify their corresponding input and key monomials;
3. Determine the middle key monomials used within E_1 by solving a MILP model;
4. Compute the value of entries in $\boldsymbol{A}_{i,\ell}$ by counting monomial trails.

The above process proposed by Hebborn et al. [5] is depicted in Figure 2.

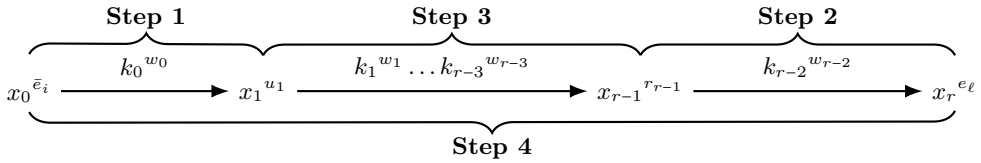


Fig. 2: The process of Hebborn et al.

Next, we take $\boldsymbol{A}_{0,0}$, determined by $\mathsf{E}_{0,0}$ and $\mathsf{E}_{2,0}$, as an example to illustrate the process of Hebborn et al. In Steps 1 and 2, the monomial trails of $\mathsf{E}_{0,0}$ and $\mathsf{E}_{2,0}$ are manually determined, as shown in Table 1.

key monomial \ input monomial	${x_0}^{\bar{e}_0}$	${x_0}^{\bar{e}_1}$	${x_0}^{\bar{e}_2}$	${x_0}^{\bar{e}_3}$
$k_0[0]k_0[2]$	${x_1}^{\bar{e}_3}$	$\star$	$\star$	$\star$
$k_0[2]$	$\varnothing$	${x_1}^{\bar{e}_3}$	$\star$	$\varnothing$
$k_0[1]$	$\varnothing$	$\varnothing$	${x_1}^{\bar{e}_3}$	$\varnothing$
$k_0[0]k_0[1]$	$\varnothing$	$\star$	$\star$	${x_1}^{\bar{e}_3}$

(a) Monomial trails of $\mathsf{E}_{0,0}$ (input side)

key monomial \ output monomial	${x_r}^{e_0}$	${x_r}^{e_1}$	${x_r}^{e_2}$	${x_r}^{e_3}$
$k_{r-2}[2]$	${x_{r-1}}^{e_1}$	$\varnothing$	$\varnothing$	$\varnothing$
$k_{r-2}[3]$	$\star$	${x_{r-1}}^{e_1}$	$\varnothing$	$\varnothing$
$k_{r-2}[0]k_{r-2}[3]$	$\star$	$\star$	${x_{r-1}}^{e_1}$	$\star$
$k_{r-2}[0]k_{r-2}[2]$	$\star$	$\star$	$\varnothing$	${x_{r-1}}^{e_1}$

(b) Monomial trails of $\mathsf{E}_{2,0}$ (output side)

Table 1: Monomial trails of $\mathsf{E}_{0,0}$ and $\mathsf{E}_{2,0}$. For the four input monomials of degree $n-1$, the same ${x_1}^{u_1}$, namely ${x_1}^{\bar{e}_3}$, is associated with four different key monomials. Likewise, for the four output monomials of degree 1, the same ${x_{r-1}}^{u_{r-1}}$, namely ${x_{r-1}}^{e_1}$, is associated with four different key monomials. The symbol $\varnothing$ indicates that no monomial trail exists, while $\star$ indicates the existence of a monomial trail that is irrelevant to the rank computation of $A_{0,0}$.

Since four input monomials of degree $n-1$ share the same ${x_1}^{u_1}$ and four output monomials of degree 1 share the same ${x_{r-1}}^{u_{r-1}}$, in Step 3, it suffices to find one key monomial used within E_1 such that ${k_1}^{w_1} \dots {k_{r-3}}^{w_{r-3}} {x_1}^{\bar{e}_3} \to {x_{r-1}}^{e_1}$.

In Step 4, certain entries of $\boldsymbol{A}_{0,0}$ can be determined directly without enumerating monomial trails. Since the monomial trail is consistent with Figure 2, each entry of $\boldsymbol{A}_{0,0}$ depends on one entry of Table 1a and one entry of Table 1b. If either entry in Table 1a or Table 1b is $\varnothing$, the corresponding entry in $\boldsymbol{A}_{0,0}$ is 0, since no $x_1^{u_1}$ can contain the corresponding ${k_0}^{w_0}{x_0}^{u_0}$ or no product ${x_{r-1}}^{u_{r-1}} \cdot {k_{r-2}}^{w_{r-2}}$ can be contained by the corresponding ${x_r}^{u_r}$. If one is $\star$ and the other is not $\varnothing$, the entry of $\boldsymbol{A}_{0,0}$ is marked $\star$[3]. If one is ${x_1}^{\bar{e}_3}$ and another is ${x_{r-1}}^{e_1}$, we check the number of corresponding monomial trails. If odd, the corresponding entry in $\boldsymbol{A}_{0,0}$ is set to 1; otherwise, it is 0. In total, only 16 entries in $\boldsymbol{A}_{0,0}$ need to be checked. We use the following matrices to represent the monomial trails of $\mathsf{E}_{0,0}$ and $\mathsf{E}_{2,0}$, where 0 represents $\varnothing$ and 1 represents ${x_1}^{\bar{e}_3}$ and

[3]Although some entries are marked with $\star$, they have no impact on the rank computation. For any possible values of these entries, the integral resistance matrix is full-rank.

${x_{r-1}}^{e_1}$.

$$\boldsymbol{M}(\mathsf{E}_{0,0}) = \begin{pmatrix} 1 & \star & \star & \star \\ 0 & 1 & \star & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \star & \star & 1 \end{pmatrix} \quad \boldsymbol{M}(\mathsf{E}_{2,0}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \star & 1 & 0 & 0 \\ \star & \star & 1 & \star \\ \star & \star & 0 & 1 \end{pmatrix}$$

The following lemma guarantees that the block matrix $\boldsymbol{A}_{0,0}$ is of full rank.

Lemma 3. *If both $\boldsymbol{M}(\mathsf{E}_{0,0})$ and $\boldsymbol{M}(\mathsf{E}_{2,0})$ are full-rank, and all the 16 entries of $\boldsymbol{A}_{0,0}$ that need to be checked take the value 1, then $\boldsymbol{A}_{0,0}$ is full-rank.*

Proof. When all the 16 entries of $\boldsymbol{A}_{0,0}$ that need to be checked are equal to 1, each entry of $\boldsymbol{A}_{0,0}$ can be expressed as the product of one entry from $\boldsymbol{M}(\mathsf{E}_{0,0})$ and one entry from $\boldsymbol{M}(\mathsf{E}_{2,0})$. Consequently, $\boldsymbol{A}_{0,0}$ can be viewed as the Kronecker product

$$\boldsymbol{A}_{0,0} = \boldsymbol{M}(\mathsf{E}_{0,0}) \otimes \boldsymbol{M}(\mathsf{E}_{2,0}).$$

Since both $\boldsymbol{M}(\mathsf{E}_{0,0})$ and $\boldsymbol{M}(\mathsf{E}_{2,0})$ are full-rank, their Kronecker product $\boldsymbol{A}_{0,0}$ is also full-rank.

□

In the proof process, Step 3 and Step 4 are usually the most time-consuming step, as it would lead to a repeated search for a middle key monomial. In Section 4, we analyze the reason for the failure of Step 3 and propose a solution to handle it. With our solution, we do not need Step 4 anymore. The whole process will terminate once the first three steps are done.

4 Improved Strategies and Results for Present

Using previous techniques, some of the searched key monomials turn out to be invalid. In this section, we investigate the root cause of the invalidity of the key monomial, namely the cancellation of monomial trails due to interfering monomials. By suppressing these interferences, the validity of the selected key monomial used within E_1 is ensured, enabling the omission of Step 4 and yielding improved results on Present.

4.1 Mind the Interfering Monomials

For the sake of clarity, we also take the process of identifying the key monomial for $\boldsymbol{A}_{0,0}$ of Present as an example. Following [5], we choose the monomial trails from Table 1 for both the first round and last round. For conciseness, let σ_i and τ_ℓ be the selected key monomials in ${k_0}^{w_0}$ and ${k_{r-2}}^{w_{r-2}}$ for ${x_0}^{\bar{e}_i}$ and ${x_r}^{e_\ell}$, respectively. Specifically, we choose ${x_1}^{\bar{e}_3}$ as ${x_1}^{u_1}$ and the four key monomials in $\mathsf{E}_{0,0}$ are chosen as

$$\sigma_0 = k_0[0]k_0[2],\ \sigma_1 = k_0[2],\ \sigma_2 = k_0[1],\ \sigma_3 = k_0[0]k_0[1].$$

Likewise, we choose ${x_{r-1}}^{e_1}$ as ${x_{r-1}}^{u_{r-1}}$. The four key monomials for $\mathsf{E}_{2,0}$ are chosen as

$$\tau_0 = k_{r-2}[2],\ \tau_1 = k_{r-2}[3],\ \tau_2 = k_{r-2}[0]k_{r-2}[3],\ \tau_3 = k_{r-2}[0]k_{r-2}[2].$$

However, besides $x_0{}^{\bar{e}_i} \cdot \sigma_i \to x_1{}^{\bar{e}_3}$, the input term can also be contained in other monomials of the output of $\mathsf{E}_{0,0}$. For example, $x_0{}^{\bar{e}_0} \cdot \sigma_0 \to x_1{}^{\bar{e}_1}$. This causes trouble that even if we find a proper $k_1{}^{w_1} \dots k_{r-3}{}^{w_{r-3}}$ that makes $x_0{}^{\bar{e}_0} \cdot \sigma_0 \cdot k_1{}^{w_1} \dots k_{r-3}{}^{w_{r-3}} \cdot \tau_\ell \to x_{r-1}{}^{e_1}$, the term $x_0{}^{\bar{e}_0}$ in $x_{r-1}{}^{e_1}$ can still be cancelled by the trails via $x_1{}^{\bar{e}_1}$. Such a situation is illustrated by Figure 3. If a key monomial $k_1{}^{w_1} \dots k_{r-3}{}^{w_{r-3}}$ is identified, Hebborn et al. compute the corresponding entry in $\boldsymbol{A}_{0,0}$ as

$$|x_0{}^{\bar{e}_0} \cdot \sigma_0 \cdot k_1{}^{w_1} \dots k_{r-3}{}^{w_{r-3}} \cdot \tau_0 \bowtie x_r{}^{e_1}| \bmod 2.$$

In this case, it is evident that the number of corresponding monomial trails is even, and $\boldsymbol{A}_{0,0}$ cannot be full rank.

