IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2025, No. 3, pp. 475–515. DOI:10.46586/tosc.v2025.i3.475-515

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@,quanquan001@e,hongyi003@e,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 17 differential trails for the `SKINNY`, `LBLOCK`, `TWINE`, and `AES` block ciphers. Experimental results show that `Trail-Estimator` consistently identifies previously undetected constraints for `SKINNY` and `AES`, 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.

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

Licensed under Creative Commons License CC-BY 4.0.

round functions and key schedules, it is unsurprising that conventional assumptions are becoming increasingly inadequate. Murphy and Robshaw [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.

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. Further work by Heys [Hey20] revealed that variations in key scheduling schemes can result in unpredictable differential properties, and a cipher can be more vulnerable to differential cryptanalysis for specific key subsets. 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] investigated the key dependencies in SPN-based designs and developed a method for detecting linear and nonlinear constraints in differential trails for `SKINNY` and `GIFT-64`. These constraints impose specific restrictions on key values, significantly affecting the probability distribution of the trails and leading to the invalidation of several previously accepted trails. 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 discovery of such constraints enables a more detailed understanding of the relationships among internal variables, which is crucial for detecting invalid trails and analyzing the distribution of the key space. 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`, `TWINE`, and `AES`.

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. For example, in the 7-round SKINNY-64 trail, we identified 2 new short nonlinear constraints (spanning less than or equal to 5 rounds) and 4 new long nonlinear constraints (spanning more than 5 rounds), as detailed in Table 1. These constraints uncover complex variable dependencies that were not identified before, enabling a more accurate characterization of the trail's probability distribution (as shown in the mini-figure in Table 2), and reducing the valid key space by $2^{-7.3}$ compared to previous works.

All source code is available at the GitHub repository:

https://github.com/Qtarox/FSE_TRAIL_ESTIMATOR

Table 1: Results comparison between Trail-Estimator and previous works. We note that no long nonlinear constraint was found by any previous work, while all linear constraints (except for AES) were recovered.

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	Figure 6 [NGJE25]
	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 10 Appendix J.2
	16	2	-	-	-	9 (2)	1	8	Table 14 [ZZDX19]
TWINE	9	0	-	-	-	3 (3)	3	0	Table 13 Appendix J.3
	15	2	-	-	-	8 (3)	1	6	Table 16 [ZZDX19]
AES	2	0	-	-	-	0 (0)	0	1	Table 14 Appendix J.4
	2	0	-	-	-	0 (0)	0	1	Table 15 Appendix J.4
	3	0	-	-	-	0 (0)	0	5	Table 16 Appendix J.4

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 Cons-Collector, including the constraint propagation rules and detection principles. Section 4 introduces Cons-Solver, including the methods for computing the probability distribution and optimization techniques. Section 5 presents the experimental results of applying Trail-Estimator to SKINNY, LBLOCK, TWINE, and AES block ciphers. 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

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

Cipher	R	Stated Prob.	Reduced Key Space	Prob. by Trail-Estimator	Average-Prob. Valid Keys	Prob. Distr.	Trail Source
SKINNY	7	2^{-52}	$2^{-7.3}$	$2^{-49} - 2^{-42.4}$	$2^{-44.6}$		Table 6 [DDH+21]
	10	2^{-46}	0	-	-	No Valid Key	Table 7 [DDH+21]
	13	2^{-55}	2^{-4}	2^{-51}	2^{-51}		Table 8 [DDH+21]
	15	2^{-54}	$2^{-6.2}$	$2^{-48} - 2^{-47}$	$2^{-47.8}$		Table 9 [DDH+21]
	14	2^{-120}	$2^{-7.7}$	$2^{-114.7} - 2^{-110.4}$	2^{-112}		Table 10 [DDH+21]
	16	$2^{-127.6}$	$2^{-7.6} - 2^{-6.2[E]}$	$2^{-127.2} - 2^{-108.2[E]}$	$2^{-120.5}$		Table 11 [DDH+21]
	17	2^{-110}	$2^{-8.37} - 2^{-6.21[E]}$	$2^{-108.2} - 2^{-98.6[E]}$	$2^{-102.7[E]}$		Figure 6 [NGJE25]
	8	$2^{-19}/2^{-17}$	$2^{-3}/2^{-2}$	$2^{-16}/2^{-15}$	$2^{-16}/2^{-15}$		Table 16 [QDW+21]
	10	2^{-42}	1	2^{-42}	2^{-42}		Table 7 [PT22]
	11	2^{-147}	0	-	-	No Valid Key	Figure 2 [ZZ18]
LBLOCK	4	2^{-18}	1	2^{-18}	2^{-18}		Table 10 Appendix J.2
	16	2^{-72}	2^{-1}	$2^{-72.7} - 2^{-69.9}$	2^{-71}		Table 14 [ZZDX19]
TWINE	9	2^{-28}	$0.99 - 1^{[E]}$	$2^{-30.3} - 2^{-26.9[E]}$	$2^{-28[E]}$		Table 13 Appendix J.3
	15	2^{-68}	$2^{-4.7}$	$2^{-67} - 2^{-61.3}$	$2^{-63.3}$		Table 16 [ZZDX19]
AES	2	2^{-35}	2^{-4}	2^{-31}	2^{-31}		Table 14 Appendix J.4
	2	2^{-30}	1	2^{-30}	2^{-30}		Table 15 Appendix J.4
	3	2^{-147}	2^{-20}	2^{-127}	2^{-127}		Table 16 Appendix J.4

