IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–31. DOI:XXXXXXXX

Trail-Estimator: An Automated Verifier for Differential Trails in Block Ciphers

Thomas Peyrin[1], Quan Quan Tan[1], Hongyi Zhang[1]✉ and Chunning Zhou[1]

Nanyang Technological University, Singapore, Singapore
thomas.peyrin@ntu.edu.sg, quanquan001@e.ntu.edu.sg, hongyi003@e.ntu.edu.sg, chunning.zhou@ntu.edu.sg

Abstract. Differential cryptanalysis is a powerful technique for attacking block ciphers, wherein the Markov cipher assumption and stochastic hypothesis are commonly employed to simplify the search and probability estimation of differential trails. However, these assumptions often neglect inherent algebraic constraints, potentially resulting in invalid trails and inaccurate probability estimates. Some studies identified violations of these assumptions and explored how they impose constraints on key material, but they have not yet fully captured all relevant ones. This study proposes `Trail-Estimator`, an automated verifier for differential trails on block ciphers, consisting of two parts: a constraint detector `Cons-Collector` and a solving tool `Cons-Solver`. We first establish the fundamental principles that will allow us to systematically identify all constraint subsets within a differential trail, upon which `Cons-Collector` is built. Then, `Cons-Solver` utilizes specialized preprocessing techniques to efficiently solve the detected constraint subsets, thereby determining the key space and providing a comprehensive probability distribution of differential trails. To validate its effectiveness, `Trail-Estimator` is applied to verify 14 differential trails for the `SKINNY`, `LBLOCK`, and `TWINE` block ciphers. Experimental results show that `Trail-Estimator` consistently identifies previously undetected constraints for `SKINNY` and discovers constraints for the first time for `LBLOCK` and `TWINE`. Notably, it is the first tool to discover long nonlinear constraints extending beyond five rounds in these ciphers. Furthermore, `Trail-Estimator`'s accuracy is validated by experiments showing its predictions closely match the real probability distribution of short-round differential trails.

Keywords: Block ciphers · Differential cryptanalysis · Constraint detection · Probability estimation

1 Introduction

Differential cryptanalysis is a powerful statistical technique for analyzing symmetric-key ciphers, relying on differential trails as distinguishers [BS90]. Over the decades, cryptanalysts have sought to identify the most effective distinguishers, either in terms of the highest probability or as components of key-recovery attacks. However, a significant limitation of the conventional approach to identifying differential trails is its dependence on assumptions, namely the Markov cipher assumption and the hypothesis of stochastic equivalence [LMM91]. Despite numerous efforts to highlight the importance of these assumptions, they have not gained widespread adoption, primarily due to the lack of a systematic approach to handling them and verifying their validity.

Biryukov and Wagner [BW99] investigated these dependencies in SPN-based designs, demonstrating that non-Markov effects can either weaken or strengthen differential attacks depending on the cipher structure. In [KM04], Knudsen and Mathiassen experimentally

analyzed the impact of simple and complex key schedules, finding that the probabilities of the best differentials in ciphers with simple key schedules do not always converge toward a uniform distribution as the number of rounds increases. Given the trend in modern cipher design favoring simple round functions and key schedules, it is unsurprising that conventional assumptions are becoming increasingly inadequate. Murphy *et al.* [MR02] further showed that certain key-dependent transformations introduce statistical dependencies, violating the Markov assumption.

Daemen and Rijmen [DR07] examined the probability distribution of differential characteristics across different keys, introducing the concept of plateau characteristics. Their findings demonstrated that for AES, all two-round characteristics exhibit plateau behavior. Further work by Dunkelman *et al.* [DKS10] revealed that ciphers with key-dependent S-boxes could exhibit unpredictable differential properties, thereby reducing the effectiveness of differential cryptanalysis.

In recent years, several studies have attempted to address these issues from both theoretical and practical perspectives. Leurent [Leu12] investigated differential attacks on ARX constructions and introduced a tool for deriving multi-bit conditions and detecting inconsistencies. Sun *et al.* [SWW18] analyzed the dependencies between differential trails and key scheduling, deriving new key constraints for `LED`64 and `Midori`64. Liu *et al.* [LIM20] demonstrated that certain differential trails in the `Gimli` permutation were infeasible and developed a Mixed-Integer Linear Programming model that accounts for differential and value transitions. Beyne and Rijmen [BR22] approached the problem from a theoretical perspective, proposing the concept of quasidifferential trails, which is analogous to their counterparts, linear trails, that explicitly considers the (fixed) key. Peyrin and Tan [PT22] developed a method for detecting linear and nonlinear constraints in differential trails for `SKINNY` and `GIFT-64`, invalidating several previously accepted paths. Sun [Sun24] extended this approach by introducing linearized nonlinear constraints, which extract linear relationships between the input and output bits of an S-box. More recently, Nageler *et al.* [NGJE25] proposed an SAT-based approach to estimate the average probability of a differential trail across all keys. Their method also provides upper bounds on key space size and derives necessary key constraints.

Despite significant progress in verifying differential trails, existing methods still have notable limitations. While the detection framework in [PT22] identifies key-dependent constraints, it fails to capture certain nonlinear dependencies, leading to inaccuracies in probability estimation. Sun [Sun24] improves upon this by linearizing some nonlinear constraints; however, this approach can only help to resolve some of the problems. The SAT-based technique in [NGJE25] provides probabilistic estimates but does not fully account for key-dependent constraints, which can significantly affect probability calculations.

Our Contributions. In this paper, we propose `Trail-Estimator`, a novel framework for identifying and analyzing constraints within differential trails that can be applied to all word-oriented block ciphers. Our framework comprises two core functionalities: `Cons-Collector` and `Cons-Solver`. `Cons-Collector` propagates constraints through both linear and nonlinear equations, capturing comprehensive dependency relationships between variables. We provide a proof establishing a principle for identifying all equation structures that contribute to effective key constraints and introduce an efficient detection method. By leveraging this approach, `Cons-Collector` identifies all possible constraints in a differential trail, including previously unaccounted constraints. The identified constraints are subsequently processed by `Cons-Solver`, a solver equipped with optimization techniques that significantly reduce the computational complexity of constraint solving.

We apply our framework to differential trails of `SKINNY`, `LBLOCK`, and `TWINE`. Compared to previous analyses of `SKINNY`, we demonstrate that these additional constraints, previously overlooked, further impact the probability distribution. A summary of our results is

presented in Table 1 and Table 2, highlighting the differences in the number of identified constraints and the updated probability distributions for the evaluated trails.

Table 1: Results comparison between `Trail-Estimator` and previous works. We note that no long nonlinear constraint could be found by any previous work, while they can find all linear constraints.

Cipher	R	#Short NLC				#Long NLC		#LC	Trail Source
		Sec. 5	[PT22]	[Sun24]	[NGJE25]	Orig.(Solv.)	Merg.		
SKINNY	7	2	0	0	-	4 (1)	1	3	Table 6 [DDH+21]
	10	0	0	0	-	4 (0)	1	3	Table 7 [DDH+21]
	13	0	0	0	-	7 (0)	1	2	Table 8 [DDH+21]
	15	1	1	1	*	5 (0)	1	2	Table 9 [DDH+21]
	14	4	3	-	-	14 (0)	1	13	Table 10 [DDH+21]
	16	6	4	-	-	16 (0)	1	13	Table 11 [DDH+21]
	17	13	-	-	*	5 (0)	1	13	Table 12 [DDH+21]
	8	0	0	-	-	0	0	1	Table 16 [QDW+21]
	10	0	0	-	-	4 (0)	1	3	Table 7 [PT22]
	11	6	-	1	-	7 (0)	1	7	Figure 2 [ZZ18]
LBLOCK	4	1	-	-	-	0	0	0	Table 9 Appendix F
	16	2	-	-	-	9 (2)	1	8	Table 14 [ZZDX19]
TWINE	9	0	-	-	-	3 (3)	3	0	Table 12 Appendix G
	15	2	-	-	-	8 (3)	1	6	Table 16 [ZZDX19]

R: the round number of the differential trail; #Short NLC (resp. #Long NLC): the number of nonlinear constraints on key values, which spanned ≤ 5 rounds (resp. more than five rounds); Orig.: the original number of long nonlinear constraints before merging; Merg.: the number of nonlinear constraints after merging the related constraints; Solv.: the number of solvable long nonlinear constraints before merging; #LC: the number of linear constraints identified on the key values.

Outline. Section 2 provides an overview of the background and related works. Section 3 introduces the `Cons-Collector` functionality of `Trail-Estimator`, which includes the detection principles and propagation rules. Section 4 presents `Cons-Solver`, which includes the optimization techniques and the method we use to derive the probability distribution. In Section 5, we show the results on `SKINNY-64`, `SKINNY-128`, `LBLOCK` and `TWINE` block ciphers when applying `Trail-Estimator` to them. Finally, we conclude and discuss our work in Section 6.

2 Preliminaries

2.1 Differential Cryptanalysis

Differential cryptanalysis [BS90] is the study of how differences in the plaintexts affect differences in ciphertexts as they propagate through a cipher. The main essence of differential cryptanalysis is caputred in the definition of the so-called differential trails.

Definition 1 (1-round differential trail [BS90])**.** Given a vectorial Boolean function $F : \{0,1\}^n \to \{0,1\}^n$, a one-round differential trail of F is defined as a pair $(\Delta_{in}, \Delta_{out})$, where Δ_{in} and Δ_{out} denote the n-bit input and output differences of F. The probability can be computed as follows: $P(\Delta_{in} \to \Delta_{out}) = \frac{\#\{F(x)\oplus F(x\oplus\Delta_{in})=\Delta_{out}, \forall\ x\ \in \mathbb{F}_2^n\}}{2^n}$.

While this definition captures what we truly look out for in a cipher, modern ciphers are usually too complex for it to be practically useful. Fortunately, the modern ciphers can be decomposed into multiple round functions, allowing us to trace input and output differences throughout these simpler functions.

Definition 2 (r-round differential trail [BS90])**.** Consider the composition of multiple vectorial Boolean functions $F = F_{r-1} \circ F_{r-2} \circ ... \circ F_0$, an r-round differential trail of F

Table 2: The analysis of SKINNY, LBLOCK and TWINE differential trails with Trail-Estimator.

Cipher	R	Stated Prob.	Reduced Key Space	Prob. by Trail-Estimator	Trail Source
SKINNY	7	2^{-52}	$2^{-7.3}$	$2^{-49} - 2^{-42.4}$	Table 6 [DDH+21]
	10	2^{-46}	0	-	Table 7 [DDH+21]
	13	2^{-55}	2^{-4}	2^{-51}	Table 8 [DDH+21]
	15	2^{-54}	$2^{-6.2}$	$2^{-48} - 2^{-47}$	Table 9 [DDH+21]
	14	2^{-120}	$2^{-10.4}$	$2^{-119.9} - 2^{-115.7}$	Table 10 [DDH+21]
	16	$2^{-127.6}$	$2^{-11.1}$	$2^{-128.2} - 2^{-108.2}$	Table 11 [DDH+21]
	17	2^{-110}	0	-	Table 12 [DDH+21]
	8	$2^{-19} - 2^{-17}$	$2^{-2}/2^{-3}$	$2^{-16} - 2^{-15}$	Table 16 [QDW+21]
	10	2^{-42}	1	2^{-42}	Table 7 [PT22]
	11	2^{-147}	0	-	Figure 2 [ZZ18]
LBLOCK	4	2^{-18}	1	2^{-18}	Table 9 Appendix F
	16	2^{-72}	2^{-1}	$2^{-72.7} - 2^{-69.9}$	Table 14 [ZZDX19]
TWINE	9	2^{-28}	$2^{-0.19}$	$2^{-31.9} - 2^{-25.1}$	Table 12 Appendix G
	15	2^{-68}	$2^{-4.7}$	$2^{-67} - 2^{-61.3}$	Table 16 [ZZDX19]

Stated Prob.: the probability reported by the original authors; Reduced Key Space: the proportion of the estimated valid key space for the differential trail; Prob. by Trail-Estimator: the probability estimation of differential trails within valid key space, given by Trail-Estimator;

is denoted as a tuple $\vec{\Delta} = (\Delta_{in} = \Delta_0, \Delta_1, ..., \Delta_r = \Delta_{out})$, where Δ_i is the n-bit output difference of $F_{i-1} \circ ... \circ F_0$ $(0 \leq i \leq r)$.

However, calculating the exact probability of an r-round differential trail is not an easy task, thus, cryptographers have to rely on the Markov cipher assumption.

Definition 3 (Markov cipher [LMM91])**.** An iterated cipher with the round function $Y = \mathcal{C}(X, k)$ is said to be a Markov cipher if there exists a group operation $\otimes$ for defining differences such that, for any nonzero choices of Δ_{in} and Δ_{out}, the probability

$$P(\Delta_Y = \Delta_{\text{out}} \mid \Delta_X = \Delta_{\text{in}}, X = \gamma) = P(\Delta_Y = \Delta_{\text{out}} \mid \Delta_X = \Delta_{\text{in}}),$$

when the subkey k is uniformly random.

In other words, the probability of a difference transition from Δ_{in} to Δ_{out} is independent of the value of X.

Here, we will also introduce the hypothesis of stochastic equivalence. Under this hypothesis, the probability of a differential trail remains approximately the same across all key values.