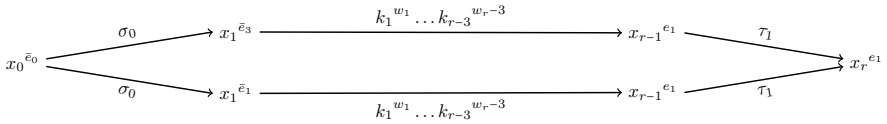


Fig. 3: Even if we find proper $w_1, w_2, \dots, w_{r-3}$, making $x_1{}^{\bar{e}_3} \cdot k_1{}^{w_1} \dots k_{r-3}{}^{w_{r-3}} \to x_{r-1}{}^{e_1}$ (at the same time $x_1{}^{\bar{e}_1} \cdot k_1{}^{w_1} \dots k_{r-3}{}^{w_{r-3}} \to x_{r-1}{}^{e_1}$), the two terms, $x_1{}^{\bar{e}_3}$ and $x_1{}^{\bar{e}_1}$, can still cancel the term $x_0{}^{\bar{e}_0}$ in each other.

When selecting a key monomial for the first round, it is very likely that two [4] monomials such as $x_1{}^{u_1}$ and $x_1{}^{u_1'}$, will appear along with the selected key monomial. We choose one, for example $x_1{}^{u_1}$, as the candidate to construct the final key monomial; the other one, $x_1{}^{u_1'}$, usually causes interference, meaning the trails attributed to it tend to cancel those from $x_1{}^{u_1}$. Similarly, for the last round, $x_{r-1}{}^{u_{r-1}}$ and $x_{r-1}{}^{u_{r-1}'}$ may appear together, and the latter typically acts as the interfering monomial of $x_{r-1}{}^{u_{r-1}}$.

To identify all interfering monomials, we model both the first and the last rounds of Present using a MILP formulation and solve the resulting models with Gurobi solver [22]. Specifically, for all input monomials of degree 63, we enumerate all feasible monomial trails of the form

$$x_0{}^{\bar{e}_i} \xrightarrow{k_0{}^{w_0}} x_1{}^{u_1}.$$

From the obtained monomial trails, we can determine which monomials of $x_1{}^{u_1}$ contain the term $x_0{}^{\bar{e}_i}$ multiplied by the chosen key monomial σ_i. In this way, all interfering monomials in the first round can be identified.

Similarly, for all output monomials of degree 1, we enumerate all feasible monomial trails of the form

$$x_{r-1}{}^{u_{r-1}} \xrightarrow{k_{r-2}{}^{w_{r-2}}} x_r{}^{e_\ell}.$$

Based on these trails, we can determine which monomials of $x_{r-1}{}^{u_{r-1}}$, when multiplied by the chosen key monomial τ_ℓ, can be contained by $x_r{}^{e_\ell}$. Consequently, all interfering monomials in the last round are obtained.

By solving all MILP models, we find that only $x_1{}^{\bar{e}_1}$ is the interfering monomial of $x_1{}^{\bar{e}_3}$ and only $x_{r-1}{}^{e_2}$ is the interfering monomial of $x_{r-1}{}^{e_1}$. Thus, there are three

[4] Actually, there might be more. But in our applications, usually only two such interfering monomials are found.

interfering monomial trails that will cause the identified complete key monomial to be invalid, which is illustrated as follows,

$$x_1{}^{\bar{e}_3} \cdot {k_1}^{w_1} \dots {k_{r-3}}^{w_{r-3}} \xrightarrow{?} {x_{r-1}}^{e_2},$$

$$x_1{}^{\bar{e}_1} \cdot {k_1}^{w_1} \dots {k_{r-3}}^{w_{r-3}} \xrightarrow{?} {x_{r-1}}^{e_1},$$
$$x_1{}^{\bar{e}_1} \cdot {k_1}^{w_1} \dots {k_{r-3}}^{w_{r-3}} \xrightarrow{?} {x_{r-1}}^{e_2}.$$

The invalidation of identified key monomials due to interfering monomials occurs frequently. When determining the key monomial used within E_1 for an 11-round Present, one such monomial we identified resulted in 239 monomial trails both from $x_1{}^{\bar{e}_3}$ to ${x_{11}}^{e_1}$ and from $x_1{}^{\bar{e}_1}$ to ${x_{11}}^{e_1}$. In addition, there were 149 trails both from $x_1{}^{\bar{e}_3}$ to ${x_{11}}^{e_2}$ and from $x_1{}^{\bar{e}_1}$ to ${x_{11}}^{e_2}$. Since these monomial trails cancel each other, even if the key monomial used within E_1 is valid, the corresponding coefficient in $A_{0,0}$ can still be zero. The presence of interfering monomials hinders the identification of valid key monomials. Therefore, it is essential to take interfering monomials into account when selecting key monomials. Next, we present two methods to address this problem. These methods ensure that, once a key monomial used within E_1 is found, the corresponding small block matrix $\boldsymbol{A}_{i,\ell}$ will be of full rank.

Adding constraints to the key monomial used within E_1. A natural idea is to add constraints to the key monomials used within E_1 to prevent interfering monomials from canceling the monomial trails. If the key monomial ${k_1}^{w_1} \dots {k_{r-3}}^{w_{r-3}}$ fulfills Equation 5, no monomial trails can go through $x_1{}^{\bar{e}_1}$.

$$\sum_{i=0}^{3} w_1[16i+12] + \sum_{i=0}^{3}\sum_{j=0}^{3} w_2[16i+4j+1] \geq 1. \tag{5}$$

Let x' be the state of the output of P_0. Note that P_0 is a bit permutation, so the only term $(x')^u$ that contains $x_1{}^{\bar{e}_1}$ is $(x')^{\bar{e}_{16}}$, and the only term that contains $x_1{}^{\bar{e}_3}$ is $(x')^{\bar{e}_{48}}$. Equation 5 consists of two basic conditions, which are discussed one by one as follows,

1. $\sum_{i=0}^{3} w_1[16i+12] \geq 1$. We consider monomial trails starting from $x_1{}^{\bar{e}_1}$ through the composition $\mathsf{K}_1 \circ \mathsf{P}_1 \circ \mathsf{S}_1 \circ \mathsf{P}_0$. The structure of $\mathsf{K}_1 \circ \mathsf{P}_1 \circ \mathsf{S}_1$ is similar to E_0; it can also be decomposed into 16 parallel small functions. Towards to $(x')^{\bar{e}_{48}}$, which is the output monomial of $x_1{}^{\bar{e}_3}$ through P_0, the input monomial to the fifth small function is $(x')^{1^4}$. Towards to $(x')^{\bar{e}_{16}}$, which is the output monomial of $x_1{}^{\bar{e}_1}$ through P_0, the input monomial to the same function is $(x')^{e_0}$ (here e_0 denotes a four-bit unit vector). According to Lemma 2, if we choose a key monomial of ${k_1}^{w_1}$ associated with the fifth small function to be nonzero, i.e.,

$$w_1[12] + w_1[28] + w_1[44] + w_1[60] \geq 1,$$

then $(x')^{1^4}$ is blocked from propagation while $(x')^{e_0}$ is allowed. Therefore, all terms related to $x_1{}^{\bar{e}_1}$ are prevented from propagating, whereas those related to $x_1{}^{\bar{e}_3}$ are allowed to propagate.

2. $\sum_{i=0}^{3}\sum_{j=0}^{3} w_2[16i+4j+1] \geq 1$ and $wt(w_1) = 0$. Here, we further consider the monomial trails starting from ${x_1}^{\bar{e}_1}$ through $\mathsf{K}_2 \circ \mathsf{P}_2 \circ \mathsf{S}_2 \circ \mathsf{K}_1 \circ \mathsf{P}_1 \circ \mathsf{S}_1 \circ \mathsf{P}_0$. When $wt(w_1) = 0$, K_1 can be omitted without affecting the result of monomial propagation. Thus, $\mathsf{P}_2 \circ \mathsf{S}_2 \circ \mathsf{K}_1 \circ \mathsf{P}_1 \circ \mathsf{S}_1$ can be regarded as a Supersbox layer. A Supersbox layer followed by a key addition exhibits a similar property to an Sbox layer followed by a key addition, which can also be represented by 16 parallel small function. Towards to $(x')^{\bar{e}_{48}}$, which is the output monomial of ${x_1}^{\bar{e}_3}$ through P_0, the input monomial to the fifth small function is $(x')^{1^{16}}$. Towards to $(x')^{\bar{e}_{16}}$, which is the output monomial of ${x_1}^{\bar{e}_1}$ through P_0, the input monomial to the same function is $(x')^{e_0}$(here e_0 denotes a 16-bit unit vector). According to Lemma 2, if we choose a key monomial of ${k_1}^{w_1}$ associated with the second small function to be nonzero, i.e.,

$$\sum_{i=0}^{3} w_2[16i+3] + \sum_{i=0}^{3} w_2[16i+7] + \sum_{i=0}^{3} w_2[16i+11] + \sum_{i=0}^{3} w_2[16i+15] \geq 1,$$

then $(x')^{1^{16}}$ is blocked from propagation while $(x')^{e_0}$ is allowed. Therefore, all terms related to ${x_1}^{\bar{e}_1}$ will be prevented from propagation, while those related to ${x_1}^{\bar{e}_3}$ are allowed.