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; Average-Prob. Valid Keys: The average probability of differential trail within the valid key space; [E]: the result is calculated using the estimation method; Prob. Distr.: The mini-figure illustrates the shape of the probability distribution. Detailed versions are provided in Section 5.1 and Appendix L. The figure displaying a horizontal line indicates a uniform distribution.

differential cryptanalysis is captured 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: $P(\Delta_{in} \to \Delta_{out}) = \frac{\#\{F(x)\oplus F(x\oplus\Delta_{in})=\Delta_{out}, \forall\ x\ \in \mathbb{F}_2^n\}}{2^n}$.

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 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 $F_{i-1} \circ ... \circ F_0$ $(0 < i \leq r)$.

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.

We 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, $1 \leq i \leq r$.

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, the `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 illustrate the concept in Figure 1.

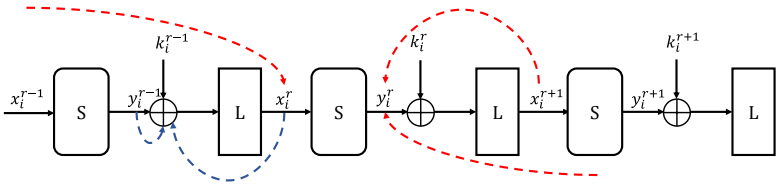


Figure 1: The formation of linear constraint (blue line) and nonlinear constraint (red line)

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 core component of Trail-Estimator designed to identify all the constraints on keys within a differential trail of a block cipher.

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 as 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):

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

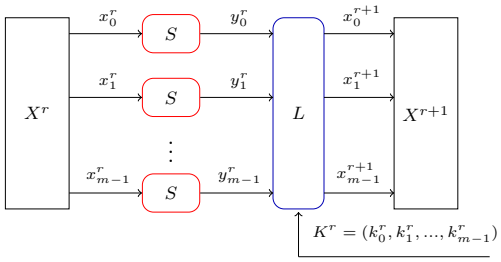


Figure 2: General notation of a round function

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 cannot 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 a 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 propagate 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 $S(x)$. 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 of `SKINNY`, `LBLOCK`, `TWINE`, and `AES`, it is non-uniform. Consequently, even when neither x nor $S(x)$ is constrained, the term $x \oplus S(x)$ is actually constrained. In our framework, we therefore consider $S(x) \oplus x$ as a constrained variable.

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:

$$[x_0] \oplus y_0 \oplus k_0 = 0, \quad [x_1] \oplus y_0 \oplus k_1 = 0.$$

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 $x_0, x_1 \in \{\mathtt{1}, \mathtt{3}, \mathtt{8}, \mathtt{10}\}$, it imposes an effective constraint on keys: $k_0 \oplus k_1 \in \{\mathtt{0}, \mathtt{2}, \mathtt{9}, \mathtt{11}\}$. Conversely, if $x_0 \in \{\mathtt{1}, \mathtt{3}, \mathtt{8}, \mathtt{10}\}$ and $x_1 \in \{\mathtt{2}, \mathtt{3}, \mathtt{6}, \mathtt{7}\}$ (x_0, x_1 are uniformly distributed in these two sets), then $x_0 \oplus x_1$ spans all possible values from `0` to `15` uniformly, 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 $(k_1, k_2) \in \{(S(i), i) \mid 0 \leq i < 2^\omega\}$. 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 the constraint subset, 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.

3.3 Principles of Constraint Detection

According to Definition 8, a constraint subset is derived from the set of equations that defines the differential trail of a cipher, and restricts the value of key variables. However, not all such subsets are equally informative. To capture the core structure of variable dependencies, we introduce the concept of minimal constraint subsets—those with the smallest number of equations that still fully characterize the restrictions. Such minimal subsets provide a concise description of the constraints, making them easier to analyze.

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

All minimal constraint subsets that share common variables are grouped together to form a complete constraint subset. Once all such minimal subsets are identified, the complete set of constraints can be obtained by combining them appropriately. The principle of the constraint detection in `Cons-Collector` is to identify all minimal constraint subsets within a differential trail. Example 1 illustrates the relation between minimal constraint subsets and the constraint subsets.