Definition 4 (Hypothesis of Stochastic Equivalence [LMM91])**.** For an r-round differential trail $(\Delta_{in} = \Delta_0, \Delta_1, ..., \Delta_r = \Delta_{out})$:

$$P(\Delta_r = p_r | \Delta_0 = p_0) \approx P(\Delta_r = p_r | \Delta_0 = p_0, k_1 = \beta_1, \cdots, k_r = \beta_r),$$

for almost all subkey values $(k_1 = \beta_1, \cdots, k_r = \beta_r)$ where k_i is the subkey of round i within the cipher $\mathcal{C}$.

Definition 5 (XDDT and YDDT [PT22])**.** The difference distribution table (DDT) is a tool that represents the distribution of all possible input and output difference pairs for a Boolean function F. However, DDT only captures the information of the difference and not the state. To retain the information of the state, we define XDDT (resp. YDDT) to be the

set containing all the valid input (resp. output) values that allow the differential transition $\Delta_{in} \to \Delta_{out}$ to happen:

$$\texttt{XDDT}(\Delta_{in}, \Delta_{out}) := \{x | F(x) \oplus F(x \oplus \Delta_{in}) = \Delta_{out}, x \in \mathbb{F}_2^n\},$$
$$\texttt{YDDT}(\Delta_{in}, \Delta_{out}) := \{F(x) | F(x) \oplus F(x \oplus \Delta_{in}) = \Delta_{out}, x \in \mathbb{F}_2^n\}.$$

2.2 Constraint Detection Method

In [PT22], the authors introduced a novel framework to identify potential constraints on key variables within a cipher. They start from the constrained inputs x_i and outputs y_i of active S-boxes in the differential trail T, which are called *half constraints* since they cannot impose constraints on the key variables by themselves. Instead, they require pairing with another half constraint to establish a complete constraint on the key variables.

Consider the r-th round of an arbitrary block cipher. It can usually be divided into three main components: a nonlinear layer S^r, a linear layer L^r, and a key addition layer (in that order). In an SPN cipher like `SKINNY` [BJK+16], we can denote the input variables of the round (also the input of S^r) as $X^r = \{x_0^r, x_1^r, \cdots, x_t^r\}$ and the corresponding output variables of S^r as $Y^r = \{y_0^r, y_1^r, \cdots, y_t^r\}$. The set of round keys is denoted as $K^r = \{k_0^r, k_1^r, \cdots, k_q^r\}$. If x_i^{r+1} and all the related y_i^r are constrained, then there is a full constraint on K^r. We will illustrate the concept below.

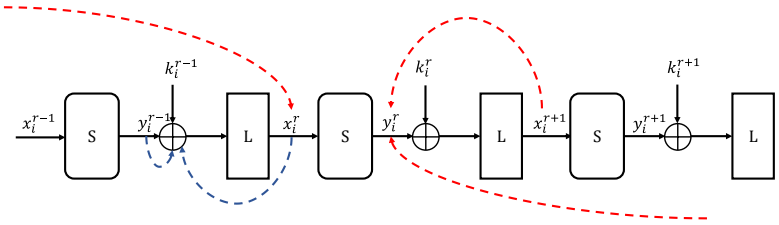


Figure 1: The formation of linear constraint (blue line) and nonlinear constraint (red line)

According to Figure 1, the linear constraint is formed by several half constraints within a single round, whereas nonlinear constraints are formed by one half constraint on x_i^r and another half constraint on y_i^r.

Linear constraint. Linear constraints do not involve the nonlinear function $S(\cdot)$ and are derived from several value restrictions within a single cipher round. As illustrated in Figure 1, the linear constraint is formed by two value-restricted variable $L^{-1}(x_i^r)$ and y_i^{r-1}. This implies that the corresponding key variable set k_i^{r-1} must satisfy the relation $k_i^{r-1} = y_i^{r-1} \oplus L^{-1}(x_i^r)$.

Nonlinear constraint. Nonlinear constraints arise from the combination of half constraints across multiple rounds. In this example, three half constraints impose value restrictions on x_i^r and y_i^r. Meanwhile, the nonlinear layer S enforces the condition $S(x_i^r) = y_i^r$, which implies that $S(L(y_i^{r-1} \oplus k_i^{r-1})) = L^{-1}(x_i^{r+1}) \oplus k_i^r$. Consequently, a constraint is established on k_i^r and k_i^{r-1}, as all non-key variables involved are restricted by half constraints. Another important feature of nonlinear constraints is that they involve constraints from several different rounds and therefore have an impact that propagates across multiple rounds.

Based on these possible constraints, the authors proposed an algorithm to automatically search for constraints within a differential trail. However, this method does not consider all possible relationships between all the variables, and thus, it is unable to capture some more complicated constraints.

3 The Cons-Collector Constraint Detection Framework

In this section, we introduce Cons-Collector, a framework for identifying all the constraints on key in a differential trail of a block cipher, giving an accurate analysis of the possible key space and probability distribution of the differential trails.

3.1 Notations

Our detection framework is general and applies to a wide range of block ciphers. This section specifies the notations we will use to describe it. We classically view an iterative block cipher as a succession of linear and nonlinear layers (denoted L and S respectively), which we will represent as a system of linear and nonlinear equations.

Formally, an R-round iterative block cipher is expressed as:

$$\begin{cases} Y^r & = S(X^r), \\ X^{r+1} & = L(Y^r, K^r),\ 0 \le r < R, \end{cases}$$

where X^r and Y^r denote the input and output of S in the r-th round respectively and K^r denotes the r-th round key (X^0 and X^R stand for the input and output of the cipher respectively).

Since the size of each internal state variable X^r, Y^r and K^r is usually quite large, directly exploring their relationships can be challenging. Therefore, we further decompose these large internal state variables into smaller cells when possible, facilitating a more tractable representation:

$$X^r = (x_0^r, x_1^r, \cdots, x_{m-1}^r), \quad Y^r = (y_0^r, y_1^r, \cdots, y_{m-1}^r), \quad K^r = (k_0^r, k_1^r, \cdots, k_{m-1}^r).$$

When the structure of S is composed of m functions independently operating on each x_i^r, y_i^r (by a slight abuse of notation we denote them S as well), and assuming that the key material is incorporated during the linear layer, this fine-grained representation will later reduce the complexity of solving the system of equations. The choice of m is usually very natural as in most cases it would typically be the number of parallel S-boxes in S. The original system now becomes (see Figure 2):

$$\mathcal{E} = \begin{cases} y_i^r & = S(x_i^r),\ 0 \le i < m \\ (x_0^{r+1}, \cdots, x_{m-1}^{r+1}) & = L(y_0^r, \cdots, y_{m-1}^r, k_0^r, \cdots, k_{m-1}^r),\ 0 \le r < R, \end{cases} \tag{1}$$

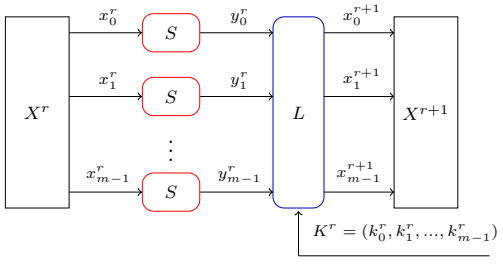


Figure 2: General notation of a round function

Note that this representation encompasses most SPN and Feistel ciphers. For word-wise ciphers, e.g., SKINNY-64, this representation is very natural: the 64-bit state is partitioned into 16 nibbles and all the operations in the round function are performed at the nibble level. For such ciphers, the computational complexity for the solving part of our framework will usually be lower. For bit-wise ciphers, e.g., GIFT-64 [BPP+17], the input cannot be

further partitioned into smaller variables due to the bit-level operations in the linear layer. If the input is divided into 16 cells (matching the size of the S-boxes), we will not be able to capture all the relationships imposed by the linear layer. Conversely, if the input is expressed at the bit level with 64 Boolean variables, the nonlinear layer becomes difficult to generalize in the form $y = S(x)$. Hence, the entire input would have to be treated as a single cell with size equal to 64, thus increasing the solving part complexity within our framework.

Regardless of this distinction, the efficiency of the solving part of the framework will also depend on the number of variables and their sizes, as well as on the algebraic structure of the cipher.

3.2 Propagation rules

The idea of `Cons-Collector` is to generate a collection of constraints on the values of the internal states and round keys, imposed by the differential trail, and then study the dependency relationships. We introduce here the propagation rules our framework will use on these value constraints. Before proceeding, we first present several necessary definitions.

Definition 6 (Free Variable)**.** In a block cipher, an n-bit variable z that is not a key variable, is a free variable if its probability distribution is uniform i.e., $P(z = z_i) = \frac{1}{2^n}$, with no constraints imposed on z.

Definition 7 (Constrained Variable)**.** An n-bit variable z is a constrained variable (denoted as $[z]$) if the probability distribution is affected by at least one constraint.

Definition 8 (Constraint Subset)**.** Given a differential trail, a constraint subset $\mathbb{E}$ is a subset of equations derived from the algebraic structure of a cipher, which impose constraints on key variables.

In a differential trail, the input and output variables of all active S-boxes are all considered constrained variables as they must satisfy the prescribed nonzero input and output differences. As these constraints propagate through both the linear and nonlinear equations, they gradually influence the other variables, which may or may not eventually lead to a reduction in the valid key space. We now list these propagation rules.

Propagation via linear equations. Each linear equation in $\mathcal{E}$ encapsulates multiple dependency relationships among the variables, offering different pathways for constraint propagation. Consider the linear equation: $x_1 \oplus y_1 \oplus k_1 = 0$, from which we can derive the following dependencies:

$$(x_1, y_1) \leftrightarrow k_1, \quad (x_1, k_1) \leftrightarrow y_1, \quad (y_1, k_1) \leftrightarrow x_1,$$

where $\leftrightarrow$ denotes a dependency relation. These dependencies form pathways or rules of how constraints can propagate through linear equations. For example, if $x_1 \oplus y_1$ is a constrained variable, the dependency allows the constraint to propagate to k_1. Conversely, if k_1 is constrained, it will propagates and introduce a constraint on the pair (x_1, y_1).

Propagation via nonlinear equations. For the nonlinear equations of the form $y = S(x)$, any constraint on x directly imposes a constraint on y since y is entirely determined by x, and vice versa. Therefore, the equation itself serves as a propagation pathway, denoted as:

$$y = S(x) \leftrightarrow x,$$

which indicates the direct dependency between x and $S(x)$. In our framework, $S(x)$ is treated as if it is x when we propagate through the constraints. One notable mention is

the distribution of $S(x) \oplus x$ for different ciphers. If we look at the probability distribution of this term for a single S-box for SKINNY, LBLOCK and TWINE, it is actually non-uniform. Therefore, even in the case where neither x or $S(x)$ are constrained, the term $x \oplus S(x)$ is actually constrained.

Propagation via summation of equations. When equations involve both constrained and free variables, analyzing them individually may not impose direct constraints on key variables. However, summing up these equations may eliminate free variables and potentially introduce constraints on key variables. For example, consider the following system of linear equations:

$$\begin{cases} [x_0] \oplus y_0 \oplus k_0 &= 0, \\ [x_1] \oplus y_0 \oplus k_1 &= 0. \end{cases}$$

Individually, the two equations above do not impose direct constraints on k_0 and k_1 as y_0 is a free variable. However, summing them eliminates y_0 and introduces a compound dependency, causing the following constraint:

$$[x_0] \oplus [x_1] \oplus k_0 \oplus k_1 = 0.$$

Given that x_0 and x_1 are constrained variables, the term $x_0 \oplus x_1$ may either remain constrained or become a free variable, depending on their specific values. This distinction directly influences the constraints imposed on $k_0 \oplus k_1$. For example, if $x_0 = x_1 \in \{1, 3, 8, 10\}$, it imposes an effective constraint on keys: $k_0 \oplus k_1 \in \{0, 2, 9, 11\}$. Conversely, if $x_0 \in \{1, 3, 8, 10\}$ and $x_1 \in \{2, 3, 6, 7\}$, then $x_0 \oplus x_1$ spans all possible values from 0 to 15, making it a free variable and imposing no restriction on $k_0 \oplus k_1$. The same technique can be applied to the case of nonlinear equations as well. Consider the following system:

$$y_1 = S(x_1), \quad y_1 \oplus k_1 = 0, \quad x_1 \oplus k_2 = 0.$$

Summing all the equations in the above system yields the following:

$$k_1 = y_1 = S(x_1) = S(k_2),$$

which establishes a constraint propagation pathway between k_1 and k_2 and restricts the pair (k_1, k_2) to a specific set of 2^ω possible outcomes, denoted as $(k_1, k_2) \in \{(S(i), i) \mid 0 \leq i < 2^\omega - 1\}$. This illustrates how constraints can arise even without any initial constrained variables.

In practice, only a small subset of dependency pathways effectively propagate constraints to key variables. Thus, instead of analyzing all dependencies in $\mathcal{E}$, it is sufficient to identify the specific subset of equations, referred to as constraint subsets, that directly impose restrictions on key variables. The challenge lies in systematically detecting all such constraint subsets and efficiently identifying all linear and nonlinear constraints on key variables.

3.3 Principles of Constraint Detection

Before proceeding, we first give the definition of a minimal constraint subset.

Definition 9 (Minimal Constraint Subset)**.** Given a differential trail, a minimal constraint subset $\mathbb{E}_{min}$ is a constraint subset which does not contain any smaller constraint subset inside.