When either condition is met, ${x_1}^{\bar{e}_3}$ multiplied by the found key monomials used within E_1 cannot be contained by any monomial. To enlarge the search space for the key monomials used within E_1, we incorporate Equation 5 into the MILP model, which simultaneously captures both conditions. Similarly, if the key monomial ${k_1}^{w_1} \ldots {k_{r-3}}^{w_{r-3}}$ fulfills Equation 6, no monomial trails can go through ${x_1}^{e_2}$.

$$\sum_{i=4}^{7} w_{r-3}[i] + \sum_{i=16}^{31} w_{r-4}[i] \geq 1 \tag{6}$$

To clarify the probability that random inner-round keys satisfy these constraints, we evaluate the probability using MILP models solved by Gurobi. For the two-round forward extension from ${x_1}^{\bar{e}_3}$, we enumerate all possible monomial trails and test their compatibility with Equation 5. We find that among 195 possible choices of random inner-round keys, 112 satisfy Equation 5. This shows that the equation holds with a significant but clearly bounded probability, rather than almost surely. Similarly, we analyze the two-round backward extension from ${x_{r-1}}^{e_1}$. In this case, we identify 154 possible monomial trails, among which 86 satisfy Equation 6. When searching a key monomial used within E_1, if Equation 5 and Equation 6 are satisfied, the interfering monomials ${x_1}^{\bar{e}_1}$ and ${x_{r-2}}^{e_2}$ will never cancel the monomial trails.

Use the Callback function. The Callback function in the Gurobi solver [22] can add a lazy constraint to the MILP model, allowing users to cut off a feasible solution during the search. For more details on the Callback functions of Gurobi, readers are referred to the Gurobi manual [23]. When the round number of the target cipher is small, the proof approaches the tight bound of the integral resistance. If all constraints from the first and last rounds are added simultaneously, the desirable monomial trails may not exist. To address this problem, we enhance our search strategy as follows.

If desirable monomial trails cannot be found, we relax the search space by adding only partial constraints (i.e., only one of Equation 5 or Equation 6) instead of all constraints, thereby expanding the search space. Naturally, this expansion enlarges the search space and increases the probability of cancellation due to interfering monomials. Nevertheless, the callback function of Gurobi allows us to prune invalid key monomials without restarting the entire process, which mitigates this problem. Consequently, the middle key monomials obtained via this method are guaranteed to be valid, as any invalid ones are pruned during the callback procedure. Applying this approach, we obtain six key monomials in our proof for the 11-round Present.

Considering the effect of interfering monomials reduces the overall cost of constructing a full-rank integral-resistance matrix for a round-reduced block cipher. When searching for a middle key monomial using a heuristic algorithm, for Present, it usually takes from several CPU minutes to several CPU hours to find a candidate middle key monomial. However, this candidate may later be proved invalid in Step 4. Although Step 4 itself only costs several minutes for Present, it may force us to repeat the expensive search in Step 3. By taking interfering monomials into account, we can guarantee that the obtained middle key monomial is valid. On the one hand, this makes Step 4 unnecessary; on the other hand, it avoids repeated executions of Step 3 for searching middle key monomials. For block ciphers with more complex Sboxes and linear layers, considering interfering monomials becomes even more effective, since both Step 3 and Step 4 are significantly more time-consuming. In our experiments on Twine and Lblock, it may take several CPU days to search out a middle key monomial.

4.2 Improved Security Guarantee for Present

In [5], the authors provided the integral-resistance property for 13-round Present while the longest integral distinguisher of Present reaches only 9 rounds. Here, we give the security guarantee for a smaller round-reduced Present.

Security guarantee of 12-round Present. For four monomials of degree $n-1$ of x_0 corresponding to the same $\mathsf{E}_{0,i}$ for $0 \le i \le 15$, we select an identical ${x_1}^{u_1}$, which is ${x_1}^{\bar{e}_{4i+3}}$. The values of corresponding key monomials ${k_0}^{w_0}$ are as follows:

$$\sigma_{4i} = k_0[4i]k_0[4i+2],\ \sigma_{4i+1} = k_0[4i+2],\ \sigma_{4i+2} = k_0[4i+1],\ \sigma_{4i+3} = k_0[4i]k_0[4i+1].$$

Similarly, for four monomials of degree 1 of x_{12} corresponding to the same $\mathsf{E}_{2,\ell}$ for $0 \le \ell \le 15$, we choose an identical ${x_{11}}^{u_{11}}$, i.e., ${x_{11}}^{e_{4\ell+1}}$. The values of corresponding key monomials ${k_{10}}^{w_{10}}$ are as follows:

$$\tau_{4\ell} = k_{10}[4\ell+2], \tau_{4\ell+1} = k_{10}[4\ell+3], \tau_{4\ell+2} = k_{10}[4\ell]k_{10}[4\ell+3], \tau_{4\ell+3} = k_{10}[4\ell]k_{10}[4\ell+2].$$

In the case of the 12-round Present, we search for all middle key monomials that correspond to a pair of ${x_1}^{u_1}$ and ${x_{11}}^{u_{11}}$ by solving a MILP model of the 10-round Present. To prevent the cancellation of the ${x_0}^{u_0}$, we incorporate all our constraints into the MILP model. All middle key monomials can be found within one CPU hour using a server equipped with 36 Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz and 534GB

of memory. Some of the programs were run in parallel, as long as there is enough memory. Therefore, we can conclude that the 12-round Present is integral-resistant.

Security guarantee of 11-round Present. The procedure for determining the first and last round key monomials is identical to that used for 12-round Present. However, seven middle-round key monomials cannot be determined directly. By adding partial constraints to the MILP model, six of these seven monomials can be identified. No monomial trail exists from ${x_1}^{\bar{e}_3}$ to ${x_{10}}^{e_1}$, which implies that all block matrices $\boldsymbol{A}_{i,\ell}$, except $\boldsymbol{A}_{0,0}$, are of full rank. The resulting integral-resistance matrix has the following form,

$$\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}} = \begin{pmatrix} \boldsymbol{A_{0,0}} & \star & \cdots & \star \\ \boldsymbol{O} & \boldsymbol{A_{0,1}} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{A_{15,15}} \end{pmatrix}. \tag{7}$$

It is sufficient that $\boldsymbol{A}_{0,0}$ is full rank to ensure that $\mathcal{I}(\mathsf{E})_{v_0,\ldots,v_{n^2-1}}$ is full rank, regardless of the structure of the other block matrices denoted by $\star$. For $\boldsymbol{A}_{0,0}$, the entries correspond to four input monomials of degree $n-1$ and four output monomials of degree 1. For four output monomials of degree 1, we will also backtrack one round by determining ${x_{10}}^{u_{10}}$ and the key monomial ${k_{10}}^{w_{10}}$. The monomial trail of the last round is similar to Table 1b. For some input monomials of degree $n-1$, we will forward track two rounds by determining ${x_2}^{u_2}$ and the key monomial ${k_0}^{w_0}{k_1}^{w_1}$. The details are as follows,

1. For ${x_0}^{\bar{e}_0}$, we choose the monomial $\frac{{x_1}^n}{x_1[0]x_1[2]}$ which is of degree 62 as ${x_1}^{u_1}$, and the corresponding key monomial is $k_0[1]$;
2. For ${x_0}^{\bar{e}_1}$, we choose the monomial $\frac{{x_2}^n}{x_2[24]x_2[28]x_2[60]}$ which is of degree 61 as ${x_2}^{u_2}$, and the corresponding key monomial is $k_1[56]$;
3. For ${x_0}^{\bar{e}_2}$, we choose the monomial $\frac{{x_2}^n}{x_2[28]x_2[52]x_2[60]}$ which is of degree 61 as ${x_2}^{u_2}$, and the corresponding key monomial is $k_1[20]$;
4. For ${x_0}^{\bar{e}_3}$, we choose the monomial $\frac{{x_1}^n}{x_1[0]x_1[3]}$ which is of degree 62 as ${x_1}^{u_1}$, and the corresponding key monomial is $k_0[1]$.

Some constraints need to be considered in searching the middle key monomials. The monomial $\frac{{x_2}^n}{x_2[24]x_2[28]x_2[60]}$ has an interfering monomial i.e., $\frac{{x_2}^n}{x_2[20]x_2[24]x_2[52]}$. We give the following constraint to eliminate the effect of this interfering monomial.

$$w_2[7] + w_2[23] + w_2[39] + w_2[55] + w_2[15] + w_2[31] + w_2[47] + w_2[63] \geq 1.$$

We note that the choice of w_0 corresponding to $x_0^{\bar{e}_0}$ and $x_0^{\bar{e}_3}$ are identical. The following constraint can eliminate the effect of the monomial $\frac{{x_1}^n}{x_1[0]x_1[3]}$.

$$\sum_{i=0}^{3} w_1[16i+8] + \sum_{i=0}^{3}\sum_{j=0}^{3} w_2[16i+4j+2] \geq 1.$$

And the effect of the monomial $\frac{{x_1}^n}{x_1[0]x_1[2]}$ is eliminated by the following constraint,

$$\sum_{i=0}^{3} w_1[16i+12] + \sum_{i=0}^{3}\sum_{j=0}^{3} w_2[16i+4j+3] \geq 1.$$

According to the strategy of Lemma 3, since we ensure that $\boldsymbol{M}(\mathsf{E}_{0,0})$ and $\boldsymbol{M}(\mathsf{E}_{2,0})$ are of full rank and search out all middle key monomials, the formed $\boldsymbol{A}_{0,0}$ is of full rank. We also generate 4096 key monomials and compute the rank of the resulting integral-resistance matrix in our experiments, thereby proving that the 11-round Present cipher admits no integral distinguisher.