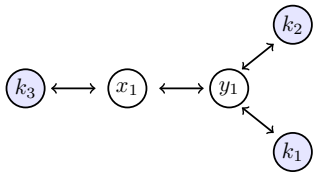
Example 1. Consider the nonlinear constraint subset $\mathbb{E}_{exp}$ found in [PT22]:

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

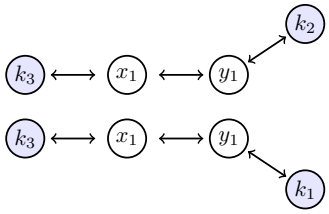
While $\mathbb{E}_{\text{exp}}$ satisfies the definition of the constraint subset, it is not minimal, as it contains a smaller subset $\{x_1 \oplus k_3 = 0,\ y_1 \oplus k_1 = 0,\ y_1 = S(x_1)\}$ which is also a constraint subset. In contrast, the `Cons-Collector` framework identifies two minimal constraint subsets:

$$\begin{aligned}\mathbb{E}_{exp1} &: x_1 \oplus k_3 = 0, \quad y_1 \oplus k_1 = 0, \quad y_1 = S(x_1);\\ \mathbb{E}_{exp2} &: x_1 \oplus k_3 = 0, \quad y_1 \oplus k_2 = 0, \quad y_1 = S(x_1),\end{aligned}$$

which together have a constraining effect equivalent to that of the set $\mathbb{E}_{\text{exp}}$. The comparison between the constraint subset $\mathbb{E}_{exp}$ and minimal constraint subsets is shown in Figure 3.



(a) Structure of the constraint subset $\mathbb{E}_{exp}$



(b) Structure of the minimal constraint subsets $\mathbb{E}_{exp1}$ and $\mathbb{E}_{exp2}$

Figure 3: Comparison between the constraint subset and minimal constraint subsets.

To identify all minimal constraint subsets within a differential trail, the core principle is to eliminate all free variables by summing up the equations. Since free variables 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 summed 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 equation imposes a constraint on the relevant key values still depends on the specific assignments of the constrained variables. We define a potential minimal constraint subset such a system of equations whose summation eliminates all free variables. To formalize this idea, we introduce the Corollary 1, and the proof is shown in Appendix B.

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 the originally constrained variables.

Based on Corollary 1, the problem of detecting all minimal constraint subsets within a differential trail is naturally transformed into identifying linearly dependent subsets of equations. This reformulation enables a systematic and efficient search through a matrix-based representation.

3.4 Efficient Detection via Matrix Representation

We introduce a binary matrix product representation for the equation system $\mathcal{E}$ to capture the constraint propagation pathways between variables, and apply Gaussian elimination to efficiently identify all constraint subsets that eliminate free variables.

Matrix representation for constraint propagation. 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 at round r 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 $a_i^r = 1$ captures the dependency relationships between variables in the propagation pathways $x_i^r \leftrightarrow y_i^r$, $0 \le i < m$, $0 \le r < R$. For the m linear equations at round r in Equation (1), each can be expressed as: $c_i^r \cdot x_i^{r+1} = (\bigoplus_{j=0}^{m-1} b_{i,j}^r \cdot y_j^r) \oplus (\bigoplus_{j=0}^{m-1} d_{i,j}^r \cdot k_j^r)$, and the full system of equations can be expressed 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 coefficients $b_{i,j}^r, c_i^r, d_{i,j}^r \in \{0, 1\}$, are determined based on the specific linear function, $0 \le i, j < m$, $0 \le r < R$. Finally, the complete algebraic system in Equation (1) can be represented in matrix product form as:

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

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

Refined matrix representation for constraint detection. According to Corollary 1, the goal of `Cons-Collector` is to identify subsets of equations whose summation eliminates all free variables, while allowing constrained and key variables to be preserved. To enable this process, we refine the coefficient matrix M by incorporating information from the given differential trail. Specifically, if an S-box is active, its corresponding input and output variables are constrained variables, since their values are fixed by the trail. As a result, their coefficients in M are set to 0, excluding these variables from the elimination process. Similarly, the coefficients for key variables are also set to 0. In contrast, free variables not constrained by the trail retain their coefficients and participate in the elimination. We denote the resulting refined matrix as M'.

Constraint detection via Gaussian elimination In `Cons-Collector`, the task of identifying all minimal constraint subsets is mathematically equivalent to locating linearly dependent subsets of rows in the refined coefficient matrix M'. Therefore, `Cons-Collector` employs a Gaussian elimination-based detection algorithm, presented in Algorithm 1 in Appendix A, which systematically detects all potential minimal constraint subsets. The computation complexity of Algorithm 1 is $O(n^2 \cdot m)$, where n and m are the number of rows and columns of the matrix M', respectively.

We present Example 2 to illustrate the process of matrix construction and constraint detection.

Example 2. Consider a toy cipher with four state cells and analyze its 1-round single-key differential trail, as depicted in Figure 4.