Identifying minimal constraint subsets. In this work, `Cons-Collector` aims to detect all minimal constraint subsets within the differential trails. One of the main principles in identifying such constraints is to eliminate all free variables involved by summing up the equations. Since free variables in a constraint subset do not affect the distribution of the associated key variables, meaningful constraints can only be imposed by either propagating some value restrictions to these free variables or eliminating them through summation. After eliminating all free variables, the resulting equation will only consist of constrained variables (either the set of initial constrained variables or the non-uniform constrained variable $S(x) \oplus x$), thereby forming a restriction on key variables. However, whether the resulting summed equation imposes a constraint on the relevant key values still depends on the specific values of all the constrained variables. We call potential constraint subset such a system of equations whose summation can remove all free variables. To formalize this idea, we introduce the following corollary:

Corollary 1. *An equation subset, $\mathbb{E} = \{e_1, e_2, \cdots, e_v\}$, forms a potential minimal constraint subset on the relevant key variables $\{k_j\}$ if and only if*

$$\sum_{i=1}^{v} e_i = \sum_{j} k_j \oplus \sum_{u} (S(x_u) \oplus x_u) \oplus \sum_{h} v_h = 0, \tag{2}$$

while for any equation subset $\mathbb{E}' \subset \mathbb{E}$, Equation (2) does not hold. According to Equation (1), e_i either takes form of $y_u = S(x_u)$ or $x_u \oplus L(y_0, \cdots, y_{m-1}, k_0, \cdots, k_{m-1}) = 0$; S refers to the nonlinear function of the cipher; and v_h refers to constrained variables.

Proof. **Sufficiency (⇒):** Consider an equation set $\mathbb{E}$ derived from the algebraic structure of a cipher, which consists of nonlinear and linear equations e_i. Assume there is an equation set $\mathbb{E}$, such that Equation (2) does not hold for $\mathbb{E}$ and any other equation subset $\mathbb{E}' \subset \mathbb{E}$, and $\mathbb{E}$ can still be a minimal constraint on key variables $\{k_0, k_1, \cdots, k_J\}$, where nonlinear equations are all of the form: $y_u = S(x_u)$. Therefore, for $\mathbb{E}$ we have:

$$\sum_{i} e_i = \sum_{j} k_j \oplus \sum_{u} (S(x_u) \oplus x_m) \oplus \sum_{h} v_h \oplus F = 0,$$

Here, F is a free variable; otherwise, it is constrained and can be included in $\sum v_h$. Then we get:

$$\sum k_j = \sum_{u} (S(x_u) \oplus x_u) \oplus \sum_{h} v_h \oplus F = F'.$$

Moreover, F' is also a free variable because the XOR operation between a free variable F and any other variables always results in a free variable F'. Consequently, no constraints are imposed on the key, leading to a contradiction. Thus, the sufficiency is proven.
Necessity (⇐): We first prove the equation set $\mathbb{E}$ is a potential constraint on keys. Assume one equation set $\mathbb{E}$ satisfies Equation (2). Thereby, we have:

$$\sum k_j = \sum_{u} (S(x_u) \oplus x_u) \oplus \sum_{h} v_h,$$

all components on the right-hand side are constrained variables, as both $S(x_u) \oplus x_u$ and v_h are constrained. Consequently, this equation could potentially impose value restrictions on the key variables $\{k_0, k_1, \cdots, k_J\}$. Therefore, $\mathbb{E}$ is a constraint;

Then, we prove equation set $\mathbb{E}$ is a minimal constraint. As no other subset $\mathbb{E}' \subset \mathbb{E}$ satisfies Equation (2), which means no other subset inside $\mathbb{E}$ is a constraint on key. Therefore, $\mathbb{E}$ is the minimal constraint on key, and the necessity is proved. □

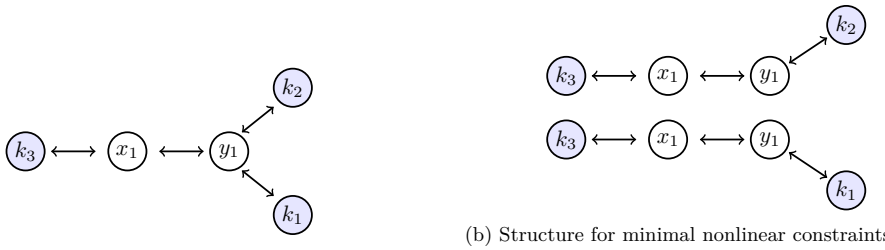
Notably, based on this corollary, Cons-Collector framework searches for all minimal nonlinear and linear constraint subsets within the algebraic structure. For example, consider the nonlinear constraint $\mathbb{E}_{exp}$ found in [PT22]:

$$x_1 \oplus k_3 = 0,\ y_1 \oplus k_1 = 0,$$
$$y_1 \oplus k_2 = 0,\ y_1 = S(x_1).$$

This nonlinear constraint is not a minimal constraint subset, as a smaller equation subset inside it, $\{x_1 \oplus k_3 = 0,\ y_1 \oplus k_1 = 0, y_1 = S(x_1)\}$, is also a constraint subset. Instead, Cons-Collector framework will detect two minimal nonlinear constraints, which are equivalent to $\mathbb{E}_{exp}$:

$$\mathbb{E}_{exp1} : x_1 \oplus k_3 = 0,\ y_1 \oplus k_1 = 0, y_1 = S(x_1);$$
$$\mathbb{E}_{exp2} : x_1 \oplus k_3 = 0,\ y_1 \oplus k_2 = 0, y_1 = S(x_1).$$

The comparison between minimal nonlinear constraints and the constraint $\mathbb{E}_{exp}$ is shown in Figure 3. Rather than searching for the entire tree-structure of nonlinear constraints, Cons-Collector identifies all minimal components within the tree structure, which can later be combined to construct larger constraints.



(a) Structure for nonlinear constraint $\mathbb{E}_{exp}$

(b) Structure for minimal nonlinear constraints $\mathbb{E}_{exp1}$ and $\mathbb{E}_{exp2}$

Figure 3: Comparison between minimal and common constraint subset.

Detecting minimal linear and nonlinear constraints enables the framework to systematically capture all possible constraints within the trail by transforming the problem into a search for linearly dependent vectors within a matrix.

Constructing the constraint subset. We introduce a binary matrix product representation for $\mathcal{E}$, which captures the constraint propagation pathways between variables. Using Gaussian elimination, we can identify linearly dependent equations efficiently.

As discussed in Section 3.2, the nonlinear equation $y = S(x)$ is regarded as a constraint propagation pathway and $x \oplus S(x)$ is a constrained variable. To express this feature, we simply denote $S(x) \leftrightarrow x$ as $x \oplus S(x) = 0$, which means the summation of $S(x)$ and x is a constrained variable. Therefore, following the notation in Section 3.1, the m nonlinear equations in Equation (1) can be expressed in the following matrix product form:

$$\underbrace{\begin{pmatrix} a_0^r & 0 & \cdots & 0 \\ 0 & a_1^r & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{m-1}^r \end{pmatrix}}_{A^r} \underbrace{\begin{pmatrix} a_0^r & 0 & \cdots & 0 \\ 0 & a_1^r & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{m-1}^r \end{pmatrix}}_{A^r} \cdot \left(X^r, Y^r\right)^T = 0,$$

where $a_i^r \in \{0,1\}, 0 \le i < m$. The matrix $(A^r\ A^r)$ captures the dependency relationships between variables in the constraint propagation pathways $x_i^r \leftrightarrow y_i^r$. According to Corollary 1, constrained variables do not need to be eliminated, so their coefficients in the

matrix are set to 0. Specifically, $a_i^r = 0$ if and only if x_i^r or y_i^r is a constrained variable (e.g., the corresponding S-box is active), and $a_i^r = 1$ otherwise.

For the m linear equations in Equation (1), each can be expressed as: $c_i^r \cdot x_i^{r+1} = (\bigoplus_{j=0}^{m-1} b_{i,j}^r \cdot y_j^r) \oplus (\bigoplus_{j=0}^{m-1} d_{i,j}^r \cdot k_j^r)$, where $0 \leq i < m$. These equations can be rewritten in the following matrix form:

$$\underbrace{\begin{pmatrix} b_{0,0}^r & \cdots & b_{0,m-1}^r \\ b_{1,0}^r & \cdots & b_{1,m-1}^r \\ \vdots & \ddots & \vdots \\ b_{m-1,0}^r & \cdots & b_{m-1,m-1}^r \end{pmatrix}}_{B^r} \underbrace{\begin{pmatrix} c_0^r & 0 & \cdots & 0 \\ 0 & c_1^r & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{m-1}^r \end{pmatrix}}_{C^r} \underbrace{\begin{pmatrix} d_{0,0}^r & \cdots & d_{0,m-1}^r \\ d_{1,0}^r & \cdots & d_{1,m-1}^r \\ \vdots & \ddots & \vdots \\ d_{m-1,0}^r & \cdots & d_{m-1,m-1}^r \end{pmatrix}}_{D^r} \cdot \left(Y^r, X^{r+1}, K^r\right)^T = 0,$$

where the parameters $b_{i,j}^r, c_i^r, d_{i,j}^r \in \{0,1\}$, are determined based on the specific linear function, $0 \leq i,j < m$. Similarly, if a variable is a constrained variable, its corresponding coefficient is set to 0. Finally, the complete algebraic system in Equation (1) can be represented in matrix product form as:

$$\underbrace{\begin{pmatrix} A^0 & A^0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & B^0 & C^0 & 0 & 0 & \cdots & 0 & 0 & D^0 \\ 0 & 0 & A^1 & A^1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & B^{R-1} & C^{R-1} & D^{R-1} \end{pmatrix}}_{M} \cdot \left(X^0, Y^0, X^1, K^0 \cdots, Y^{R-1}, X^R, K^{R-1}\right)^T = 0,$$

where the coefficient matrix, denoted as M, represents the constraint propagation pathways of all linear and nonlinear functions.

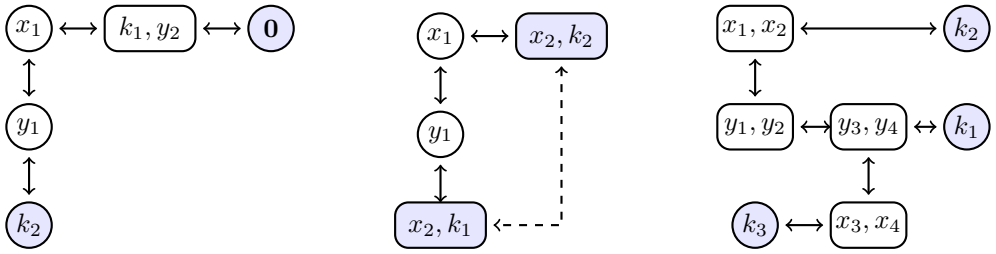
Identifying constraint subsets that satisfy Corollary 1 is equivalent to searching for linearly dependent vector subsets within the matrix M with all coefficients of keys: $D^r = \vec{0}$, as `Cons-Collector` searches for constraints on K^r and only focuses on the relation between x and y variables. Therefore, Gaussian eliminationcan be applied to efficiently detect these linearly dependent subsets. The Gaussian elimination-based detection algorithm of `Cons-Collector`, presented in Algorithm 1 in Appendix A, enables the automatic detection and retrieval of all potential constraint subsets.

To summarize, by applying Corollary 1 and Algorithm 1, `Cons-Collector` systematically identifies potential linear and nonlinear constraints within the equation system $\mathcal{E}$. This process reveals key-variable dependencies by eliminating free variables and utilizing constraint propagation pathways. As a result, `Cons-Collector` effectively detects all potential constraints on key variables.

3.4 Uncovering new dependencies

Previous research [PT22] has extensively studied linear constraints, which can be efficiently detected in our framework `Cons-Collector`. In contrast, detecting nonlinear constraints is significantly more challenging due to the complex interactions introduced by nonlinear layers, which cannot be directly represented like linear constraints. Our proposed detection framework leverages constraint propagation pathways to systematically capture dependencies imposed by nonlinear layers, uncovering previously unidentified constraints on key and addressing gaps left by prior research. To demonstrate the effectiveness of our framework, we present three examples illustrating some undiscovered dependencies.

Example 1. Consider the following system of equations involving the free variables x_1,



(a) Dependencies in Example 1 (b) Dependencies in Example 2 (c) Dependencies in Example 3

Figure 4: An illustration of the propagation pathways that were undiscovered.

y_1, y_2, and key variables k_1 and k_2:

$$\begin{cases} y_1 & = S(x_1), \\ y_1 \oplus k_2 & = 0, \\ y_2 \oplus k_1 & = 0, \\ y_2 \oplus x_1 \oplus k_1 & = 0. \end{cases}$$

Note that in this example, all of the non-key variables are free and the only S-box involved is inactive. However, based on this system of equation, one can still conclusively determine that $k_2 = S(0)$. The propagation pathway of constraints is shown in Figure 4a. The constraint propagates from knowing the value of x_1 to y_1 which ultimately imposes a restriction on k_2.