Weak security guarantee of 10-round Present. For the proof of 10-round Present, we need to search middle key monomials within 8-round Present. However, we observe that 175 out of the 256 middle key monomials do not exist. For example, ${x_0}^{\bar{e}_1}$ cannot be contained in ${x_{10}}^{e_0}$ within 8-round Present. Consequently, our current method is insufficient to guarantee the resistance of the 10-round Present against integral attacks. Concerning whether a single output bit is balanced, a weaker result can be obtained for the 10-round Present. By regrouping the columns associated to the same output coordinate, the integral-resistance matrix can be transformed into the following form,

$$\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}} = \begin{pmatrix} \lambda^{e_0}_{\bar{e}_0,v_0} & \lambda^{e_0}_{\bar{e}_1,v_0} & \cdots & \lambda^{e_0}_{\bar{e}_{n-1},v_0} & \lambda^{e_1}_{\bar{e}_0,v_0} & \lambda^{e_1}_{\bar{e}_1,v_0} & \cdots & \lambda^{e_\ell}_{\bar{e}_i,v_0} & \cdots & \lambda^{e_{n-1}}_{\bar{e}_{n-1},v_0} \\ \lambda^{e_0}_{\bar{e}_0,v_1} & \lambda^{e_0}_{\bar{e}_1,v_1} & & \lambda^{e_0}_{\bar{e}_{n-1},v_1} & \lambda^{e_1}_{\bar{e}_0,v_1} & \lambda^{e_1}_{\bar{e}_1,v_1} & & \lambda^{e_\ell}_{\bar{e}_i,v_1} & & \lambda^{e_{n-1}}_{\bar{e}_{n-1},v_1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ \lambda^{e_0}_{\bar{e}_0,v_s} & \lambda^{e_0}_{\bar{e}_1,v_s} & & \lambda^{e_0}_{\bar{e}_{n-1},v_s} & \lambda^{e_1}_{\bar{e}_0,v_s} & \lambda^{e_1}_{\bar{e}_1,v_s} & & \lambda^{e_\ell}_{\bar{e}_i,v_s} & & \lambda^{e_{n-1}}_{\bar{e}_{n-1},v_s} \end{pmatrix}.$$

We represent $\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}}$ as the following block matrix.

$$\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}} = \begin{pmatrix} \boldsymbol{B_0} & \boldsymbol{B_1} & \cdots & \boldsymbol{B_{n-2}} & \boldsymbol{B_{n-1}} \end{pmatrix}$$

In cases where we are unable to find sufficient key monomials $k_1^{w_1}\cdots k_7^{w_7}$, we can resort to forcing $\boldsymbol{B_i}$ in $\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}}$ to be full rank. If $\boldsymbol{B_i}$ is of full rank, we can conclude the i-th bit of the ciphertext cannot give a constant sum, based on the integral-resistance property.

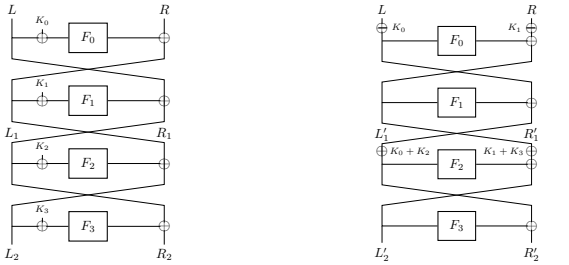
By identifying the key monomials, we can conclude that all matrices $\boldsymbol{B_i}$, except $\boldsymbol{B_0}$, are of full rank. As for $\boldsymbol{B_0}$, there are four key monomials that we cannot obtain. These findings suggest that, excluding the first bit, all other output bits cannot give a constant sum, namely, the sum of any of these 63 output bits (excluding the 0-th bit) for any subset of plaintexts is dependent on the key.

5 How to Provide Security Guarantee for Twine

In this section, we present the integral security guarantees for Twine, a lightweight block cipher utilizing a GFN structure, whose specification is shown in Appendix C.

5.1 Integral-resistance Property for Twine

To prove the integral-resistance property for a cipher by means of Theorem 1, we first need to make sure that the cipher has a whitening key and that the key addition is added to the full state. In the Feistel structure, there may be no whitening key, and the size of the subkey used in one round is assumed to be half of the block size (i.e., $k_i \in \{0,1\}^{n/2}$). But for a Feistel structure block cipher where the F-function involves a key addition and a random function, we can transform the structure to add the round key to the entire state, which is called an equivalent key transformation, as illustrated in Figure 4.



(a) 4 rounds before transformation

(b) 2 rounds after transformation

Fig. 4: We compare the original 4-round Feistel block cipher with its 2-round equivalent version after the key transformation. L and R are the inputs, and L_i, R_i (where $i = 1$ or 2), L'_i, R'_i correspond to the outputs of the original round (or the equivalent round).

Note that the output of the equivalent round differs from the original two rounds due to the key addition occurring before the COPY operation of the left branch, as illustrated below:

$$L'_1 = L_1 \oplus K_1, \qquad R'_1 = R_1 \oplus K_0.$$

We can eliminate this difference by adjusting the subkeys of the next equivalent round. Specifically, the subkey of the left branch becomes $K_0 + K_2$, while the subkey of the right branch becomes $K_1 + K_3$. Consequently, the output of the equivalent 2-round function differs from the original 4-round result:

$$L'_2 = L_2 \oplus K_3 \qquad R'_2 = R_2 \oplus K_2.$$

Regardless of the number of equivalent rounds, the output is equivalent to the original output plus a subkey. Although the output after an equivalent key transformation is different from the original one, it does not change the monomials of degree $n - 1$ and their coefficients. Therefore, if the equivalent cipher after an equivalent key transformation is integral-resistant, we can conclude that the original cipher has no integral distinguisher.

Twine can also take an equivalent key transformation. The equivalent round function after the equivalent key transformation is illustrated in Figure 5.

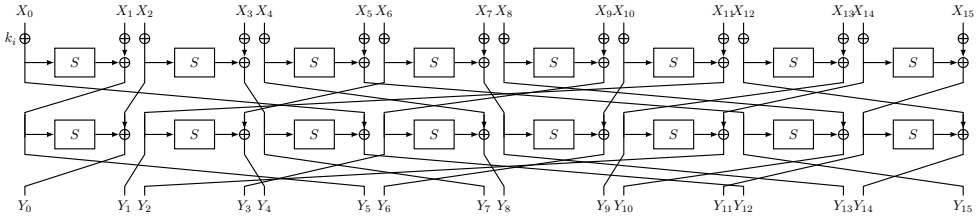


Fig. 5: Round function of Twine after equivalent transformation, where X_i and Y_i denote the input and output, respectively.

In the equivalent round, we observe that certain variables generate only specific variables, which can be viewed as a Supersbox.

$$X_0, X_1, X_2, X_3 \xrightarrow{Supersbox} Y_0, Y_5, Y_7, Y_{12}$$
$$X_4, X_5, X_8, X_9 \xrightarrow{Supersbox} Y_3, Y_8, Y_{10}, Y_{15}$$
$$X_6, X_7, X_{10}, X_{11} \xrightarrow{Supersbox} Y_1, Y_4, Y_6, Y_{13}$$
$$X_{12}, X_{13}, X_{14}, X_{15} \xrightarrow{Supersbox} Y_2, Y_9, Y_{11}, Y_{14}$$

The equivalent round function consists of a key addition, a Supersbox layer, and a permutation, which is similar to Present. The r-round Twine can also be written as Equation 2. Similarly, $\mathsf{E}_0 = \mathsf{K}_0 \circ \mathsf{S}_0$ can be represented by 4 parallel small functions composed of the Supersbox operation followed by a 16-bit key addition. E_2 can also be represented by 4 parallel small functions composed of a 16-bit key addition followed by the Supersbox operation. Thus, the integral-resistance matrix of Twine can be transformed to be a block diagonal matrix, where each block is determined by an $\mathsf{E}_{0,i}$ and $\mathsf{E}_{2,\ell}$.

$$\mathcal{I}(\mathsf{E})_{v_0,\cdots,v_{n^2-1}} = \begin{pmatrix} \boldsymbol{A_{0,0}} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{A_{0,1}} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{A_{3,3}} \end{pmatrix} \tag{8}$$

According to the same strategy of Lemma 3, it is straightforward to ensure the full rank of $\boldsymbol{A}_{i,\ell}$ by keeping $\boldsymbol{M}(\mathsf{E}_{0,i})$, $\boldsymbol{M}(\mathsf{E}_{2,\ell})$ full rank and searching all valid middle key monomials. For example, for $\boldsymbol{A}_{0,0}$, the details of the corresponding key monomial, along with the monomial trails of the first and last round, are provided in Appendix D. Unlike in Present, it is impossible to select a common ${x_1}^{u_1}$ for the 16 monomials ${x_0}^{\bar{e}_i}$, nor a common ${x_{r-1}}^{u_{r-1}}$ for the 16 monomials ${x_r}^{e_\ell}$.

1. We choose ${x_1}^{\bar{e}_{22}}$, ${x_1}^{\bar{e}_{23}}$, ${x_1}^{\bar{e}_{30}}$, ${x_1}^{\bar{e}_{31}}$ as ${x_1}^{u_1}$ for the 16 monomials of degree $n-1$.
2. We choose ${x_{r-1}}^{e_6}$, ${x_{r-1}}^{e_7}$, ${x_{r-1}}^{e_{46}}$ and ${x_{r-1}}^{e_{47}}$ as ${x_{r-1}}^{u_{r-1}}$ for the 16 monomials of degree 1.

The interfering monomials that exist are as follows:

1. ${x_1}^{\bar{e}_{22}}$ is the interfering monomial of ${x_1}^{\bar{e}_{23}}$;
2. ${x_1}^{\bar{e}_{30}}$ is the interfering monomial of ${x_1}^{\bar{e}_{31}}$;
3. ${x_{r-1}}^{e_6}$ is the interfering monomial of ${x_{r-1}}^{e_7}$;
4. ${x_{r-1}}^{e_{46}}$ is the interfering monomial of ${x_{r-1}}^{e_{47}}$.