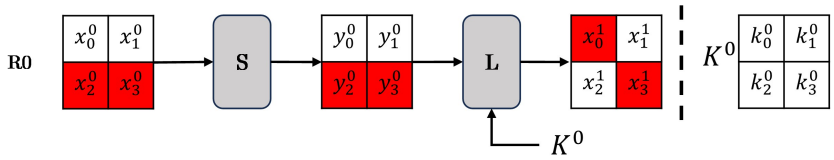


Figure 4: 1-round differential trail of a toy cipher

The complete set of linear and nonlinear equations involved in the cipher is given below:

$$
\begin{aligned}
y_i^0 = S(x_i^0), 0 \le i \le 1,\\
[y_i^0] = S([x_i^0]), 2 \le i \le 3,\\
y_0^0 + y_1^0 + [y_3^0] + [x_0^1] + k_0^0 = 0,\\
y_0^0 + x_1^1 + k_1^0 = 0,\\
y_0^0 + y_1^0 + x_2^1 + k_3^0 = 0,\\
y_0^0 + y_1^0 + [y_2^0] + [x_3^1] + k_2^0 = 0.
\end{aligned}
$$

The Cons-Collector formulates these equations into the matrix representation:

$$\underbrace{\begin{pmatrix}1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&0&1&1&0&1&1&0&0&0&1&0&0&0\\0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0\\0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&1\\0&0&0&0&1&1&1&0&0&0&0&1&0&0&1&0\end{pmatrix}}_{M} \longrightarrow \underbrace{\begin{pmatrix}1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0\\0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\end{pmatrix}}_{M'} \cdot \left(X^0, Y^0, X^1, K^0\right)^T = 0,$$

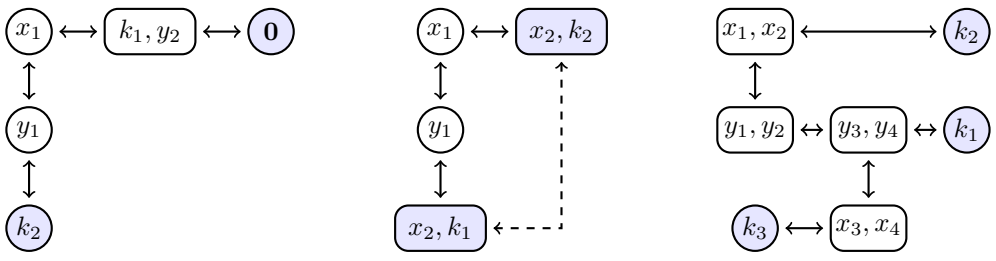
where $X^i = (x_0^i, x_1^i, x_2^i, x_3^i)^T, i \in \{0, 1\}$, $Y^0 = (y_0^0, y_1^0, y_2^0, y_3^0)^T$ and $K^0 = (k_0^0, k_1^0, k_2^0, k_3^0)^T$. And the red-marked zero entries in M' correspond to key and constrained variables, which are determined by the active cells of the differential trail.

After applying Algorithm 1 to the matrix M', Cons-Collector finds that only the 5th and 8th rows are linearly dependent, which correspond to the linear equations $y_0^0 + y_1^0 + [y_3^0] + [x_0^1] + k_0^0 = 0$ and $y_0^0 + y_1^0 + [y_2^0] + [x_3^1] + k_2^0 = 0$, respectively. Therefore, we get one minimal constraint subset on key variables k_0^0 and k_2^0.

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

3.5 Uncovering New Dependencies

Previous research 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.



(a) Dependencies in Example 3 (b) Dependencies in Example 4 (c) Dependencies in Example 5

Figure 5: An illustration of the propagation pathways that were undiscovered.

Example 3. Consider the following system of equations involving the free variables x_1,

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 equations, one can still conclusively determine that $k_2 = S(0)$. The propagation pathway of constraints is shown in Figure 5a. The constraint propagates from knowing the value of x_1 to y_1, which ultimately imposes a restriction on k_2.

Example 4. 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 5b.

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 5. 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 variables. The propagation pathway is illustrated in Figure 5c. 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 based on the constraint programming (CP) solver. Given the constraint subsets identified by `Cons-Collector`, `Cons-Solver` converts them into CP constraints and solves for the corresponding solution

space. Cons-Solver computes the probability distribution of a differential trail that closely approximates the true distribution, and applies constraint simplification techniques before solving to improve efficiency.

4.1 Probability Computation

Typically, Cons-Collector detects multiple constraint subsets $\mathbb{E}_0$, $\mathbb{E}_1$, $\cdots$, $\mathbb{E}_v$ within a differential trail. However, these subsets are generally interdependent, as they may share common variables and impose overlapping conditions. Therefore, the equations are combined into a unified system and solved collectively to obtain the solution space, denoted as $\mathbb{S} = Sol(\mathbb{E}_0 \cup \mathbb{E}_1 \cdots \cup \mathbb{E}_v)$. Next, we introduce two methods to calculate the probability distribution of differential trails.