Example 2. Consider the following system of equations involving the free variables x_1, y_1, x_2, and key variables k_1 and k_2:

$$\begin{cases} y_1 & = S(x_1), \\ y_1 \oplus x_2 \oplus k_1 & = 0, \\ x_1 \oplus x_2 \oplus k_2 & = 0. \end{cases}$$

Again, one can derive the relationship between k_1 and k_2: $k_1 = S(x_2 \oplus k_2) \oplus x_2$ by simplifying the system. This allows us to analyze the non-uniform probability distribution of the key pair (k_1, k_2). The propagation pathway is illustrated in Figure 4b.

This example demonstrates how the non-uniformity of $S(x) \oplus x$ can introduce effective constraints on key variables, even when all input variables are free. In our framework, `Cons-Collector` managed to identify this type of pathway as a constraint, as it imposes a potential constraint on the pair (k_1, k_2).

Example 3. Consider the following equation set, where all variables are free variables:

$$\begin{cases} y_i & = S(x_i), \ (0 < i \leq 4), \\ y_1 \oplus y_2 \oplus x_3 \oplus x_4 \oplus k_1 & = 0, \\ x_1 \oplus x_2 \oplus k_2 & = 0, \\ y_3 \oplus y_4 \oplus k_3 & = 0. \end{cases}$$

In the above equations, if $k_2 = 0$, then $y_1 \oplus y_2 = S(x_1) \oplus S(x_2) = 0$. Consequently, the second equation can be simplified as: $x_3 \oplus x_4 \oplus k_1 = 0$. By setting $k_1 = 0$, it follows that $x_3 \oplus x_4 = 0$, which leads to $y_3 \oplus y_4 = 0 = k_3$. As a result, when $k_2 = 0$ and $k_1 = 0$, it also determines that $k_3 = 0$. Therefore, this system of equations imposes a nonlinear constraint on key variables (k_1, k_2, k_3). This example demonstrates again that, even if all variables

are free variables, as long as they are linked to each other within the same equation set, the equations can still impose potential constraints on key. The propagation pathway is illustrated in Figure 4c. If certain restrictions are placed on k_1 and k_2, the constraints propagate from (x_1, x_2) to (y_1, y_2), then to (y_3, y_4), subsequently reaching (x_3, x_4), and ultimately imposing a constraint on k_3.

4 `Cons-Solver`: An Automated Predictor for Probability Distribution

This section introduces `Cons-Solver`, an automated solving tool that parses the constraint subsets identified by `Cons-Collector` and computes a probability distribution that is close to the true distribution for the differential trail.

4.1 Probability Computation

Typically, `Cons-Collector` detects multiple subsets $\mathbb{E}_0, \mathbb{E}_1, \cdots, \mathbb{E}_v$ within a differential trail. However, these subsets are not necessarily independent. Thus, we would have to consider all of them at once. We will denote that solution space of the equations as $\mathbb{S} = Sol(\mathbb{E}_0 \cup \mathbb{E}_1 \cdots \cup \mathbb{E}_v)$. Next, we will introduce two methods to calculate the probability distribution of differential trails.

An accurate method for probability calculation. When the number of variables and/or equations is not too large, we can enumerate all the possible solutions for $\mathbb{E}_0 \cup \mathbb{E}_1 \cdots \cup \mathbb{E}_v$ and compute the frequency of the relevant key variables, $F_{\vec{k}}$ as well as the solution space $\mathbb{S}$. Finally, the probability of the differential trail T under a specific key assignment $\vec{k}$ can be computed as:

$$P_{T|\vec{k}} = P_0 \cdot \frac{F_{\vec{k}}}{|\mathbb{S}|} \cdot 2^{|k|},$$

where $2^{|k|}$ denotes the size of the involved key k and P_0 is the original estimated probability of T under the Markov assumption.

An estimate method for probability calculation. When the number of equations in $\mathbb{E}_0 \cup \mathbb{E}_1 \cdots \cup \mathbb{E}_v$ is too large, computing the exact solution set is impractical due to excessive time and memory requirements. Therefore, in this case, we can turn to estimation via sampling. We randomly generate a set of samples for the involved variables. More specifically, we generate 2^{m_1} values for the variables x (with corresponding $y = S(x)$ values determined accordingly) and 2^{m_2} values for the relevant keys. This results in a total of $2^{m_1+m_2}$ samples. In our experiments, we set $m_1 = 34$ and $m_2 = 10$ to balance computational efficiency and statistical accuracy.

Similar to the computation above, we can estimate the probability distribution of T under a given key assignment $\vec{k}$ by estimating the frequency $F'_{\vec{k}}$ and the solution space $\mathbb{S}'$:

$$P_{T|\vec{k}} = P_0 \cdot \frac{F'_{\vec{k}}}{|\mathbb{S}'|} \cdot 2^{|k|},$$

However, as the size of the constraint subset increases, this sampling method will still be inefficient if we want to obtain an accurate profile. Thus, to enhance its efficiency, we incorporated a pre-processing phase.

4.2 Pre-processing the Constraint Subsets

This section introduces three preprocessing techniques to optimize the processing of identified constraint subsets, reducing the computational complexity of solving equations.

Merging equations and eliminating redundant variables. For linear equations within the same linear layer, some contain free variables that do not appear in any nonlinear equations. These free variables can be eliminated by summing up the corresponding linear equations and generating a new linear equation that does not contain these irrelevant variables without altering the constraints that affect the key. For example, consider the constraint subset $\mathbb{E}_3$ in Table 5. The equations $y_1^0 \oplus x_5^1 \oplus k_1 = 0$ and $y_1^0 \oplus [y_{11}^0] \oplus [x_{13}^1] \oplus k_1 = 0$ belong to the same linear layer in the first round, and y_1^0 is a completely free variable, independent of any other variables in $\mathbb{E}_3$. Therefore, summing these two equations produces a simplified equation $x_5^1 \oplus [y_{11}^0] \oplus [x_{13}^1] = 0$, which eliminates y_1^0 and k_1. However, if $y_1^0 = S(x_1^0)$ is also included in the constraint $\mathbb{E}_3$, these two linear equations cannot be combined. This is because the nonlinear relationship between x_1^0 and y_1^0 could propagate the dependencies, influencing the probability distribution of k_1. The constraint subsets $\mathbb{E}_1$ and $\mathbb{E}_2$ in Table 5 do not contain such free variables that can be eliminated and therefore remain unchanged. This process reduces system complexity by eliminating redundant variables that do not introduce new information.

Grouping subsets and eliminating duplicate equations. We also applied a grouping strategy to speed up the solving process. The subsets involving the same key variables are merged into a single set, and duplicated equations are eliminated. Meanwhile, equations that are independent of each other are handled separately. For example, the constraint subsets $\mathbb{E}_1$ and $\mathbb{E}_2$ in Table 5 are closely related with each other, thus, they are combined into a single set. In contrast, the equations in $\mathbb{E}_3$ remain independent of those in $\mathbb{E}_1$ and $\mathbb{E}_2$. Therefore, `Cons-Solver` organizes the three constraint sets into two groups: $\mathbb{E}_1 \cup \mathbb{E}_2$ and $\mathbb{E}_3$. The global solution set is then obtained by independently solving the equations in each group and combining their solutions, leading to $|Sol(\mathbb{E}_1 \cup \mathbb{E}_2 \cup \mathbb{E}_3)| = |Sol(\mathbb{E}_1 \cup \mathbb{E}_2)| \times |Sol(\mathbb{E}_3)|$. This grouping improves efficiency by reducing the complexity of solving a large system, allowing each subset to be processed separately.

Merging multiple constrained variables into a single variable. For each equation, we merge the constrained variables into a new auxiliary variable and analyze if remains constrained. If this auxiliary variable is still constrained, the constraint set remains effective. However, if it becomes a free variable, the entire set will not impose restrictions on key variables and, therefore, has no impact on the probability distribution of the differential trail. By doing so, we can reduce the number of variables. For example, in the equations of $\mathbb{E}_1$ in Table 5, the constrained variables y_4^0 and y_{11}^0 are combined as $y_4^0 \oplus y_{11}^0$. It is possible that $y_4^0 \oplus y_{11}^0$ becomes a free variable, even though both y_4^0 and y_{11}^0 are constrained variables, making the entire constraint ineffective. This process eliminates redundant constraints, ensuring that only meaningful constraints are retained. If it remains a constrained variable, the constraint is simplified by reducing the number of involved variables.

5 Experimental Results

In this section, we used `Trail-Estimator` to analyze differential trails of `SKINNY-64`, `SKINNY-128`, `TWINE`, and `LBLOCK`. For each targeted differential trail, `Cons-Collector` detected all linear and nonlinear constraints, while `Cons-Solver` computed the probability distribution. The detailed number of constraints identified and a comparison to

previous works are summarized in Table 1, highlighting that our tool identifies more nonlinear constraints than previous works. The probability distributions obtained from the constraints are presented in Table 2. To validate the accuracy of `Trail-Estimator`, we conducted experiments to obtain the experimental probability distribution of some short-round differential trails to compare with `Trail-Estimator`'s predictions.

5.1 Application to SKINNY

`SKINNY` is a lightweight block cipher family introduced by Beierle et al. [BJK+16], featuring a tweakable SPN-based structure. We applied `Trail-Estimator` to a total of 11 differential trails from the `SKINNY` family of ciphers. We obtained these from various sources [ZZ18, DDH+21, PT22, QDW+21]. For every single trail, `Trail-Estimator` detected more nonlinear constraints than all the previous works and managed to obtain a more precise estimation of the probability distribution. The detailed probability distributions derived from these constraints are presented in Table 3.

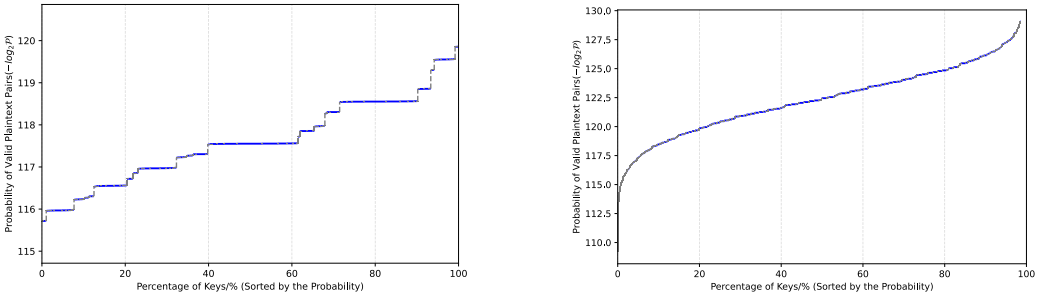
Table 3: Experimental results for `SKINNY`

Version	Rds	Reduced key space	Stated prob.	Probability Distribution Percentage of prob.					Source
64-64	7	$2^{-7.3}$	52	49-47.01 9.76%	47-46.01 31.38%	46-45.01 36.46%	45-44.01 18.18%	44-42.42 4.12%	Table 6 [DDH+21]
	10	0	46			—			Table 7 [DDH+21]
	10	1	42			42 100%			Table 7 [PT22]
64-128	13	2^{-4}	55			51 100%			Table 8 [DDH+21]
	8	1	17-19			15 or 16 100%			Table 16 [QDW+21]
64-192	15	$2^{-6.2}$	54		48 85.71%		47 14.28%		Table 9 [DDH+21]
	11	0	147			—			Figure 2 [ZZ18]
128-128	14	$2^{-10.4}$	120	119.9-118.9 6.66%	118.8-117.8 27.92%	117.7-116.7 42.29%	116.6-115.71 23.12%		Table 10 [DDH+21]
128-256	16	$2^{-11.1}$	127.6	128.2-125.2 9.30%	125.1-122.1 31.30%	122.0-117.0 35.67%	116.9-113.9 17.95%	113.8-108.2 5.78%	Table 11 [DDH+21]
128-384	17	0	110			—			Table 12 [DDH+21]

Stated Prob.: the $-log_2$ probability reported in the original papers; Reduced key Space: refers to the proportion of the estimated valid key space for the differential trail.

Results summary. These trails have been analyzed before in previous work, so we will use the rest of this subsection to highlight the difference with previous works. For instance, in the case of the differential trail from Table 10 of [DDH+21], `Trail-Estimator` identified six solvable nonlinear constraints, including one new nonlinear constraint, as detailed in Appendix H.5. In the case of Table 11 of [DDH+21], we found two new solvable nonlinear constraints (see Appendix H.6). The predicted probability distribution of the differential trails can be found in Figure 5. We observed that as the complexity of the constraints increases, the resulting probability distribution approaches a Gaussian-like distribution. Additionally, for `SKINNY-64` `TK1`, `TK2`, and `TK3` differential trails, we incorporated key scheduling into the solving phase. Due to the substantial size of the nonlinear constraints, only one remains solvable within 15-round `SKINNY-64-192` trail. For `SKINNY-64` `TK2` and `TK3`, our results are consistent with those reported in [PT22]; however, we identify additional nonlinear constraints by considering the `TK2` scheduling. For 16-round `SKINNY-128 -256`

trail [DDH+21], when we incorporated key scheduling into solving, the complexity increase drastically and only linear constraints and three nonlinear constraints remain solvable in this case. For the infeasible SKINNY -128-384 trail, we considered TK3 scheduling and provided all detected constraints on three tweakeys. Based on solvable constraints, Trail-Estimator provided a comprehensive prediction of the probability distribution for SKINNY-128 differential trails of Table 10 and Table 11 in [DDH+21] for the first time, as shown in Figure 5.