Since $\boldsymbol{M}(\mathsf{E}_{0,0})$ and $\boldsymbol{M}(\mathsf{E}_{2,0})$ are of full rank, if all middle key monomials that take into account interfering monomials are searched out, the full-rank small block matrix $\boldsymbol{A}_{0,0}$ is formed.

5.2 Accelerating the Search for Middle Key Monomial

When the round function is complex, finding a middle key monomial becomes harder. Hebborn et al. encountered difficulties in a 12-round Skinny-64 due to its complex round function, as introduced in [17]. Since we treat 2-round Twine as one equivalent round, finding a valid key monomial is similarly challenging. To address this, we present some acceleration strategies.

5.2.1 Efficient Search Algorithm with a Backtracking Strategy

Since ${k_0}^{w_0}$, ${x_1}^{u_1}$ and ${k_{r-2}}^{w_{r-2}}$,${x_{r-1}}^{u_{r-1}}$ are determined, the goal of the search algorithm is to find a middle key monomial ${k_1}^{w_1} \dots {k_{r-3}}^{w_{r-3}}$ which satisfies $|{x_1}^{u_1}{k_1}^{w_1} \cdots {k_{r-3}}^{w_{r-3}} \bowtie {x_{r-1}}^{u_{r-1}}|$ is odd. The heuristic search process described in [5] proceeds as follows. Starting from the initial node $({k_{r-2}}^{w_{r-2}}, {x_{r-1}}^{u_{r-1}})$, the authors compute the corresponding predecessor monomials ${k_{r-3}}^{w_{r-3}}$ and ${x_{r-2}}^{u_{r-2}}$, which is called a backward propagation step. Based on the obtained node ${k_{r-3}}^{w_{r-3}}, {x_{r-2}}^{u_{r-2}}$, they further derive the monomials ${k_{r-4}}^{w_{r-4}}$ and ${x_{r-3}}^{u_{r-3}}$. By iteratively applying this procedure, the search propagates backward to the node $({k_{i-1}}^{w_{i-1}}, {x_i}^{u_i})$. If the Hamming weight of u_i exceeds 40, the backward propagation is terminated, and all remaining key monomials ${k_1}^{w_1} \cdots {k_{i-2}}^{w_{i-2}}$ are exhaustively searched in a single step. The process of searching the middle key monomial is illustrated as in Figure 6.

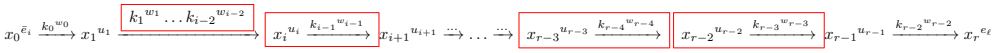


Fig. 6: The heuristic search algorithm in [5].

Note that in each backward propagation step, multiple candidate solutions may exist; however, in [5] only one node is randomly selected to proceed to the next step. During the search process, several failure scenarios may occur. Given a computed $({x_i}^{u_i}, {k_{i-1}}^{w_{i-1}})$, it may be impossible to derive any valid predecessor nodes $({x_{i-1}}^{u_{i-1}}, {k_{i-2}}^{w_{i-2}})$. Similarly, in the final step, the remaining key monomial may not be computable. In such cases, Hebborn *et al.* employ a restarting strategy, in which the entire search process is restarted from the initial node. As a consequence, the search procedure may incur significant overhead due to frequent restarts. Moreover, since a single candidate node is chosen randomly at each backward propagation step, the restarting strategy may lead to repeated attempts involving the same failing nodes.

In contrast, we adopt a backtracking strategy to avoid restarting the search from the initial node. During the backward propagation, multiple candidate solutions are stored at each step. Specifically, if a chosen $({x_i}^{u_i}, {k_{i-1}}^{w_{i-1}})$ cannot yield any valid predecessor nodes $({x_{i-1}}^{u_{i-1}}, {k_{i-2}}^{w_{i-2}})$, or if the remaining key monomial cannot be computed in the final step, the backtracking mechanism is triggered. In this case, the algorithm first returns to another stored node $({x_i}^{u_i}, {k_{i-1}}^{w_{i-1}})$ at the same round and continues the search process. If all nodes at this round lead to failure, the algorithm backtracks to the previous round and resumes the search from the corresponding stored nodes. This procedure continues until a valid key monomial is found or the backtracking stack becomes empty. The overall algorithm based on a stack is illustrated in Algorithm 1.

Algorithm 1: Our stack-based search algorithm for ${k_1}^{w_1} \dots {k_{r-3}}^{w_{r-3}}$

Input : the monomials ${x_1}^{u_1}$, ${k_{r-2}}^{w_{r-2}}$, ${x_{r-1}}^{u_{r-1}}$, and the MILP model $\mathcal{M}$
Output: the key monomial ${k_1}^{w_1} \dots {k_{r-3}}^{w_{r-3}}$

Initialize a stack $\mathbb{S}$ *to store all intermediate nodes*;
Initialize an array K *to store the recovered key monomials*;
Push $({x_{r-1}}^{u_{r-1}}, {k_{r-2}}^{w_{r-2}})$ *onto* $\mathbb{S}$;
while $\mathbb{S}$ *is not empty* **do**
  Pop the top node $({x_i}^{u_i}, {k_{i-1}}^{w_{i-1}})$ *from* $\mathbb{S}$;
  Set $K[i-1] \leftarrow {k_{i-1}}^{w_{i-1}}$;
  if $\mathrm{wt}(u_i) \geq 40$ **then**
    Solve the remaining rounds and enforce that $|{x_1}^{u_1} \times {k_1}^{w_1} \cdots {k_{i-2}}^{w_{i-2}} \bowtie {x_i}^{u_i}|$ *and* $|{x_1}^{u_1} \times {k_1}^{w_1} \cdots {k_{r-3}}^{w_{r-3}} \bowtie {x_{r-1}}^{u_{r-1}}|$ *are odd via Callback*;
    if *a feasible solution* ${k_1}^{w_1} \cdots {k_{i-2}}^{w_{i-2}}$ *is found* **then**
      for $j \leftarrow 1$ **to** $i-2$ **do**
        Set $K[j] \leftarrow {k_j}^{w_j}$;
      Terminate the search and return K;
  else
    Solve all solution $({x_{i-1}}^{u_{i-1}}, {k_{i-2}}^{w_{i-2}})$ *within a reasonable time, ensuring that* $|{x_{i-1}}^{u_{i-1}} \times {k_{i-2}}^{w_{i-2}} \bowtie {x_i}^{u_i}|$ *and* $|{x_{i-1}}^{u_{i-1}} \times {k_{i-2}}^{w_{i-2}} \cdots {k_{r-3}}^{w_{r-3}} \bowtie {x_{r-1}}^{u_{r-1}}|$ *are odd via Callback*;
    Push all feasible solutions onto $\mathbb{S}$;

Our algorithm stores computed nodes in a stack for backtracking, offering an efficient approach. We restart from these stored nodes rather than from scratch each time, which can help in avoiding repetitive attempts at failed nodes. An obvious example is when searching for valid key monomials in a 12-round Skinny-64, a valid key monomial from ${x_0}^{\bar{e}_0}$ to ${x_{12}}^{e_i}$, where $i \in \{26, 27, 42, 43, 59\}$ cannot be found by the algorithm of Hebborn et al., even if the computation time exceeds 1,500,000 CPU seconds. For our

algorithm, we can find a valid monomial no more than 40,000 CPU seconds. A comparative analysis of the running times of restarting strategy and backtracking strategy in both 12-round Present and 12-round Skinny-64 ciphers is shown in Figure 7.

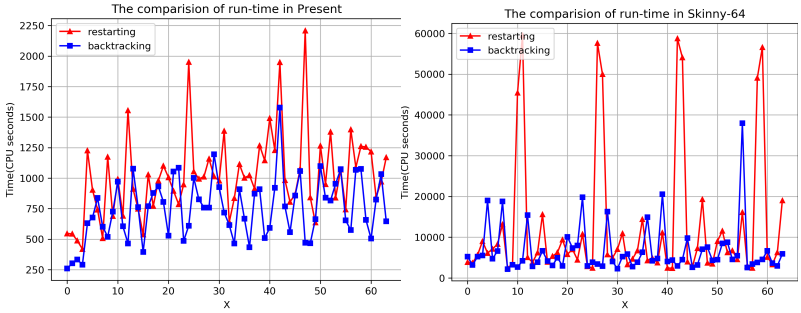


Fig. 7: Each data point gives the average search time over 64 key monomials for Present and 40 for Skinny-64. Each key monomial consists of a degree-$(n-1)$ monomial in the plaintext and a degree-1 monomial in the ciphertext that involves only the x-th bit; the horizontal coordinate of the data point is exactly this bit index x. For Skinny-64, because the restarting strategy may cause excessive runtime, we interrupt each search at 200,000 seconds and use this truncated value to compute the average.

5.2.2 Reducing Redundant Monomial Trails of Supersbox

In the proving process of Twine and Lblock, the MILP model of the equivalent round function is crucial when searching for a valid monomial trail(the operation COPY and XOR will cause large redundant monomial trails). Here, we use the first Supersbox in Twine as an example, illustrated in Figure 8, to explain how we model the Supersbox operation and how to accelerate the search process by reducing the redundant monomial trails.

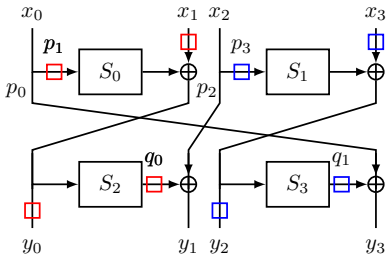


Fig. 8: The feature of the first Supersbox in the equivalent round. We designate the nibble x_i for $i \in \{0, \cdots, 3\}$ as the input, and the nibble y_j for $j \in \{0, \cdots, 3\}$ as the output. The Supersbox comprises four smaller Sboxes along with additional COPY and XOR operations. Furthermore, we denote the nibble p_1 (p_3) as the input for S_0 (S_1), and the nibble q_0 (q_1) as the output for S_2 (S_3).