An accurate method for probability calculation. When the number of variables involved in the equations is not too large, we can enumerate all the possible solutions in $\mathbb{S}$ for the involved subkeys. For each subkey assignment $\vec{k}$, we compute its occurrence frequency $F_{\vec{k}}$ and evaluate the probability of the differential trail T:

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

where n_k denotes the size of the involved subkey and P_0 is the original estimated probability of the trail T under the Markov assumption. Cons-Solver can also estimate the master key space. For ciphers with word-oriented key scheduling (e.g., SKINNY), subkey variables can be expressed as functions of the master key variables in a word-wise manner. In this case, the subkey space directly reflects the master key space. In contrast, for ciphers with bit-oriented key schedules (e.g., LBLOCK), subkey variables are derived from the master key through more complex mappings. In such cases, Cons-Solver assumes subkeys are independent and uses the reduction ratio over the subkey space to estimate the valid master key space. Specifically, suppose the number of valid subkeys satisfying the constraints is 2^{n_v}(i.e., those with $F_{\vec{k}} > 0$), then the reduction ratio over the subkey space is $2^{n_v - n_k}$. The reduction ratio is then applied to the master key space of size 2^{n_K}, yielding an estimated valid master key space of $2^{n_K} \cdot 2^{n_v - n_k}$.

An estimate method for probability calculation. When the number of variables involved in the constraint subsets is too large, enumerating all solutions becomes computationally infeasible due to excessive time and memory requirements. In such cases, Cons-Solver adopts a sampling-based estimation strategy. Specifically, Cons-Solver uniformly and randomly generates 2^{m_1} values for variables x and computes the corresponding $y = S(x)$. It also samples 2^{m_2} master keys, which are expanded through the key schedule to produce the subkeys involved in the constraints. This results in a total of $2^{m_1+m_2}$ samples. Each sample is checked against the constraint system, and those satisfying all constraints form the estimated solution space $\mathbb{S}'$. For each subkey assignment $\vec{k}$, its frequency $F'_{\vec{k}}$ over all samples is used to estimate the probability of the differential trail T:

$$P_{T|\vec{k}} = P_0 \cdot \frac{F'_{\vec{k}}}{|\mathbb{S}'|} \cdot 2^{m_2}.$$

For each sampled master key, Cons-Solver checks whether there exists at least one value of x such that the combined sample satisfies all constraints. If so, the master key is considered valid. Let $\#K$ denote the number of such valid master keys among the 2^{m_2} samples. The reduction ratio of the master key space is approximated by $\frac{\#K}{2^{m_2}}$, and the estimated size of the valid master key space is given by $2^{n_K} \cdot \frac{\#K}{2^{m_2}}$. To assess the reliability of the estimation, we construct a 95% confidence interval using the Wilson score interval [Wil27], which

provides upper and lower bounds within which the true key space size is expected to fall. The detailed formula is provided in Appendix C.

Complexity Analysis of `Cons-Solver`. `Cons-Solver` supports two methods for probability calculation, selected based on the size of the variable space in the constraints. The accurate method has a time complexity of $O(2^{N_b})$, where N_b is the total number of bits in the variables involved. When N_b is small (e.g., $N_b \leq 44$ in our experiments), this method is feasible and preferred. Otherwise, `Cons-Solver` adopts the estimate method with the time complexity $O(2^{m_1+m_2})$, where m_1 is the number of sampled values for the variable x, and m_2 is the number of sampled master keys. A larger m_1 improves the statistical reliability of estimating the lowest probability within the trail, and should be chosen to satisfy the condition $m_1 > -log_2(min_{\vec{k}}(P_{T|\vec{k}}))$. Meanwhile, increasing m_2 enhances the accuracy of the estimated probability distribution curve by sampling over a larger set of master keys. Consequently, m_2 is typically set to the largest value permitted by available computational resources. The selection of m_1 and m_2 is described in the experiment in Section 5, and the detailed execution time of `Cons-Collector` and `Cons-Solver` is listed in Table 4 in Appendix D.

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 6. 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 6 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 6 are closely related to 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 subsets 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 it remains constrained. If this auxiliary variable is still constrained, the constraint subset remains effective. However, if it becomes a free variable, the entire set will not impose restrictions on key variables, thus, it 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 6, 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 applied `Trail-Estimator` to analyze differential trails of `SKINNY-64`, `SKINNY-128`, `TWINE`, `LBLOCK`, and `AES-128`. For targeted differential trails, `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. In addition, the results for `AES` are shown in Appendix H.

5.1 Application to SKINNY

`SKINNY` is a lightweight block cipher family introduced by Beierle et al. [BJK+16], adopting a tweakable SPN-based structure. We applied `Trail-Estimator` to 10 differential trails of the `SKINNY` family, obtained from [ZZ18, DDH+21, PT22, QDW+21]. For each trail, `Trail-Estimator` detected more nonlinear constraints than previous works and achieved a more precise estimation of the probability distribution. The detailed probability distributions derived from these constraints are presented in Table 3.