(a) 14-round SKINNY-128 trail [DDH+21] (b) 16-round SKINNY-128 trail [DDH+21]

Figure 5: Predicted probability distribution of SKINNY-128 differential trails

Comparison with previous works. We compare our results for SKINNY with previous studies and find that Trail-Estimator consistently identifies more nonlinear constraints. In particular, Trail-Estimator discovered a significant number of nonlinear constraints that span over five or more rounds within a differential trail. Some of these extended nonlinear constraints span up to 14 rounds, which have not reported before. As discussed in Section 3.2, detecting all constraints requires considering all possible constraint propagation pathways between variables. However, previous works more or less overlooked some relationship between variables. The detection framework in [PT22] detects half-constraints, which represent only a small portion of the relationships between constrained variables and free variables.

The work in [Sun24] focuses on the linear relationships between the input and output bits of S-boxes. For the 3rd-5th rounds of the 11-round differential trail in [ZZ18], it identifies one linearized nonlinear constraints, which captures only certain linear relationships between specific bits while overlooking nonlinear dependencies in the trail. In contrast, Trail-Estimator identifies 2 different nonlinear constraints, as shown in Appendix H. Additionally, in SKINNY-128, some XDDT and YDDT of active cells do not form affine subspaces, making them inexpressible through linearized equations and thus ignored in [Sun24]. In [NGJE25], an SAT-based tool is introduced that utilizes SAT solvers and model counting to estimate the average probability of differential trails and the size of valid key spaces. In their case, they identified the 17-round SKINNY-128 trail from Table 12 of [DDH+21] as feasible but we managed to identify a constraint subset $\mathbb{E}_8$(details are in Appendix H.7) which rendered this differential trail infeasible.

Results verification. We extracted the first three rounds of the differential trail from Table 7 of [DDH+21] and denoted it as T_{S0}. We chose this particular segment to verify as it contains a short nonlinear constraint that was not detected by previous work. This extra constraint is shown in Appendix H.2. The probability for this differential trail under the Markov cipher assumption is 2^{-32}. A step-by-step illustration of how Cons-Collector finds the constraint subsets of T_{S0} is given in Appendix D. In Figure 6, we show the graph of the probability distribution derived from experimentation (in red) and compare it with

the predicted distribution in a Monte Carlo simulation. For the experiment, we generated a total of 1000 random master keys and for each key, we generated 2^{38} plaintext pairs. We note that there is a slight discrepancy between the two distribution, roughly at key number 670. We attribute this discrepancy to the small sample size of master keys.

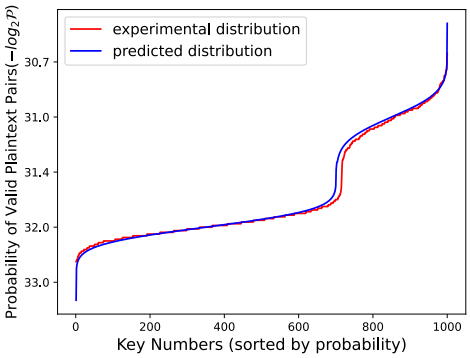


Figure 6: Predicted (blue) and experimental (red) probability distribution of the `SKINNY-64` trail T_{S0}

5.2 Application to `LBLOCK`

`LBLOCK` is a lightweight block cipher that has the Feistel structure, proposed by Wu and Zhang [WZ11]. We identified and applied `Trail-Estimator` to the optimal 16-round differential trail of `LBLOCK` from Zhou's paper [ZZDX19], denoted as T_{LBLOCK}.

For the 16-round differential trail T_{LBLOCK}, `Trail-Estimator` finds 8 linear and 11 nonlinear constraints and computes the predicted probability distribution. A graph of the predicted probability distribution is shown in Figure 7. We remark that the distribution closely follows a Gaussian distribution, with an average probability of 2^{-71} for the valid key space, which is now reduced by half, instead of the original probability of 2^{-72} that was stated in [ZZDX19], due to the effect of a linear constraint.

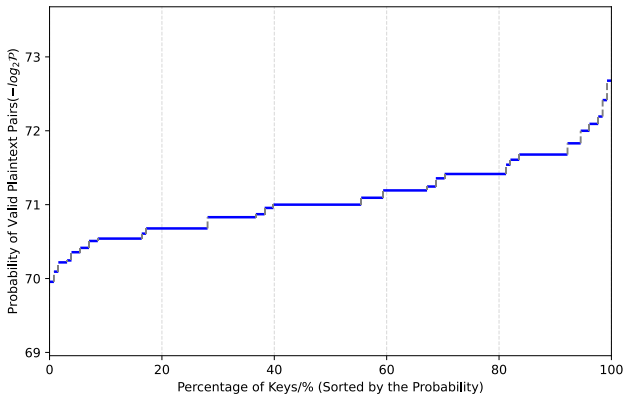
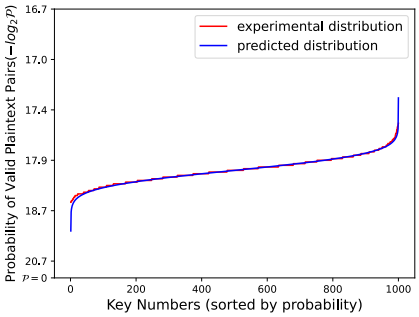


Figure 7: Predicted probability distribution of 16-round `LBLOCK` trail from [ZZDX19]

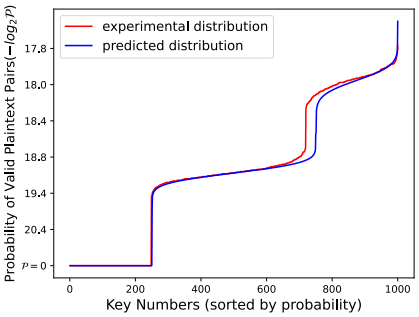
To validate the accuracy of probability predictions, both `Trail-Estimator` estimations and experimental probability distribution experiments are conducted on two 4-round differential trails, as shown in Figure 8. The first trail, denoted as T_{L0}, is extracted from the first four rounds of the optimal 16-round trail in [ZZDX19]. The second trail, denoted as T_{L1}, is derived by modifying the output difference of the 6th nibble of the final round

in T_{L0}. The details of two trails are presented in Table 9 and Table 10 in Appendix F, respectively. For T_{L0} and T_{L1}, `Cons-Collector` identifies the same nonlinear constraint, as detailed in Appendix I.1. Since the active S-box patterns remain identical in both trails, their nonlinear constraint structures are also the same, potentially influencing the probability distributions of T_{L0} and T_{L1}. However, due to the distinct output difference, the constraints on the variable values differ accordingly. Consequently, the predicted probability distributions of the two differential trails may be different.

For T_{L0}, the calculation of `Cons-Solver` shows that the nonlinear constraint does not affect the original probability distribution. According to Figure 8a, the experimental probability distribution of the trail aligns with our estimated results, following a Gaussian distribution with an average probability of approximately 2^{-18}. For T_{L1}, although it differs from T_{L0} by only a single nibble, the constraint has a significantly different impact on its probability distribution. According to `Cons-Solver`, T_{L1} has an average probability of 2^{-19} for 50% of the keys, 2^{-18} for 25% of the keys, while the remaining 25% of the keys are infeasible. As shown in Figure 8b, the predicted probability distribution from `Trail-Estimator` closely aligns with the experimental probability distribution, except for a slight discrepancy around key number 700, which is attributed to the effects of key scheduling. As `Trail-Estimator` assumes that all round keys are mutually independent, it might introduce some imprecisions. To validate this, we analyze `LBLOCK` with fully independent round keys as well. As shown in Figure 11, the predicted and actual probability distributions align closely, with only a gap less than 2% between them.



(a) Probability distribution of T_{L0}



(b) Probability distribution of T_{L1}

Figure 8: Predicted (blue) and experimental (red) probability distribution of 4-round trails T_{L0} and T_{L1}

In summary, for the differential trails of `LBLOCK`, our method not only identifies all constraints on the key space but also provides the probability predictions closely aligning with experimental distributions. Additionally, based on the results from T_{L0} and T_{L1}, we observe that nonlinear constraints with same structure could potentially influence the probability distribution of trails differently.

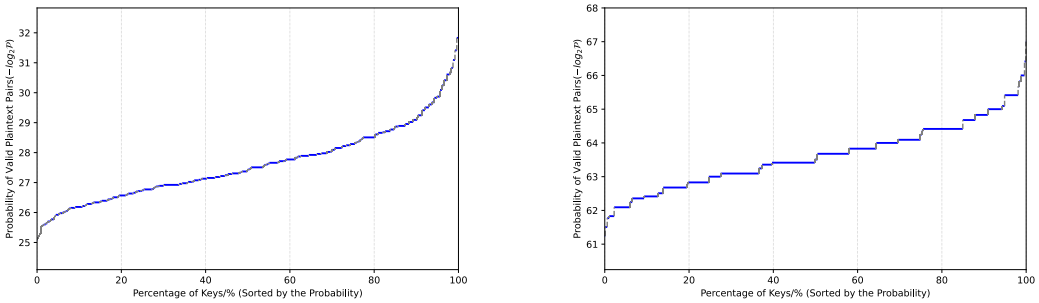
5.3 Application to `TWINE`

`TWINE` is a lightweight block cipher proposed by Suzaki et al. [SMMK12], based on a Feistel structure. In the experiment, `Trail-Estimator` is applied to two differential trails of `TWINE`: an optimal 9-round trail discovered using the Mixed Integer Linear Programming (MILP) solver (detailed in Table 12 in Appendix G) and an optimal 15-round differential trail from Table 16 [ZZDX19]. This study is the first to perform differential trail verification for `TWINE`.

For the 9-round differential trail in Table 12, denoted as T_{TW_1}, `Trail-Estimator` identified three nonlinear constraints but no linear constraint, as shown in Table 22, in

Appendix J. All three nonlinear constraints are within a solvable range. The predicted probability distribution for the trail is shown in Figure 9a. Under the Markov cipher assumption, this trail has a probability of 2^{-28}. Based on our predictions, the average probability of the trail within valid key space is $2^{-27.5\%}$, with a maximum of $2^{-25.14}$ and the minimum is $2^{-31.85}$. Further, according to `Cons-Solver`, over 22% keys are invalid. The overall effect of three nonlinear constraints is highly complex, and the final probability distribution follows closely to the Gaussian distribution.

For the 15-round differential trail ([ZZDX19]), denoted as T_{TW_2}, `Trail-Estimator` identified 10 nonlinear constraints and 6 linear constraints. However, only 4 of the nonlinear constraints are solvable [1], as shown in Table 23, Appendix J. Based on these 4 nonlinear equations and all linear equations, the predicted probability distribution is shown in Figure 9b. The 4 nonlinear constraints in T_{TW_2} are simpler than those in T_{TW1}, resulting in the probability distribution remains Gaussian-like but with greater discontinuities in the curve. Additionally, based on our probability distribution, the average probability for valid key space is 2^{-64} , which is 16 times higher than the probability stated in [ZZDX19]. This discrepancy is attributed to 4 linear constraints, each reducing the key space by half. As a result, only less than 6% of keys are valid for trail T_{TW_2}.



(a) Probability distribution of T_{TW1} (b) Probability distribution of T_{TW_2}

Figure 9: The predicted probability distributions of two `TWINE` differential trails by `Trail-Estimator`

6 Discussion and Conclusion

This paper introduced `Trail-Estimator`, an automated verification tool comprising the `Cons-Collector` constraints detection framework and the `Cons-Solver` solving tool. `Cons-Collector` systematically identifies all linear and nonlinear constraints on the key material, while `Cons-Solver` solves these constraints to estimate the probability distribution of differential trails.

`Trail-Estimator` was applied to analyze various differential trails of `SKINNY-64`, `SKINNY-128`, `LBLOCK`, and `TWINE` block ciphers, achieving state-of-the-art results. It effectively found all linear and nonlinear constraints and computed the overall probability distribution based on solvable constraints, with its estimations closely matching actual measured distributions. Based on our experimental results, we provide the following insights.

1. Certain linear constraints cause the valid subkey space to form an affine subspace, leading to an increase in the average probability of the trail within valid key space. On the other hand, nonlinear constraints introduce greater complexity into the key distribution, generally altering the probability distribution of the trail. Additionally,

[1] We define 2^{44} as the maximum alloted time complexity to be considered as "solvable".

as the number and complexity of constraints increase, the probability distribution progressively resembles a Gaussian distribution.

2. Distinct differential trails with the same active S-box patterns can exhibit significantly different probability distributions due to variations in difference values, which alter variable ranges and impact probabilities, even with the same constraint structures.

3. For word-wise key schedule algorithms, incorporating them into our framework ensures that probability predictions closely match actual values. Otherwise, we observed that assuming independent round keys leads anyway to a tolerable error compared to real values.

The advantage of `Trail-Estimator` compared to previous methods lies in its ability to identify more nonlinear constraints, allowing for a more comprehensive estimation of the probability distribution for differential trails. However, while `Trail-Estimator` is capable of identifying all potential constraints within differential trails, it can only solve the constraints within a solvable range. Its applicability declines when the input of the cipher cannot be further divided into smaller cells (e.g., `GIFT-64`). In future work, we aim to develop a more generic detection framework that can be applied to a wider range of block ciphers. Additionally, we plan to enhance the calculation efficiency of `Cons-Solver` by integrating new algorithms.