In the MILP model, it is necessary to generate a set of inequalities to represent whether the monomial x^u can be contained in y^v as it passes through the Supersbox operation. The most precise approach is to directly model this propagation based on the ANFs of the Supersbox. However, due to the 16-bit input and output of the Supersbox, the overall model becomes overly complex, making it nearly impractical. Another model approach involves representing the propagation of the monomials by individually modelling each operation, including COPY, XOR, and the small Sbox. But this method introduces many variables, and many redundant trails may also pass through the Supersbox. When searching for a monomial trail of r rounds, we utilize Equation 1 to compute whether the number of monomial trails is odd. In this case, enumerating the number of trails becomes time-consuming.

Therefore, we model the monomial propagation of the Supersbox operation in a different way. It is worth noting that p_1, x_1 pass through a series of operations only to produce y_0 and q_0. We can treat the series operations as an equivalent Sbox operation, denoted as S'. Similarly, p_3 and x_3 passing through S' will produce y_2 and q_2 without other variables. As a result, the Supersbox operation can be split into the following operations,

$$
\begin{aligned}
&x_0 \xrightarrow{COPY} (p_0, p_1), \quad x_2 \xrightarrow{COPY} (p_2, p_3),\\
&(p_1, x_1) \xrightarrow{S'} (y_0, q_0), \quad (p_3, x_3) \xrightarrow{S'} (y_2, q_1),\\
&(q_0, x_2) \xrightarrow{XOR} y_1, \quad (q_1, x_0) \xrightarrow{XOR} y_3.
\end{aligned}
$$

In our method, we only need to introduce 6 additional variables, namely p_i for $0 \le i \le 3$ and q_i for $i \in \{0, 1\}$. Employing this model method, one equivalent round can eliminate approximately 17,000 even monomial trails that would emerge when modelling the Supersbox by individually modelling each operation. While enumerating the number of monomial trails is challenging, our model will reduce the counting time.

5.3 Results for 20-round Twine

To demonstrate the resistance of a 20-round Twine against integral distinguishers, we first need to take an equivalent key transformation. This transformation yields an equivalent 10-round cipher. As illustrated above, we determine the key monomials by ensuring the key monomial of the first round ${k_0}^{w_0}$ and the last round ${k_8}^{w_8}$ are determined first. Subsequently, we search for the remaining middle key monomials ${k_1}^{w_1} \cdots {k_7}^{w_7}$.

For each monomial of degree $n-1$ of x_0, the chosen ${x_1}^{u_1}$ are ${x_1}^{\bar{e}_{8i+6}}$ and ${x_1}^{\bar{e}_{8i+7}}$, where i ranges from 0 to 7. Similarly, for each monomial of degree 1 of x_{10}, we choose ${x_9}^{e_{8i+6}}$ or ${x_9}^{e_{8i+7}}$ as ${x_9}^{u_9}$, where i ranges from 0 to 7. From the monomial trails of the first and last round, which are obtained by solving a MILP model, we observe the following interfering monomials for the chosen monomials:

1. ${x_1}^{\bar{e}_{8i+6}}$ is the interfering monomial of ${x_1}^{\bar{e}_{8i+7}}$,
2. ${x_9}^{e_{8i+6}}$ is the interfering monomial of ${x_9}^{e_{8i+7}}$.

When searching for the middle round key monomial ${k_1}^{w_1}\cdots{k_7}^{w_7}$, it is crucial to account for interfering monomials and eliminate their influences. However, when we incorporate all constraints into the MILP model, the monomial trail may become non-existent. To address this, we relax the search space and utilize the Callback function of Gurobi to eliminate the impact of interfering monomials. By searching, we successfully obtain all 256 middle round key monomials. Consequently, we construct a full-rank integral-resistance matrix. This leads us to infer the absence of an integral distinguisher for the equivalent 10-round cipher, which mirrors the original 20-round Twine.

6 Security Guarantee of Lblock

Lblock is quite similar to Twine. It is a block cipher based on the balanced Feistel structure, whose round function consists of 8 4-bit Sboxes followed by a 4-bit block-wise permutation. A detailed introduction of Lblock is illustrated in Appendix E. Such a structure can be transformed into a further generalized Type-2 GFS as proposed by [24]. Therefore, by using an equivalent key transformation, the integral-resistance property can be directly applied to Lblock. Similar to Twine, the equivalent round function of Lblock consists of a key addition, a Supersbox layer, and a permutation. The strategies used in Present and Twine are also useful to Lblock. It should be noted that since all Sboxes used in Lblock are different, the Supersboxes formed are also different. Therefore, when choosing the key monomials for the first and last rounds, we should choose them separately according to the specific Supersbox.

For the 20-round Lblock, we first apply an equivalent key transformation to obtain an equivalent 10-round function. Next, we determine the monomials of the key by first specifying the key monomials for the first round, ${k_0}^{w_0}$, and the last round, ${k_8}^{w_8}$. Fortunately, there are no interfering monomials for the chosen monomials, allowing us to directly search for the middle round key monomials. Applying the same strategies of Present and Twine, we can ensure each block will be of full rank. Then, we search for the middle key monomials, ${k_1}^{w_1}\cdots{k_7}^{w_7}$, using the accelerating strategies of Twine. For every monomial of degree $n-1$ of x_0, we select a monomial ${x_1}^{u_1}$ and the corresponding ${k_0}^{w_0}$. The monomials ${x_1}^{u_1}$ chosen are ${x_1}^{e_{8i+2}}$ for $0 \le i \le 7$. Similarly, for each monomial of degree 1 of x_{10}, we choose ${x_9}^{e_{8i+2}}$ for $0 \le i \le 7$ as ${x_9}^{u_9}$. By employing the acceleration method, we can identify all 64 middle round key monomials. Consequently, we can form a full-rank integral-resistance matrix for Lblock. It can be inferred that there is no integral distinguisher for the equivalent 10-round Lblock, which corresponds to the original 20-round Lblock.

7 Conclusion

In this paper, we present some improved strategies for determining key monomials, which play a crucial role in proving the security of block ciphers against integral attacks under the assumption of independent round keys. We have narrowed the gap between the longest integral distinguisher and strong arguments against integral distinguishers for Present, demonstrating that 12 and 11 rounds of Present do not have any integral distinguishers, and all output bits except the 0-th bit of 10-round Present cannot give a constant sum. Besides, by an equivalent key transformation and some accelerating

strategy of searching monomial trails, we can confirm that 20-round Twine and Lblock do not have any integral distinguishers.

Further work is needed to prove the security of 10-round Present against integral attacks or to discover a 10-round integral distinguisher. Additionally, the requirement for the integral-resistance matrix to be of full rank as a condition for proving security against integral attacks is quite strict. It would be meaningful to explore a relaxed condition that guarantees security while allowing for more flexibility. Improving the search algorithm is also beneficial, particularly when dealing with complex round functions where finding a suitable middle round key monomial can be challenging. Although our search algorithm can find solutions, it may be time-consuming. Therefore, further enhancements to the search algorithm are warranted.

Appendix A MILP Model for the Monomial Trail

Next, we will give the monomial propagation rules for some basic operations.

Proposition 2. ***(COPY Propagation)****. Let* $\boldsymbol{f} : x \to (x, x)$*, the monomial trail* $\boldsymbol{x^u \to y^v}$ *can be* $0 \to (0,0)$*,* $1 \to (1,0)$ *or* $(0,1)$ *or* $(1,1)$*.*

For the COPY operation, MILP model $u \to (v_1, v_2)$ can use the following constraints

$$\begin{cases} u, v_1, v_2 \text{ are binary variables} \\ u = v_1 \vee v_2 \end{cases}$$

Proposition 3. ***(XOR Propagation)****.Let* $\boldsymbol{f} : (x_0, x_1) \to x_0 \oplus x_1$*, the monomial trail* $\boldsymbol{x^u \to y^v}$ *can be* $(0,0) \to 0$*,* $(1,0) \to 1$ *and* $(0,1) \to 1$*.*

For the XOR operation, MILP model $(u_1, u_2) \to v$ can use the following constraints

$$\begin{cases} u_1, v_2, v \text{ are binary variables} \\ v = u_1 + u_2 \end{cases}$$

Proposition 4. ***(AND Propagation)****.Let* $\boldsymbol{f} : (x_0, x_1) \to x_0 x_1$*, the monomial trail* $\boldsymbol{x^u \to y^v}$ *can be* $(0,0) \to 0$*,* $(1,1) \to 1$*.*

For the AND operation, MILP model $(u_1, u_2) \to v$ can use the following constraints

$$\begin{cases} u_1, u_2, v \text{ are binary variables} \\ u_1 = v, u_2 = v \end{cases}$$

For the nonlinear layer Sbox, we first obtain a monomial propagation table by ANF. To generate inequalities for monomial trails of each function, we follow Xiang et al.'s approach in [18] to derive linear inequalities by Sage [25] and then use the greedy algorithm to simplify them.

Appendix B Present Block Cipher

Present is a lightweight block cipher, published at CHES 2007 [26], with a 64-bit block size and a key size of either 80 or 128 bits. Its round function is very simple, consisting of a key addition, an Sbox layer of 4-bit Sboxes, and a bit permutation. After 31

rounds of iterations, one additional key addition is appended at the end. In this paper, we denote the Sbox, bit permutation and key addition in the i-th round by S_i, P_i and K_i, respectively. The whitening key is then denoted by K_{-1}. An r-round Present is represented as

$$\mathsf{E} = (\mathsf{K}_{r-1} \circ \mathsf{P}_{r-1} \circ \mathsf{S}_{r-1}) \circ \cdots \circ (\mathsf{K}_0 \circ \mathsf{P}_0 \circ \mathsf{S}_0) \circ \mathsf{K}_{-1} \tag{B1}$$

The Sbox is given by the following table:

Table B1: The Sbox of Present

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

The bit permutation changes bit positions as follows:

$$\mathsf{P}(i) = (i \bmod 4) \times 16 + \lfloor i/4 \rfloor.$$

It is worth noting that the round function of the Present features the utilization of the Supersbox. As a result, a depiction of the 2-round Present can be presented through the use of the Supersbox, as illustrated in Figure B1.