Results summary. We applied `Trail-Estimator` to 10 `SKINNY` differential trails and highlighted here the main differences with previous works. For the 7-round `SKINNY-64-64` trail, `Trail-Estimator` identified 3 new solvable nonlinear constraints. For the 15-round `SKINNY-64-192` trail from [DDH+21], due to the substantial size of the nonlinear constraints, only one remains solvable [1]. While for `SKINNY-64-128` and `SKINNY-64-192` trails, our results are consistent with those reported in [PT22]; and we identified more long nonlinear constraints that span more than 5 rounds. For the 14-round differential trail of `SKINNY-128-128` from Table 10 of [DDH+21], `Trail-Estimator` identified four solvable nonlinear constraints, including one new nonlinear constraint, as detailed in Appendix K.5. For the 16-round trail of `SKINNY-128-256` from Table 11 of [DDH+21], `Trail-Estimator` found two new solvable nonlinear constraints (see Appendix K.6). For the 17-round `SKINNY-128-384` trail presented in Figure 6 of [NGJE25], `Trail-Estimator` identified 7 solvable nonlinear constraints, which were merged into two final nonlinear constraints, as shown in Table 23 in Appendix K.7. The predicted probability distribution of three `SKINNY-128` trails can be found in Figure 6. We observed that as the complexity of the constraints increases, the resulting probability distribution approaches a Gaussian-like

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

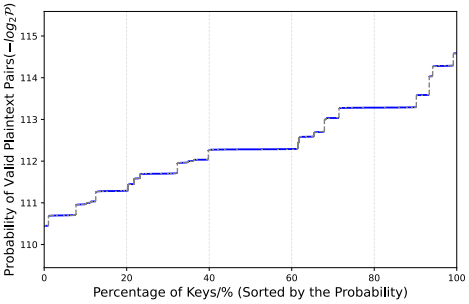
Table 3: Experimental results for SKINNY

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	$2^{-7.7}$	120	114.7-113.4 10.01%	113.3-113 21.41%	112.9-112.4 28.92%	112.3-111.4 27.21%	111.3-110.4 12.44%	Table 10 [DDH+21]
128-256	16	$2^{-7.6} - 2^{-6.2}$	127.6	127.2-125.2 9.30%	125.1-122.1 31.30%	122.0-117.0 35.67%	116.9-114.9 17.95%	114.8-108.2 5.78%	Table 11 [DDH+21]
128-384	17	$2^{-8.37} - 2^{-6.21}$	110	108.2-105.4 19.98%	105.3-104.3 23.85%	104.2-103.2 21.84%	103.1-102.1 19.02%	102.0-98.6 15.31%	Figure 6 [NGJE25]

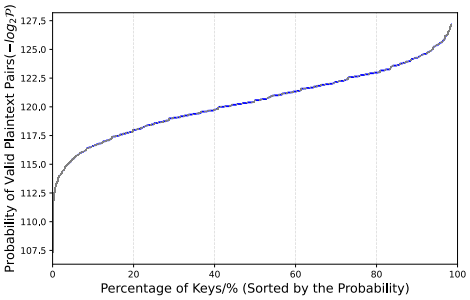
Stated Prob.: the $-log_2$ probability reported in the original papers; Reduced key Space: refers to the proportion of the estimated valid key space for the differential trail.

distribution. For the 14-round SKINNY-128-128 trail, we integrated the key scheduling into the solving phase and solved all linear and solvable nonlinear constraints using the accurate method. However, for the 16-round SKINNY-128-256 trail, when we incorporated key scheduling into solving, the complexity increases drastically, thereby only linear constraints and three nonlinear constraints remain solvable in this case. And for the 17-round SKINNY-128-384 trail, the Trail-Estimator exhaustively solved 13 linear constraints and one simpler nonlinear constraint (Nonlinear Constraint 1 in Table 23). For the Nonlinear Constraint 2, Trail-Estimator estimated its key space and probability distribution using the estimation method with 2^{10} random master keys and 2^{34} sampled x variables. Based on solvable constraints, Trail-Estimator provided, for the first time, a comprehensive prediction of the probability distribution for SKINNY-128 differential trails.

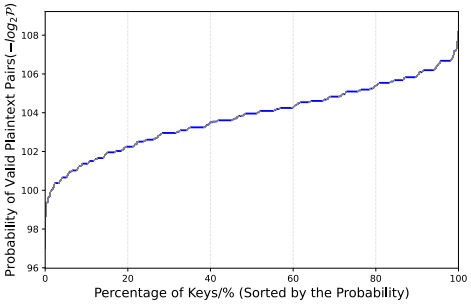
Comparison with previous works. We compare our results for SKINNY with previous studies and find that Trail-Estimator consistently identifies more nonlinear constraints. In particular, Trail-Estimator discovers a significant number of nonlinear constraints that span over five or more rounds within a differential trail. Some of these extended nonlinear constraints span up to 14 rounds, which have not been 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 overlook 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 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 constraint, which captures only certain linear relationships between specific bits while overlooking nonlinear dependencies in the trail. In contrast, Trail-Estimator identifies 2 new nonlinear constraints, as shown in Appendix K. 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]. The AutoDiVer tool in [NGJE25] can derive necessary linear and nonlinear key