Acknowledgement

The 1st, 3rd and 4th authors are supported by the Singapore NRF Investigatorship grant NRF-NRFI08-2022-0013.

References

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 321–345. Springer, 2017.

[BR22] Tim Beyne and Vincent Rijmen. Differential Cryptanalysis in the Fixed-Key Model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 687–716. Springer, 2022.

[BS90] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in*

Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[BW99] Alex Biryukov and David A. Wagner. Slide Attacks. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999.

[DDH+21] Stéphanie Delaune, Patrick Derbez, Paul Huynh, Marine Minier, Victor Mollimard, and Charles Prud'homme. Efficient Methods to Search for Best Differential Characteristics on SKINNY. In *Applied Cryptography and Network Security - 19th International Conference, ACNS 2021, Kamakura, Japan, June 21-24, 2021, Proceedings, Part II*, volume 12727 of *Lecture Notes in Computer Science*, pages 184–207. Springer, 2021.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 393–410. Springer, 2010.

[DR07] Joan Daemen and Vincent Rijmen. Plateau characteristics. *IET Information Security*, 1(1):11–18, 2007.

[KM04] Lars R. Knudsen and John Erik Mathiassen. On the Role of Key Schedules in Attacks on Iterated Ciphers. In Pierangela Samarati, Peter Y. A. Ryan, Dieter Gollmann, and Refik Molva, editors, *Computer Security - ESORICS 2004, 9th European Symposium on Research Computer Security, Sophia Antipolis, France, September 13-15, 2004, Proceedings*, volume 3193 of *Lecture Notes in Computer Science*, pages 322–334. Springer, 2004.

[Leu12] Gaëtan Leurent. Analysis of Differential Attacks in ARX Constructions. volume 7658 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2012.

[LIM20] Fukang Liu, Takanori Isobe, and Willi Meier. Automatic Verification of Differential Characteristics: Application to Reduced Gimli. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 219–248. Springer, 2020.

[LMM91] Xuejia Lai, James L. Massey, and Sean Murphy. Markov Ciphers and Differential Cryptanalysis. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer, 1991.

[MR02] Sean Murphy and Matthew J. B. Robshaw. Key-Dependent S-Boxes and Differential Cryptanalysis. *Des. Codes Cryptogr.*, 27(3):229–255, 2002.

[NGJE25] Marcel Nageler, Shibam Ghosh, Marlene Jüttler, and Maria Eichlseder. AutoDiVer: Automatically Verifying Differential Characteristics and Learning Key Conditions. Cryptology ePrint Archive, Paper 2025/185, 2025.

[PT22] Thomas Peyrin and Quan Quan Tan. Mind Your Path: On (Key) Dependencies in Differential Characteristics. *IACR Trans. Symmetric Cryptol.*, 2022(4):179–207, 2022.

[QDW+21] Lingyue Qin, Xiaoyang Dong, Xiaoyun Wang, Keting Jia, and Yunwen Liu. Automated Search Oriented to Key Recovery on Ciphers with Linear Key Schedule Applications to Boomerangs in SKINNY and ForkSkinny. *IACR Trans. Symmetric Cryptol.*, 2021(2):249–291, 2021.

[SMMK12] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE: A Lightweight Block Cipher for Multiple Platforms. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012.

[Sun24] Ling Sun. A Linearisation Method for Identifying Dependencies in Differential Characteristics: Examining the Intersection of Deterministic Linear Relations and Nonlinear Constraints. Cryptology ePrint Archive, Paper 2024/1849, 2024.

[SWW18] Ling Sun, Wei Wang, and Meiqin Wang. More Accurate Differential Properties of LED64 and Midori64. *IACR Trans. Symmetric Cryptol.*, 2018(3):93–123, 2018.

[WZ11] Wenling Wu and Lei Zhang. LBlock: A Lightweight Block Cipher. In Javier López and Gene Tsudik, editors, *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, volume 6715 of *Lecture Notes in Computer Science*, pages 327–344, 2011.

[ZZ18] Pei Zhang and Wenying Zhang. Differential Cryptanalysis on Block Cipher Skinny with MILP Program. *Secur. Commun. Networks*, 2018:3780407:1–3780407:11, 2018.

[ZZDX19] Chunning Zhou, Wentao Zhang, Tianyou Ding, and Zejun Xiang. Improving the MILP-based Security Evaluation Algorithm against Differential/Linear Cryptanalysis Using A Divide-and-Conquer Approach. *IACR Trans. Symmetric Cryptol.*, 2019(4):438–469, 2019.

A Gaussian Elimination-Based Algorithm for Constraint Detection

Algorithm 1 Gauss-Collector algorithm

Input:

A $M \times N$ binary matrix MAT used to represent all linear equations in a block cipher;

Output:

A list which include the index of linear equations in the original linear matrix MAT;

Initialize M=MAT.size(); //row number of MAT
Initialize ROW_IND_MAT[M]=$\emptyset$;
Initialize cur_row=1;
while $cur\_row < M$ **do**
    while $findPivot(MAT, cur\_row) <= findPivot(MAT, cur\_row - 1)$ **do**:
        $MAT[cur\_row] \oplus = MAT[cur\_row - 1]$
        $ROW\_IND\_MAT[cur\_row] \oplus = ROW\_IND\_MAT[cur\_row - 1]$
        $REF(MAT, ROW\_IND\_MAT, cur\_row, N)$
    end while
    cur_row++;
end while
return ROW_IND_MAT;

B The Non-Uniformity of Variable $S(x) \oplus x$

In the following table, the uniformity of $S(x) \oplus x$ in different ciphers are shown. If $S(x) \oplus x$ is uniform, then $P(S(x) \oplus x = c) = \frac{1}{2^\omega}$ for any $0 \leq c < 2^\omega$, where ω is the size of variable x. However, for `SKINNY`, `LBLOCK` and `TWINE` ciphers, $\#(S(x) \oplus x) < 2^\omega$ always holds, which means $P(S(x) \oplus x = c) = 0$ for some c, thus $S(x) \oplus x$ is non-uniform and a constrained variable according to Definition 7.

Table 4: Non-Uniformity of $S(x) \oplus x$ $(0 \leq x \leq 2^\omega - 1)$ for ω-bit S-boxes of various ciphers

Cipher	$\#(S(x) \oplus x)$	**Value of** 2^ω	**Uniformity of** $S(x) \oplus x$
`SKINNY-64`	12	16	Non-Uniform
`SKINNY-128`	178	256	Non-Uniform
`LBLOCK`	≤ 12	16	Non-Uniform
`TWINE`	14	16	Non-Uniform

C An Example of Preprocessing Phase of `Cons-Solver`

The following 3 nonlinear constraints are detected by `Cons-Collector` from the `SKINNY-64` differential trail in Zhang's paper [ZZ18], where the variables inside square brackets are constrained variables.

In the preprocessing phase, the `Cons-Solver` first combine the linear equations, which have no impact on any variables, within the same linear layers. Thereby, the first two equations within nonlinear constraint $\mathbb{E}_3$ are XORed with each other. Then the nonlinear constraint $\mathbb{E}_1$ and $\mathbb{E}_2$ are grouped together as they are related with each other.

Table 5: Nonlinear constraint subsets of SKINNY-64 differential trail

Nonlinear Constraint $\mathbb{E}_1$	**Nonlinear Constraint** $\mathbb{E}_2$	**Nonlinear Constraint** $\mathbb{E}_3$
$[y_4^0] \oplus [y_{11}^0] \oplus x_9^1 \oplus k_4 = 0$	$[y_4^0] \oplus [y_{11}^0] \oplus x_9^1 \oplus k_4 = 0$	$y_1^0 \oplus x_5^1 \oplus k_1 = 0$
$[y_3^1] \oplus y_9^1 \oplus [x_{15}^2] \oplus k_{13} = 0$	$[y_3^1] \oplus y_9^1 \oplus [y_{12}^1] \oplus [x_3^2] \oplus k_{13} = 0$	$y_1^0 \oplus [y_{11}^0] \oplus [x_{13}^1] \oplus k_1 = 0$
$S(x_9^1) = y_9^1$	$S(x_9^1) = y_9^1$	$y_5^1 \oplus [y_8^1] \oplus [x_{10}^2] \oplus k_{14} = 0$
		$S(x_5^1) = y_5^1$

As a result, the subsets are finally grouped as $\mathbb{E}_1 \cup \mathbb{E}_2$ and $\mathbb{E}_3$, as shown in Table 6.

Table 6: Preprocessed constraint subsets from Table 5

Merged $\mathbb{E}_1 \cup \mathbb{E}_2$	Independent $\mathbb{E}_3$
$[y_4^0 \oplus y_{11}^0] \oplus x_9^1 \oplus k_4 = 0$	$x_5^1 \oplus [y_{11}^0 \oplus x_{13}^1] = 0$
$[y_3^1 \oplus x_{15}^2] \oplus y_9^1 \oplus k_{13} = 0$	$y_5^1 \oplus [y_8^1 \oplus x_{10}^2] \oplus k_{14} = 0$
$[y_3^1 \oplus y_{12}^1 \oplus x_3^2] \oplus y_9^1 \oplus k_{13} = 0$	$S(x_5^1) = y_5^1$
$S(x_9^1) = y_9^1$	

D Detailed Detection Process of SKINNY-64 Trail T_{S0}

The following example demonstrates how Cons-Collector identifies constraints in a differential trail. In this example, we focus on the first three rounds of a SKINNY-64 differential trail from Table 5 in [DDH+21], as depicted in Figure 10.

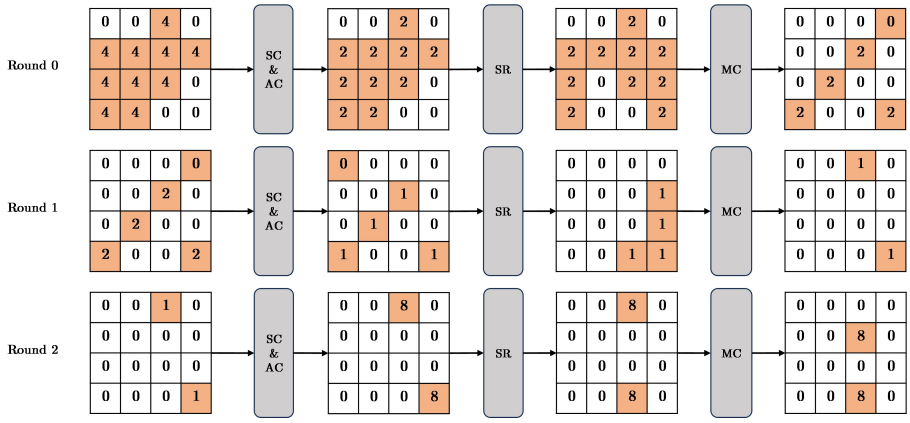


Figure 10: The first 3 rounds of SK differential characteristic from the Table 6 in [DDH+21], cells with orange color are active S-boxes

The Cons-Collector first generalize the algebraic structure of SKINNY-64 cipher into an equation system $\mathcal{E}$. In the equation system $\mathcal{E}$, consider the following Equation (3), which is then transformed into a dependency equation set:

$$\begin{cases} y_3^0 \oplus [y_9^0] \oplus [y_{12}^0] \oplus k_3 = x_3^1 \\ y_3^0 \oplus [y_9^0] \oplus k_3 = [x_{15}^1] \\ y_3^1 \oplus [y_9^1] \oplus [x_{15}^2] \oplus k_{13} = 0 \\ y_3^1 = S(x_3^1) \end{cases} \rightarrow \begin{cases} y_3^0 \leftrightarrow [y_9^0] \leftrightarrow [y_{12}^0] \leftrightarrow k_3 \leftrightarrow x_3^1 \\ y_3^0 \leftrightarrow [y_9^0] \leftrightarrow k_3 \leftrightarrow [x_{15}^1] \\ y_3^1 \leftrightarrow [y_9^1] \leftrightarrow [x_{15}^2] \leftrightarrow k_{13} \\ y_3^1 \leftrightarrow x_3^1, \end{cases} \tag{3}$$

The first three equations of the subset are extracted from the linear layers of 1st and 2nd rounds of the cipher. The fourth equation is the nonlinear function between x_3^1 and

y_3^1. This equation set satisfies the Corollary 1 in Section 3.4, and the combination result of the dependency equations is $k_{13} \leftrightarrow [y_{12}^0] \leftrightarrow [y_9^1] \leftrightarrow [x_{15}^2]$, which forms a constraint on key variables.