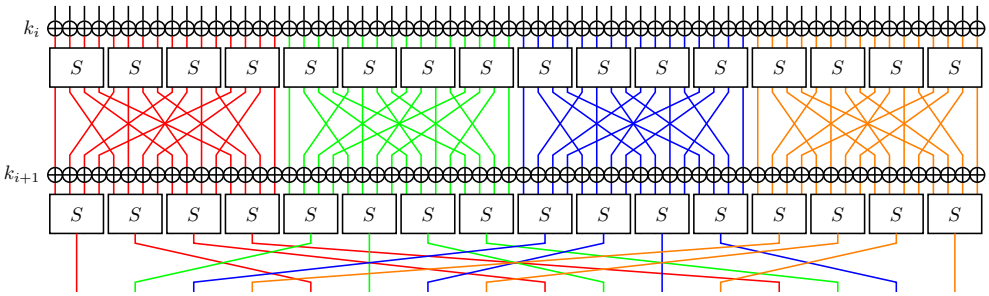


Fig. B1: Round function of Present using Supersbox.

We used a technique that was proposed in [5, 17], where we replace the Sbox in the first and last round by an affine equivalent one. This maintains the correctness of our results while making it much easier to compute.

Table B2: The Sbox in the first round of Present

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S'(x)$	8	5	0	2	A	9	4	C	E	7	B	D	6	1	F	3

Table B3: The Sbox in the first round of Present

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S''(x)$	6	7	E	3	1	0	8	D	9	4	F	A	C	5	B	2

Appendix C Twine Block Cipher

Twine is a family of 64-bit lightweight block ciphers with a generalized Feistel structure, designed by Suzaki et al. [27]. The family includes two members: Twine-80 and Twine-128, which support 80-bit and 128-bit keys, respectively. In this paper, we will treat the round keys as independent. The round functions of Twine are visualized in Figure C2, and the F-function consists of a key addition followed by an Sbox operation, which is illustrated in Table C4. For more details on the cipher, we refer the reader to [27].

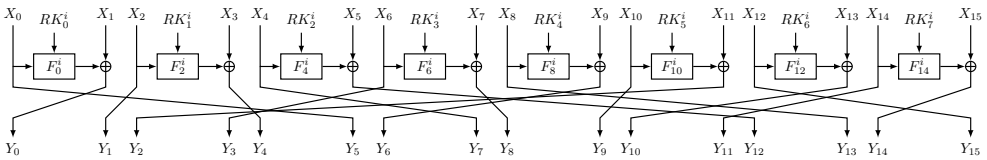


Fig. C2: Round function of Twine

Table C4: The Sbox of Twine

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	0	F	A	2	B	9	5	8	3	D	7	1	E	6	4

Appendix D Small Block $A_{0,0}$ of Twine

The monomial trials we chose for the first Supersbox in the first equivalent round are as follows:

$$
\begin{aligned}
{x_0}^{\bar{e}_0} &\to k_1[28]k_1[31]{x_1}^{\bar{e}_{30}}, & {x_0}^{\bar{e}_1} &\to k_1[29]k_1[31]{x_1}^{\bar{e}_{30}},\\
{x_0}^{\bar{e}_2} &\to k_1[28]k_1[29]{x_1}^{\bar{e}_{30}}, & {x_0}^{\bar{e}_3} &\to k_1[28]k_1[29]{x_1}^{\bar{e}_{31}},\\
{x_0}^{\bar{e}_4} &\to k_1[28]k_1[31]k_1[50]k_1[51]{x_1}^{\bar{e}_{30}}, & {x_0}^{\bar{e}_5} &\to k_1[29]k_1[31]k_1[50]k_1[51]{x_1}^{\bar{e}_{30}},\\
{x_0}^{\bar{e}_6} &\to k_1[28]k_1[29]k_1[48]k_1[49]{x_1}^{\bar{e}_{30}}, & {x_0}^{\bar{e}_7} &\to k_1[29]k_1[31]k_1[48]k_1[51]{x_1}^{\bar{e}_{30}},\\
{x_0}^{\bar{e}_8} &\to k_1[20]k_1[23]{x_1}^{\bar{e}_{22}}, & {x_0}^{\bar{e}_9} &\to k_1[21]k_1[23]{x_1}^{\bar{e}_{22}},\\
{x_0}^{\bar{e}_{10}} &\to k_1[20]k_1[21]{x_1}^{\bar{e}_{22}}, & {x_0}^{\bar{e}_{11}} &\to k_1[20]k_1[21]{x_1}^{\bar{e}_{23}},\\
{x_0}^{\bar{e}_{12}} &\to k_1[0]k_1[3]k_1[20]k_1[21]{x_1}^{\bar{e}_{22}}, & {x_0}^{\bar{e}_{13}} &\to k_1[1]k_1[3]k_1[20]k_1[21]{x_1}^{\bar{e}_{22}},\\
{x_0}^{\bar{e}_{14}} &\to k_1[0]k_1[1]k_1[20]k_1[21]{x_1}^{\bar{e}_{22}}, & {x_0}^{\bar{e}_{15}} &\to k_1[0]k_1[3]k_1[21]k_1[23]{x_1}^{\bar{e}_{22}}.
\end{aligned}
$$

Then, we can get all key monomials of the first equivalent round, the monomial trail of $\mathsf{E}_{0,0}$ is as follows,

Table D5: Monomial trails of $\mathsf{E}_{0,0}$. $\varnothing$ represents no monomial trails and $\star$ represents that the monomial trail is uncertain but not irrelevant in computing the rank of $A_{0,0}$. $\bar{e}_{(r)}$ denotes the monomial of ${x_1}^{\bar{e}_i}$ and σ_i denotes the corresponding chosen key monomial.

key \ input	${x_0}^{\bar{e}_0}$	${x_0}^{\bar{e}_1}$	${x_0}^{\bar{e}_2}$	${x_0}^{\bar{e}_3}$	${x_0}^{\bar{e}_4}$	${x_0}^{\bar{e}_5}$	${x_0}^{\bar{e}_6}$	${x_0}^{\bar{e}_7}$	${x_0}^{\bar{e}_8}$	${x_0}^{\bar{e}_9}$	${x_0}^{\bar{e}_{10}}$	${x_0}^{\bar{e}_{11}}$	${x_0}^{\bar{e}_{12}}$	${x_0}^{\bar{e}_{13}}$	${x_0}^{\bar{e}_{14}}$	${x_0}^{\bar{e}_{15}}$
σ_0	$\bar{e}_{30}$	$\varnothing$	$\varnothing$	$\star$	$\star$	$\star$	$\star$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
σ_1	$\varnothing$	$\bar{e}_{30}$	$\varnothing$	$\star$	$\star$	$\star$	$\star$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
σ_2	$\varnothing$	$\varnothing$	$\bar{e}_{30}$	$\star$	$\star$	$\star$	$\star$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
σ_3	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{31}$	$\star$	$\star$	$\star$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
σ_4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{30}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
σ_5	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{30}$	$\varnothing$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
σ_6	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{30}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
σ_7	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{30}$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
σ_8	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{22}$	$\varnothing$	$\varnothing$	$\star$	$\star$	$\star$	$\star$	$\star$
σ_9	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{22}$	$\varnothing$	$\star$	$\star$	$\star$	$\star$	$\star$
σ_{10}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{22}$	$\star$	$\star$	$\star$	$\star$	$\star$
σ_{11}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{23}$	$\star$	$\star$	$\star$	$\star$
σ_{12}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{22}$	$\varnothing$	$\varnothing$	$\varnothing$
σ_{13}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{22}$	$\varnothing$	$\star$
σ_{14}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{22}$	$\star$
σ_{15}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\bar{e}_{22}$

The monomial trials we chose for the first Supersbox in the last equivalent round are as follows:

$$
\begin{aligned}
&k_{r-2}[0]k_{r-2}[3]{x_{r-1}}^{e_6} \to {x_r}^{e_0}, \quad k_{r-2}[1]k_{r-2}[3]{x_{r-1}}^{e_6} \to {x_r}^{e_1},\\
&k_{r-2}[0]k_{r-2}[1]{x_{r-1}}^{e_6} \to {x_r}^{e_2}, \quad k_{r-2}[0]k_{r-2}[1]{x_{r-1}}^{e_7} \to {x_r}^{e_3},\\
&k_{r-2}[0]k_{r-2}[1]k_{r-2}[4]k_{r-2}[7]{x_{r-1}}^{e_6} \to {x_r}^{e_4},\\
&k_{r-2}[0]k_{r-2}[1]k_{r-2}[5]k_{r-2}[7]{x_{r-1}}^{e_6} \to {x_r}^{e_5},\\
&k_{r-2}[0]k_{r-2}[1]k_{r-2}[4]k_{r-2}[5]{x_{r-1}}^{e_6} \to {x_r}^{e_6},\\
&k_{r-2}[0]k_{r-2}[3]k_{r-2}[5]k_{r-2}[7]{x_{r-1}}^{e_6} \to {x_r}^{e_7},\\
&k_{r-2}[8]k_{r-2}[11]{x_{r-1}}^{e_{46}} \to {x_r}^{e_{16}}, \quad k_{r-2}[9]k_{r-2}[11]{x_{r-1}}^{e_{46}} \to {x_r}^{e_{17}},\\
&k_{r-2}[8]k_{r-2}[9]{x_{r-1}}^{e_{46}} \to {x_r}^{e_{18}}, \quad k_{r-2}[8]k_{r-2}[9]{x_{r-1}}^{e_{47}} \to {x_r}^{e_{19}},\\
&k_{r-2}[8]k_{r-2}[9]k_{r-2}[12]k_{r-2}[15]{x_{r-1}}^{e_{46}} \to {x_r}^{e_{20}},\\
&k_{r-2}[8]k_{r-2}[9]k_{r-2}[13]k_{r-2}[15]{x_{r-1}}^{e_{46}} \to {x_r}^{e_{21}},\\
&k_{r-2}[8]k_{r-2}[9]k_{r-2}[12]k_{r-2}[13]{x_{r-1}}^{e_{46}} \to {x_r}^{e_{22}},\\
&k_{r-2}[8]k_{r-2}[11]k_{r-2}[13]k_{r-2}[15]{x_{r-1}}^{e_{46}} \to {x_r}^{e_{23}}.
\end{aligned}
$$

Then, we can get all key monomials of the last equivalent round, the monomial trail of $\mathsf{E}_{2,0}$ is illustrated in Table D6.