(a) Distribution of 14-round trail [DDH+21]



(b) Distribution of 16-round trail [DDH+21]



(c) Distribution of 17-round trail [NGJE25]

Figure 6: Predicted probability distributions of SKINNY-128 differential trails

conditions, while it overlooks some nonlinear constraints. In contrast, Trail-Estimator extracts all potential linear and nonlinear constraint subsets within the cipher, offering a more complete description of the key-related conditions. Concerning execution time, the AutoDiVer tool can rapidly estimate the size of the valid key space, whereas the Trail-Estimator requires more time to generate a complete probability distribution. A key advantage of Trail-Estimator is its ability to identify all the constraints within seconds for word-oriented block ciphers. In our experiments, the key space estimations provided by Trail-Estimator for the 15-round SKINNY-64-192 and the 17-round SKINNY-128-384 trails are in close agreement with those reported in [NGJE25], demonstrating consistency with prior findings.

Results verification. We extracted the first three rounds of the differential trail from Table 6 of [DDH+21] and denoted it as T_{S_0}. We chose to verify this particular segment, as it contains a short nonlinear constraint that was not detected in previous work. This extra constraint is shown in Appendix K.2. The probability for this differential trail under the Markov cipher assumption is 2^{-32}. A step-by-step illustration of how the Cons-Collector finds the constraint subsets of T_{S_0} is given in Appendix G. In Figure 7, we show the graph of the probability distribution derived from experimentation (in red) and compare it with the predicted distribution (in blue) 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 distributions, roughly at key number 670. We attribute this discrepancy to the small sample size of master keys. The details of the Monte Carlo simulation are shown in Appendix I.

Notably, the nonlinear constraint in T_{S_0} not only reduces the valid key space but also reshapes the probability distribution. This effect arises from the nonlinear component $S(\cdot)$ within the nonlinear constraint. Specifically, when solving the nonlinear equation: $S([y_{12}^0] \oplus [x_{15}^1]) \oplus [y_9^1] \oplus [x_{15}^2] \oplus k_{13} = 0$, the resulting values for k_{13} uniformly belong to the

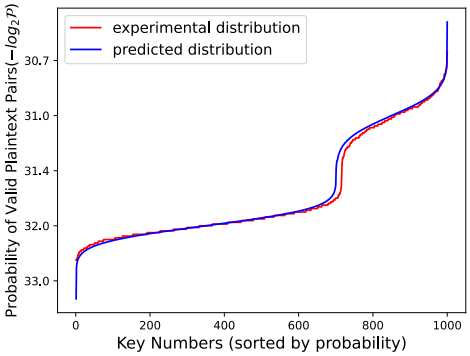


Figure 7: Predicted and experimental probability distribution of the SKINNY-64 trail T_{S_0}

multiset $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 8, 9, 9, 12, 12, 13, 13\}$. This leads to a non-uniform distribution for k_{13}. Additionally, for most differential trails involving nonlinear constraints, the actual probability distribution is expected to deviate from uniformity.

5.2 Application to LBLOCK

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

For the 16-round differential trail T_{LBLOCK}, Trail-Estimator found 8 linear and 11 nonlinear constraints (as shown in Table 25). Cons-Solver then computed the predicted probability distribution, which is shown in Figure 13 in Appendix L.2. 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} stated in [ZZDX19], due to the effect of a linear constraint. Meanwhile, the presence of nonlinear constraints results in a non-uniform probability distribution.

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 14. The first trail, denoted as T_{L_0}, is extracted from the first four rounds of the optimal 16-round trail in [ZZDX19]. The second trail, denoted as T_{L_1}, is derived by modifying the output difference of the 6th nibble of the final round in T_{L_0}. The details of two trails are presented in Table 10 and Table 11 in Appendix J.2, respectively. For T_{L_0} and T_{L_1}, Cons-Collector identified the same nonlinear constraint, as detailed in Appendix K.8. Since the active S-box positions remain identical in both trails, their nonlinear constraint structures are also the same, potentially influencing the probability distributions of T_{L_0} and T_{L_1}. 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_{L_0}, the calculation of Cons-Solver shows that the nonlinear constraint does not affect the original probability distribution. According to Figure 14a in Appendix L.3, the experimental probability distribution of the trail aligns with predicted results, following a Gaussian distribution with an average probability of approximately 2^{-18}. For T_{L_1}, although it differs from T_{L_0} by only a single nibble, the constraint has a significantly different impact on its probability distribution. According to Cons-Solver, T_{L_1} 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 14b in Appendix L.3, 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 for LBLOCK, it might introduce some imprecision. To validate this, we analyzed LBLOCK with fully independent round keys as well. As shown in Figure 15, the predicted and experimental probability distributions align closely, with only a gap of less than 2% between them.