In the context of the `Cons-Collector`, all linear and nonlinear equations are converted into a binary matrix M. Consequently, the Equation (3) can be reformulated into the matrix product form:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} y_3^0 \\ y_9^0 \\ y_{12}^0 \\ x_3^1 \\ x_{15}^1 \\ y_3^1 \\ y_9^1 \\ k_3 \\ k_{13} \end{pmatrix} = M \cdot (\vec{X}, \vec{Y}, \vec{K}),$$

In M, all the coefficients of $\vec{X}$ and $\vec{Y}$ can be eliminated by XOR summation. This process yields the final result: $k_{13} = 0$, indicating that the Equation (3) satisfies Corollary 1 and imposes a constraint on k_{13}:

$$S([y_{12}^0] \oplus [x_{15}^1]) \oplus [y_9^1] \oplus [x_{15}^2] \oplus k_{13} = 0$$

where the variables in orange color are constrained variables produced by the active cells in differential trail. Based on this constraint, the valid space of key variable k_{13} is:

$$k_{13} \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13\}.$$

E Differential Trails of `SKINNY-64`

In the verification experiment, the trail T_{S0} is the first 3-round extracted from trail in Table 6 [DDH+21]. The differential trail is shown below:

Table 7: 3-round differential trail T_{S0} derived from Table 6 [DDH+21]

Round	ΔI_s	ΔO_s
0	0x0040444444404400	0x0020222222202200
1	0x0000002002002002	0x0000001001001001
2	0x0010000000000001	0x0080000000000008

F Differential Trails of `LBLOCK`

Trail T_{L0} is the first 4 round extracted from 16-round trail in Zhou's paper [ZZDX19].While, trail T_{L1} in Table 9 has the totally same structure as T_{L0}, with only the 6-th output nibble value different. The original 16-round trail is denoted as T_{BLOCK}.

Table 9: 4-round differential trail of `LBLOCK`: T_{L0}

Round	ΔI_s	ΔO_s
0	00040000	00020000
1	40400000	40200000
2	04420000	04650000
3	05060040	04020040

Table 10: 4-round differential trail of LBLOCK: T_{L1}

Round	ΔI_s	ΔO_s
0	00040000	00020000
1	40400000	40200000
2	04420000	04650000
3	05060040	06020040

Table 11: 16-round differential trail of LBLOCK from [ZZDX19]

Round	ΔI_s	ΔO_s
0	0x00424000	0x00218000
1	0x00040000	0x00020000
2	0x40400000	0x40200000
3	0x04420000	0x04650000
4	0x05060040	0x04020040
5	0x00000000	0x00000000
6	0x06004005	0x0100C00A
7	0x10000BC0	0xF0000260
8	0x00B02500	0x00A01C00
9	0x00000000	0x00000000
10	0xB0250000	0x20120000
11	0x02210000	0x02150000
12	0x000100B0	0x00010020
13	0x20000000	0xA0000000
14	0x01A0B000	0x0A102000
15	0xA0010000	0xE0050000

G Differential Trail of TWINE

Table 12 below shows the optimal 9-round differential trail for TWINE, generated using MILP solver.

Table 12: 9-Round differential trail T_{TW1} of TWINE with probability of 2^{-28}

Round	ΔI_s	ΔO_s
0	00870007	00390009
1	0A090000	07080000
2	07000000	09000000
3	A0000000	70000000
4	00000000	00000000
5	000000A0	00000070
6	00000700	00000900
7	0900000A	08000007
8	00870007	00390009

H Important Nonlinear and Linear Constraints in SKINNY Trails

H.1 Infeasible Nonlinear and Linear Constraints of 3-round Trail from Figure 2 [ZZ18]

In this part, two nonlinear constraint subsets and one linear constraint subset within 3-5 round of the 11 round-trail from Figure 2 [ZZ18] are listed. The prediction made by `Cons-Solver` can be easily verified through simple calculations, as $x_{15}^2 = [\texttt{0x8}, \texttt{0xD}], y_{12}^1 = [\texttt{0x0}, \texttt{0x8}], x_3^2 = [\texttt{0x3}, \texttt{0xE}]$ and $x_{15}^2 \oplus y_{12}^1 \oplus x_3^2 = [\texttt{0x3}, \texttt{0xB}, \texttt{0x6}, \texttt{0xE}] \neq 0$. Meanwhile, `Cons-Solver` output $Sol(\mathbb{E}_1 \cup \mathbb{E}_2) = \varnothing$, which means two nonlinear constraints together make the trail infeasible as well.

Table 13: Infeasible nonlinear constraints and linear constraint within Figure 2 [ZZ18]

Nonlinear Constraint $\mathbb{E}_1$	**Nonlinear Constraint** $\mathbb{E}_2$	**Linear Constraint** $\mathbb{E}_3$
$[y_4^0] \oplus [y_{11}^0] \oplus x_9^1 \oplus k_4 = 0$	$[y_4^0] \oplus [y_{11}^0] \oplus x_9^1 \oplus k_4 = 0$	$[x_{15}^2] \oplus [y_{12}^1] \oplus [x_3^2] = 0$
$[y_3^1] \oplus y_9^1 \oplus [x_{15}^2] \oplus k_{13} = 0$	$[y_3^1] \oplus y_9^1 \oplus [y_{12}^1] \oplus [x_3^2] + k_{13} = 0$	
$S(x_9^1) = y_9^1$	$S(x_9^1) = y_9^1$	

H.2 Nonlinear Constraints found in first 3 rounds of Table 6 [DDH+21]

A new nonlinear constraint is detected in the first 3 rounds of the trail from Table 6 [DDH+21]. This trail is denoted as T_{S0}.

Table 14: Nonlinear constraint within first 3 rounds of the trail in Table 6 [DDH+21]

Nonlinear Constraint $\mathbb{E}_6$
$y_3^0 \oplus [y_9^0] \oplus [y_{12}^0] \oplus x_3^1 \oplus k_3 = 0$
$y_3^0 \oplus [y_9^0] \oplus [x_{15}^1] \oplus k_3 = 0$
$y_3^1 \oplus [y_9^1] \oplus [x_{15}^2] \oplus k_{13} = 0$
$S(x_3^1) = y_3^1$

H.3 Other Solvable Nonlinear Constraints within Table 6 [DDH+21]

Besides the above nonlinear constraint $\mathbb{E}_6$, we find another two minimal nonlinear constraints which are solvable.

Table 15: Nonlinear constraints within Table 6 [DDH+21]

Nonlinear Constraint $\mathbb{E}_4$	**Nonlinear Constraint** $\mathbb{E}_5$
$y_3^0 \oplus x_7^1 \oplus k_3 = 0$	$y_0^0 \oplus [y_{10}^0] \oplus [y_{13}^0] \oplus x_0^1 \oplus k_0 = 0$
$y_3^0 \oplus [y_9^0] \oplus [x_{15}^1] \oplus k_3 = 0$	$y_0^0 \oplus [y_{10}^0] \oplus [x_{12}^1] \oplus k_0 = 0$
$[y_5^0] \oplus [y_8^0] \oplus x_{10}^1 \oplus k_5 = 0$	$y_0^1 \oplus x_4^2 \oplus k_9 = 0$
$y_7^1 \oplus y_1^10 \oplus x_8^2 \oplus k_{11} = 0$	$[y_6^1] \oplus [y_9^1] \oplus x_{11}^2 \oplus k_{12} = 0$
$[y_2^2] \oplus y_8^2 \oplus [x_{14}^3] \oplus k_0 = 0$	$y_4^2 \oplus y_1^21 \oplus x_9^3 \oplus k_2 = 0$
$S(x_7^1) = y_7^1$	$[y_6^3] \oplus y_9^3 \oplus [x_{11}^4] \oplus k_{10} = 0$
$S(x_{10}^1) = y_{10}^1$	$S(x_0^1) = y_0^1$
$S(x_8^2) = y_8^2$	$S(x_4^2) = y_4^2$
	$S(x_{11}^2) = y_{11}^2$
	$S(x_9^3) = y_9^3$

H.4 Infeasible Linear Constraints of 10-round SKINNY-64 trail

In Table 16, a linear constraint that causes infeasibility in 10-round trail from Table 7 [DDH+21].

Table 16: Infeasible linear constraint within Table 7 [DDH+21]

Infeasible Linear Constraint
$y_0^0 + [y_1^0 0] + [y_{13}^0] + [x_0^1] + k_0 = 0$
$y_0^0 + [x_4^1] + k_0 = 0$

H.5 New Solvable Nonlinear Constraint in 14-round Trail

Trail-Estimator find a new nonlinear constraint which reduce the probability of the differential trail, without imposing constraints on any keys. The constraint set value restriction for constrained variable $[y_{12}^1] \oplus [x_3^2]$ and $[x_{14}^3] \oplus [x_2^3]$.

Table 17: Nonlinear constraint $\mathbb{E}_7$

Nonlinear Constraint $\mathbb{E}_7$
$y_3^1 \oplus y_9^1 \oplus [y_{12}^1] \oplus [x_3^2] \oplus k_{13} = 0$
$y_3^1 \oplus y_9^1 \oplus x_{15}^2 \oplus k_{13} = 0$
$y_2^2 \oplus [y_8^2] \oplus y_{15}^2 \oplus [x_2^3] \oplus k_0 = 0$
$y_2^2 \oplus [y_8^2] \oplus [x_{14}^3] \oplus k_0 = 0$
$S(x_{15}^2) = y_{15}^2$

H.6 New Solvable Nonlinear Constraints in 16-round SKINNY-128 Trail

Two new solvable nonlinear constraints are detected in 16-round SKINNY-128 trail, as shown in Table 18.

Table 18: New solvable nonlinear constraints in Table 11 [DDH+21]

Nonlinear Constraint $\mathbb{E}_8$	**Nonlinear Constraint** $\mathbb{E}_9$
$[y_1^2] + y_{11}^2 + [y_1^2 4] + x_1^3 + k_1^2 = 0$	$y_0^3 + y_{10}^3 + [y_{13}^3] + [x_0^4] + k_0^3 = 0$
$[y_1^2] + y_{11}^2 + [x_{13}^3] + k_1^2 = 0$	$y_0^3 + y_{10}^3 + x_{12}^4 + k_0^3 = 0$
$y_1^3 + [y_{11}^3] + [x_{13}^4] + k_1^3 = 0$	$[y_3^4] + [y_9^4] + y_{12}^4 + [x_3^5] + k_3^4 = 0$
$S(x_1^3) + y_1^3 = 0$	$S(x_{12}^4) + y_{12}^4 = 0$

H.7 Infeasible Linear Constraints of SKINNY-128 trail in Table 12 [DDH+21]

The 17-round trail in [DDH+21] is invalid due to the following two linear constraints on subkey k_2. Specifically, for the 6-th S-box in the first round of this trail, we obtain that XDDT(0x32,0x92)=$\varnothing$ (which we believe is a typo), thus it is an invalid trail.

Table 19: Infeasible linear constraint within Table 12 [DDH+21]

2 Infeasible Linear Constraints
$[y_2^0] \oplus [y_8^0] \oplus [x_{14}^1] \oplus k_2 = 0$
$[y_2^0] \oplus [x_6^1] \oplus k_2 = 0$

I Solvable Nonlinear and Linear Constraints in LBLOCK

I.1 The Only Constraint in Trail T_{L0} and T_{L1}

Since the active cell positions in T_{L0} and T_{L1} are all the same, so these two trails have the same nonlinear constraint structure, while the value of constrained variable x_6^3 is different, which result in totally different probability distribution for these two trails.

Table 20: The only nonlinear constraint found in T_{L0} and T_{L1}

Nonlinear Constraint E_{L0}
$[x_4^0] \oplus x_{12}^1 \oplus k_4^0 = 0$
$x_{12}^1 \oplus y_4^1 \oplus [x_6^2] \oplus k_6^1 = 0$
$x_4^1 \oplus x_{12}^2 \oplus k_4^1 = 0$
$x_{12}^2 \oplus [y_4^2] \oplus [x_6^3] \oplus k_6^2 = 0$
$S_4(x_4^1) = y_4^1$

I.2 Solvable Nonlinear and Linear Constraint of 16-round Trail from [ZZDX19]

Within the 16-round trail from [ZZDX19], `Trail-Estimator` found the following 5 solvable nonlinear constraints, two of them are long nonlinear constraints spanning over 5 rounds, shown in Table 21.