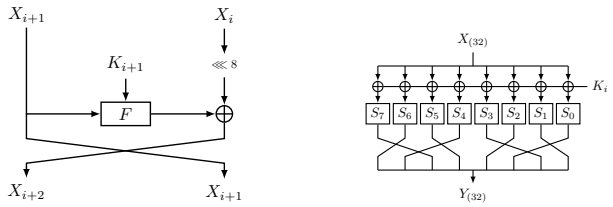
According to the same strategy in Lemma 3, considering the influence of interfering monomials, if we obtain all the middle round key monomials, $\boldsymbol{A}_{0,0}$ is of full rank.

Table D6: Monomial trails of $\mathsf{E}_{2,0}$. $\varnothing$ represents no monomial trails, and $\star$ represents that the monomial trail is uncertain but not relevant in computing the rank of $A_{0,0}$. e_i denotes the monomial of ${x_{r-1}}^{e_i}$ and τ_i denotes the corresponding chosen key monomial.

key \ output	${x_r}^{e_0}$	${x_r}^{e_1}$	${x_r}^{e_2}$	${x_r}^{e_3}$	${x_r}^{e_4}$	${x_r}^{e_5}$	${x_r}^{e_6}$	${x_r}^{e_7}$	${x_r}^{e_{16}}$	${x_r}^{e_{17}}$	${x_r}^{e_{18}}$	${x_r}^{e_{19}}$	${x_r}^{e_{20}}$	${x_r}^{e_{21}}$	${x_r}^{e_{22}}$	${x_r}^{e_{23}}$
τ_r	e_6	$\varnothing$	$\varnothing$	$\star$	$\star$	$\star$	$\star$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
τ_1	$\varnothing$	e_6	$\varnothing$	$\star$	$\star$	$\star$	$\star$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
τ_2	$\varnothing$	$\varnothing$	e_6	$\star$	$\star$	$\star$	$\star$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
τ_3	$\varnothing$	$\varnothing$	$\varnothing$	e_7	$\star$	$\star$	$\star$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
τ_4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_6	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
τ_5	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_6	$\varnothing$	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
τ_6	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_6	$\star$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
τ_7	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_6	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
τ_8	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_{46}	$\varnothing$	$\varnothing$	$\star$	$\star$	$\star$	$\star$	$\star$
τ_9	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_{46}	$\varnothing$	$\star$	$\star$	$\star$	$\star$	$\star$
τ_{10}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_{46}	$\star$	$\star$	$\star$	$\star$	$\star$
τ_{11}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_{47}	$\star$	$\star$	$\star$	$\star$
τ_{12}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_{46}	$\varnothing$	$\varnothing$	$\varnothing$
τ_{13}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_{46}	$\varnothing$	$\varnothing$
τ_{14}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_{46}	$\varnothing$
τ_{15}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	e_{46}

Appendix E Lblock Block Cipher

LBlock is a lightweight 64-bit block cipher with an 80-bit key designed by Wu et al. in 2011 [28]. It is designed based on a variant of the Feistel structure and contains 32 rounds. The round function and the key schedule algorithms for LBlock are visualized in Figure E3. We refer the readers to [28] for more details of the cipher.



(a) The round function of Lblock

(b) The F-function of round function

Fig. E3: The fearture of Lblock.

Table E7: The Sbox of Lblock

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_0(x)$	E	9	F	0	D	4	A	B	1	2	8	3	7	6	C	5
$S_1(x)$	4	B	E	9	F	D	0	A	7	C	5	6	2	8	1	3
$S_2(x)$	1	E	7	C	F	D	0	6	B	5	9	3	2	4	8	a
$S_3(x)$	7	6	8	B	0	F	3	E	9	A	C	D	5	2	4	1
$S_4(x)$	E	5	F	0	7	2	C	D	1	8	4	9	B	A	6	3
$S_5(x)$	2	D	B	C	F	E	0	9	7	A	6	3	1	8	4	5
$S_6(x)$	B	9	4	E	0	F	A	D	6	C	5	7	3	8	1	2
$S_7(x)$	D	A	F	0	E	4	9	B	2	1	8	3	7	5	12	6

References

[1] Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers. Lecture Notes in Computer Science, pp. 112–127. Springer, Leuven (2002). https://doi.org/10.1007/3-540-45661-9_9

[2] Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings. Lecture Notes in Computer Science, pp. 149–165. Springer, Haifa (1997). https://doi.org/10.1007/BFB0052343

[3] Matsui, M.: New block encryption algorithm MISTY. In: Biham, E. (ed.) Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1267, pp. 54–68. Springer, Haifa (1997). https://doi.org/10.1007/BFB0052334

[4] Todo, Y.: Integral cryptanalysis on full MISTY1. J. Cryptol. **30**(3), 920–959 (2017) https://doi.org/10.1007/S00145-016-9240-X

[5] Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Strong and tight security guarantees against integral distinguishers. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I. Lecture Notes in Computer Science, pp. 362–391. Springer, Singapore (2021). https://doi.org/10.1007/978-3-030-92062-3_13

[6] Beierle, C., Hebborn, P., Leander, G., Perehuda, Y.: Integral resistance of block ciphers with key whitening by modular addition. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part V. Lecture Notes in Computer Science, pp. 497–529. Springer, Santa Barbara (2025). https://doi.org/10.1007/978-3-032-01901-1_16

[7] Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, pp. 287–314. Springer, Sofia (2015). https://doi.org/10.1007/978-3-662-46800-5_12

[8] Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: MILP-aided method of searching division property using three subsets and applications. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 398–427. Springer, Kobe (2019).

https://doi.org/10.1007/978-3-030-34618-8_14

[9] Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset - improved cube attacks against trivium and grain-128aead. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 466–495. Springer, Zagreb (2020). https://doi.org/10.1007/978-3-030-45721-1_17

[10] Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset. J. Cryptol. **34**(3), 22 (2021) https://doi.org/10.1007/s00145-021-09383-2

[11] Boura, C., Canteaut, A.: Another view of the division property. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, pp. 654–682. Springer, Santa Barbara (2016). https://doi.org/10.1007/978-3-662-53018-4_24

[12] Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I. Lecture Notes in Computer Science, pp. 446–476. Springer, Daejeon (2020). https://doi.org/10.1007/978-3-030-64837-4_15

[13] Beyne, T., Verbauwhede, M.: Integral cryptanalysis using algebraic transition matrices. IACR Trans. Symmetric Cryptol. **2023**(4), 244–269 (2023) https://doi.org/10.46586/TOSC.V2023.I4.244-269

[14] Beyne, T., Verbauwhede, M.: Ultrametric integral cryptanalysis. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part VII. Lecture Notes in Computer Science, vol. 15490, pp. 392–423. Springer, Kolkata (2024). https://doi.org/10.1007/978-981-96-0941-3_13

[15] Zeng, F., Tian, T.: A new method for constructing integral-resistance matrix for 5-round AES. IET Inf. Secur. **2025**(1) (2025) https://doi.org/10.1049/ISE2/3447652

[16] Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 357–377. Springer, Bochum (2016).

https://doi.org/10.1007/978-3-662-52993-5_18

[17] Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Lower bounds on the degree of block ciphers. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 537–566. Springer, Daejeon (2020). https://doi.org/10.1007/978-3-030-64837-4_18

[18] Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 648–678 (2016). https://doi.org/10.1007/978-3-662-53887-6_24

[19] Boura, C., Coggia, D.: Efficient MILP modelings for sboxes and linear layers of SPN ciphers. IACR Trans. Symmetric Cryptol. **2020**(3), 327–361 (2020) https://doi.org/10.13154/TOSC.V2020.I3.327-361

[20] Sasaki, Y., Todo, Y.: New algorithm for modeling S-box in MILP based differential and division trail search. In: Farshim, P., Simion, E. (eds.) Innovative Security Solutions for Information Technology and Communications - 10th International Conference, SecITC 2017, Bucharest, Romania, June 8-9, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10543, pp. 150–165. Springer, Bucharest (2017). https://doi.org/10.1007/978-3-319-69284-5_11

[21] Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 158–178. Springer, Kaoshiung (2014). https://doi.org/10.1007/978-3-662-45611-8_9

[22] Gurobi Optimization. https://www.gurobi.com

[23] Gurobi Optimization Reference Manual. https://docs.gurobi.com/_/downloads/optimizer/en/12.0/pdf/

[24] Suzaki, T., Minematsu, K.: Improving the generalized Feistel. In: Hong, S., Iwata, T. (eds.) Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6147, pp. 19–39. Springer, Seoul (2010). https://doi.org/

10.1007/978-3-642-13858-4_2

[25] Sagemath. https://www.sagemath.org/

[26] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings. Lecture Notes in Computer Science, pp. 450–466. Springer, Vienna (2007). https://doi.org/10.1007/978-3-540-74735-2_31

[27] Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE : A lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7707, pp. 339–354. Springer, Windsor (2012). https://doi.org/10.1007/978-3-642-35999-6_22

[28] Wu, W., Zhang, L.: LBlock: A lightweight block cipher. In: López, J., Tsudik, G. (eds.) Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6715, pp. 327–344 (2011). https://doi.org/10.1007/978-3-642-21554-4_19