Table 21: Solvable nonlinear constraint subsets within T_{LBLOCK}, which has impact on the distribution of trail's probability

Nonlinear Constraint $\mathbb{E}_{L1}$	**Linear Constraint** $\mathbb{E}_{L2}$	**Nonlinear Constraint** $\mathbb{E}_{L3}$
$[x_4^1] + x_{12}^2 + k_4^1 = 0$	$[x_2^7] + x_{10}^8 + k_2^7 = 0$	$[x_4^{10}] + x^1 1_1 2 + k^1 0_4 = 0$
$x_{12}^2 + y_4^2 + [x_6^3] + k_6^2 = 0$	$x_{10}^8 + [y_5^8] + x_4^9 + k_4^8 = 0$	$x_{12}^{11} + [y^1 1_4] + x_6^{12} + k_6^{11} = 0$
$x_4^2 + x_{12}^3 + k_4^2 = 0$	$x_4^9 + x_{12}^{10} + k_4^9 = 0$	$[x_5^{11}] + x_1^{12} 3 + k_5^{11} = 0$
$x_{12}^3 + [y_4^3] + [x_6^4] + k_6^3 = 0$	$x_{12}^{10} + [y_4^{10}] + [x_6^{11}] + k_6^{10} = 0$	$x_{13}^{12} + y_6^{12} + [x_7^{13}] + k_7^{12} = 0$
$S(x_4^2) + y_4^2 = 0$		$S(x_6^{12}) = y_6^{12}$
Nonlinear Constraint $\mathbb{E}_{L4}$	**Nonlinear Constraint** $\mathbb{E}_{L5}$	
$[x_7^7] + x_{15}^8 + k_7^7 = 0$	$[x_5^2] + x_{13}^3 + k_5^2 = 0$	
$x_{15}^8 + [y_3^8] + x_1^9 + k_1^8 = 0$	$[x_7^2] + x_{15}^3 + k_7^2 = 0$	
$x_1^9 + x_9^{10} + k_1^9 = 0$	$x_{15}^3 + y_3^3 + [x_1^4] + k_1^3 = 0$	
$x_9^{10} + y_2^{10} + x_3^{11} + k_3^{10} = 0$	$x_{13}^3 + [y_6^3] + x_7^4 + k_7^3 = 0$	
$x_2^{10} + x_{10}^{11} + k_2^{10} = 0$	$x_3^3 + x_{11}^4 + k_3^3 = 0$	
$[x_7^{10}] + x_1^{11} 5 + k_7^{10} = 0$	$x_{11}^4 + y_7^4 + x_5^5 + k_5^4 = 0$	
$x_1^{11} 5 + y_3^{11} + [x_1^{12}] + k_1^{11} = 0$	$x_5^5 + x_1^6 3 + k_5^5 = 0$	
$x_{10}^{11} + [y_5^{11}] + [x_4^{12}] + k_4^{11} = 0$	$x_{13}^6 + [y_6^6] + [x_7^7] + k_7^6 = 0$	
$S(x_2^{10}) + y_2^{10} = 0$	$S(x_3^3) + y_3^3 = 0$	
$S(x_3^{11}) + y_3^{11} = 0$	$S(x_7^4) + y_7^4 = 0$	

J Solvable Nonlinear and Linear Constraints in TWINE

In Table 22 and Table 23, we show the solvable nonlinear minimal constraint subsets identified in `TWINE`. For T_{TW1}, we identified 3 long constraints, all of which span across

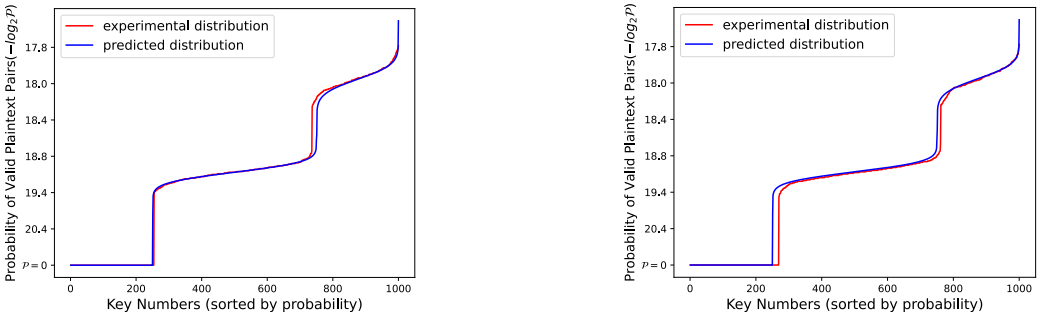
over 10 rounds. While in T_{TW2}, there are 4 nonlinear constraints and 4 linear constraints which can be solved by `Cons-Solver`.

Table 22: Solvable nonlinear constraint subsets within T_{TW1}

Nonlinear Constraint $\mathbb{E}_{tw0}$	**Nonlinear Constraint** $\mathbb{E}_{tw1}$	**Nonlinear Constraint** $\mathbb{E}_{tw2}$
$x_2^2 \oplus x_1^3 \oplus k_1^2 = 0$	$[x_2^1] \oplus x_1^2 \oplus k_1^1 = 0$	$x_4^0 \oplus x_7^1 \oplus k_2^0 = 0$
$x_1^3 \oplus y_0^3 \oplus x_0^4 \oplus k_0^4 = 0$	$[x_6^1] \oplus x_3^2 \oplus k_3^1 = 0$	$x_6^0 \oplus x_3^1 \oplus k_3^0 = 0$
$x_0^4 \oplus x_5^5 \oplus k_0^4 = 0$	$[x_0^2] \oplus x_5^3 \oplus k_0^2 = 0$	$x_{14}^0 \oplus x_{11}^1 \oplus k_7^0 = 0$
$[x_4^5] \oplus x_7^6 \oplus k_2^5 = 0$	$x_1^2 \oplus [y_0^2] \oplus x_0^3 \oplus k_0^3 = 0$	$x_3^1 \oplus [y_2^1] \oplus [x_4^2] \oplus k_2^2 = 0$
$x_5^5 \oplus [y_4^5] \oplus [x_{12}^6] \oplus k_6^6 = 0$	$x_3^2 \oplus y_2^2 \oplus [x_4^3] \oplus k_2^3 = 0$	$x_7^1 \oplus [y_6^1] \oplus [x_8^2] \oplus k_4^2 = 0$
$[x_6^6] \oplus x_3^7 \oplus k_3^6 = 0$	$x_5^3 \oplus [y_4^3] \oplus x_{12}^4 \oplus k_6^4 = 0$	$x_{10}^1 \oplus x_9^2 \oplus k_5^1 = 0$
$x_7^6 \oplus [y_6^6] \oplus x_8^7 \oplus k_4^7 = 0$	$x_{12}^4 \oplus x_{15}^5 \oplus k_6^4 = 0$	$x_{11}^1 \oplus y_{10}^1 \oplus x_2^2 \oplus k_1^2 = 0$
$x_1^60 \oplus x_9^7 \oplus k_5^6 = 0$	$x_{12}^5 \oplus x_{15}^6 \oplus k_6^5 = 0$	$[x_4^2] \oplus x_7^3 \oplus k_2^2 = 0$
$[x_{12}^6] \oplus x_{15}^7 \oplus k_6^6 = 0$	$[x_{14}^5] \oplus x_{11}^6 \oplus k_7^5 = 0$	$x_9^2 \oplus [y_8^2] \oplus [x_6^3] \oplus k_3^3 = 0$
$x_3^7 \oplus [y_2^7] \oplus x_4^8 \oplus k_2^8 = 0$	$x_{15}^5 \oplus [y_{14}^5] \oplus [x_{14}^6] \oplus k_7^6 = 0$	$x_7^3 \oplus [y_6^3] \oplus x_8^4 \oplus k_4^4 = 0$
$x_9^7 \oplus y_8^7 \oplus x_6^8 \oplus k_3^8 = 0$	$x_{11}^6 \oplus y_{10}^6 \oplus [x_2^7] \oplus k_1^7 = 0$	$x_8^4 \oplus x_{13}^5 \oplus k_4^4 = 0$
$x_{15}^7 \oplus [y_{14}^7] \oplus x_{14}^8 \oplus k_7^8 = 0$	$x_{15}^6 \oplus [y_{14}^6] \oplus [x_{14}^7] \oplus k_7^7 = 0$	$x_{13}^5 \oplus y_{12}^5 \oplus x_{10}^6 \oplus k_5^6 = 0$
$x_{10}^{11} \oplus x_9^{12} \oplus k_5^{11} = 0$	$[x_0^{10}] \oplus x_5^{11} \oplus k_0^{10} = 0$	$x_{13}^9 \oplus y_{12}^9 \oplus x_{10}^{10} \oplus k_5^{10} = 0$
$x_3^{12} \oplus y_2^{12} \oplus [x_4^{13}] \oplus k_2^{13} = 0$	$x_{10}^{10} \oplus x_9^{11} \oplus k_5^{10} = 0$	$[x_4^{10}] \oplus x_7^{11} \oplus k_2^{10} = 0$
$x_{12}^{12} \oplus x_{15}^{13} \oplus k_6^{12} = 0$	$x_{15}^{11} \oplus y_{14}^{11} \oplus x_{14}^{12} \oplus k_7^{12} = 0$	$x_{11}^{10} \oplus y_{10}^{10} \oplus x_2^{11} \oplus k_1^{11} = 0$
$y_i^r = S_i(x_i^r)$	$x_7^{12} \oplus y_6^{12} \oplus [x_8^{13}] \oplus k_4^{13} = 0$	$y_i^r = S_i(x_i^r)$
	$y_i^r = S_i(x_i^r)$	

K Probability Distribution of LBLOCK Differential Trails

The following figures show the impact of key scheduling on our estimation. A `LBLOCK` cipher without key scheduling is made, and we ran two simulation experiments with it to see the discrepancy between prediction and experiment probability distribution. In Figure 11a, the prediction is lower than real probability around key number 750, while we got opposite results in Figure 11b, which shows the discrepancy is caused by randomness in master key selecting.



(a) Case 1

(b) Case 2

Figure 11: The predicted and real probability distribution of the 4-round trail T_{L1} of `LBLOCK` without key scheduling

Table 23: Solvable nonlinear constraint subsets within T_{TW2}

Nonlinear Cons $\mathbb{E}_{tw3}$	**Nonlinear Cons** $\mathbb{E}_{tw4}$	**Nonlinear Cons** $\mathbb{E}_{tw5}$
$[x_{12}^6] \oplus x_{15}^7 \oplus k_6^6 = 0$	$[x_2^1] \oplus x_1^2 \oplus k_1^1 = 0$	$[x_4^2] \oplus x_7^3 \oplus k_2^2 = 0$
$x_{15}^7 \oplus [y_{14}^7] \oplus x_{14}^8 \oplus k_7^8 = 0$	$[x_6^1] \oplus x_3^2 \oplus k_3^1 = 0$	$x_7^3 \oplus [y_6^3] \oplus x_8^4 \oplus k_4^4 = 0$
$x_{14}^8 \oplus x_{11}^9 \oplus k_7^8 = 0$	$x_1^2 \oplus [y_0^2] \oplus x_0^3 \oplus k_0^3 = 0$	$x_8^4 \oplus x_{13}^5 \oplus k_4^4 = 0$
$[x_6^9] \oplus x_3^{10} \oplus k_3^9 = 0$	$x_2^2 \oplus x_1^3 \oplus k_1^2 = 0$	$x_{12}^5 \oplus x_{15}^6 \oplus k_6^5 = 0$
$x_{10}^9 \oplus x_9^{10} \oplus k_5^9 = 0$	$x_3^2 \oplus y_2^2 \oplus [x_4^3] \oplus k_2^3 = 0$	$x_{13}^5 \oplus y_{12}^5 \oplus x_{10}^6 \oplus k_5^6 = 0$
$x_{11}^9 \oplus y_{10}^9 \oplus x_2^{10} \oplus k^1 0_1 = 0$	$x_1^3 \oplus y_0^3 \oplus x_0^4 \oplus k_0^4 = 0$	$[x_{14}^5] \oplus x_{11}^6 \oplus k_7^5 = 0$
$x_3^{10} \oplus y_2^{10} \oplus [x_4^{11}] \oplus k^1 1_2 = 0$	$x_0^4 \oplus x_5^5 \oplus k_0^4 = 0$	$x_{11}^6 \oplus y_{10}^6 \oplus [x_2^7] \oplus k_1^7 = 0$
$x_9^{10} \oplus [y_8^{10}] \oplus [x_6^{11}] \oplus k^1 1_3 = 0$	$x_5^5 \oplus [y_4^5] \oplus [x_{12}^6] \oplus k_6^6 = 0$	$x_{15}^6 \oplus [y_{14}^6] \oplus [x_{14}^7] \oplus k_7^7 = 0$
$S(x_{10}^9) \oplus y_{10}^9 = 0$	$S(x_2^2) \oplus y_2^2 = 0$	$S(x_{12}^5) \oplus y_{12}^5 = 0$
$S(x_2^{10}) \oplus y_2^{10} = 0$	$S(x_0^3) \oplus y_0^3 = 0$	$S(x_{10}^6) \oplus y_{10}^6 = 0$
Nonlinear Cons $\mathbb{E}_{tw3}$	**Linear Cons** $\mathbb{E}_{tw4}$	**Linear Cons** $\mathbb{E}_{tw5}$
$[x_0^9] \oplus x_5^{10} \oplus k_0^9 = 0$	$[x_8^2] + x_{13}^3 + k_4^2 = 0$	$[x_8^5] + x_{13}^6 + k_4^5 = 0$
$x_5^{10} \oplus [y_4^{10}] \oplus x_{12}^{11} \oplus k^1 1_6 = 0$	$x_{13}^3 + [y_{12}^3] + x_{10}^4 + k_5^4 = 0$	$x_{13}^6 + [y_{12}^6] + [x_{10}^7] + k_5^7 = 0$
$[x_8^{10}] \oplus x_{13}^{11} \oplus k^1 0_4 = 0$	$x_{10}^4 + x_9^5 + k_5^4 = 0$	
$x_{13}^{11} \oplus y_{12}^{11} \oplus [x_{10}^{12}] \oplus k^1 2_5 = 0$	$x_9^5 + [y_8^5] + [x_6^6] + k_3^6 = 0$	
$S(x_{12}^{11}) \oplus y_{12}^{11} = 0$		
Linear Cons $\mathbb{E}_{tw6}$	**Linear Cons** $\mathbb{E}_{tw7}$	
$[x_{14}^6] + x_{11}^7 + k_7^6 = 0$	$[x_6^6] + x_3^7 + k_3^6 = 0$	
$x_{11}^7 + [y_{10}^7] + x_2^8 + k_1^8 = 0$	$x_3^7 + [y_2^7] + x_4^8 + k_2^8 = 0$	
$x_2^8 + x_1^9 + k_1^8 = 0$	$x_4^8 + x_7^9 + k_2^8 = 0$	
$x_1^9 + [y_0^9] + [x_0^{10}] + k_0^{10} = 0$	$x_7^9 + [y_6^9] + [x_8^{10}] + k_4^{10} = 0$	