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_{L_0} and T_{L_1}, we observe that nonlinear constraints with the 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 was applied to two differential trails of TWINE: an optimal 9-round trail discovered by the Mixed Integer Linear Programming (MILP) solver (detailed in Table 13 in Appendix J.3) and an optimal 15-round differential trail from Table 16 in [ZZDX19]. This study is the first to perform differential trail verification for TWINE.

For the 9-round differential trail in Table 13, denoted as T_{TW_1}, Trail-Estimator identified three nonlinear constraints but no linear constraint, as shown in Table 26 in Appendix K.10. All three nonlinear constraints are within a solvable range. The predicted probability distribution for the trail is shown in Figure 16a in Appendix L.4. Under the Markov cipher assumption, this trail has a probability of 2^{-28}. In the experiment, the Cons-Solver applied the estimation method to constraints using 2^{10} randomly sampled master keys. The resulting average probability of the trail within valid key space is 2^{-28}, with a maximum of $2^{-26.9}$ and a minimum of $2^{-30.3}$. And the valid key space size range of confidence level 95% is $[0.99, 1]$, which means at least over 99% keys are valid with a confidence level of 95%. The overall effect of three nonlinear constraints is highly complex, and the final probability distribution follows closely the Gaussian distribution.

For the 15-round differential trail from [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, as shown in Table 27, Appendix K.10. Based on these 4 nonlinear constraints and all linear constraints, the predicted probability distribution is shown in Figure 16b. The nonlinear constraints in T_{TW_2} are simpler than those in T_{TW_1}, resulting in the probability distribution remaining Gaussian-like but with greater discontinuities in the curve. Additionally, based on our probability distribution, the average probability for valid key space is $2^{-64.3}$, which is 16 times higher than the probability stated in [ZZDX19]. This discrepancy is mainly 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}.

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, 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, LBLOCK, TWINE, and AES 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, increasing the average probability of the trail within the valid key space. On the other hand, nonlinear constraints introduce greater complexity into the key distribution, generally reshaping the probability distribution of the trail. Additionally, as the number and complexity of constraints increase, the probability distribution progressively resembles a Gaussian distribution.

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

3. The results presented in Table 2 indicate that, for the majority of differential trails, the average probability within the valid key space exceeds the originally stated probability. Consequently, these differential trails may serve as effective components in weak key attacks, potentially reducing the required data complexity of cryptanalysis.

4. For ciphers with word-wise key schedules, incorporating them into our framework ensures that probability predictions closely match actual values. Otherwise, we observed that assuming independent round keys leads 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`). 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 authors are grateful to the anonymous reviewers for their detailed and insightful comments that improved the quality of the paper. 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.

[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.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael*, volume 2. Springer, 2002.

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

[Hey20] Howard M. Heys. Key dependency of differentials: Experiments in the differential cryptanalysis of block ciphers using small s-boxes. *IACR Cryptol. ePrint Arch.*, page 1349, 2020.

[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.

[Wil27] Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[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

This section shows the Gaussian Elimination-based algorithm used in Cons-Collector. The Algorithm 1 is used to search for minimal constraint subsets within word-oriented block ciphers using a $GF(2)$ linear layer.

In Algorithm 1, function FindPivot(MAT, r) is used to find the index of first nonzero-element in row MAT[r]. And function REF(MAT, ROW_IND_MAT, cur_row) keeps matrix MAT in the row echelon form.

Algorithm 1 Gauss-Collector Algorithm

Input: A binary matrix $\texttt{MAT} \in \mathbb{F}_2^{M \times N}$ representing all equations in a block cipher;
Output: A list ROW_IND_MAT containing the lists of indices of linearly dependent rows retained from the matrix MAT;

Initialize $M \leftarrow$ number of rows in MAT
Initialize ROW_IND_MAT $\leftarrow [\emptyset, \ldots, \emptyset]$ of size M
for $i = 0$ to $M - 1$ **do**
    ROW_IND_MAT$[i] \leftarrow \{i\}$
end for
cur_row $\leftarrow 1$
while cur_row $< M$ **do**
    while FindPivot(MAT, cur_row) $\leq$ FindPivot(MAT, cur_row $- 1$) **do**
        MAT[cur_row] $\leftarrow$ MAT[cur_row] $\oplus$ MAT[cur_row $- 1$]
        ROW_IND_MAT[cur_row] $\leftarrow$ ROW_IND_MAT[cur_row $- 1$] $\cup$ ROW_IND_MAT[cur_row]
        REF(MAT, ROW_IND_MAT, cur_row)
    end while
    cur_row $\leftarrow$ cur_row $+ 1$
end while
return ROW_IND_MAT

We also design an Algorithm 2 for ciphers with a $GF(2^8)$ linear layer. For the Gauss-Collector algorithm in $GF(2^8)$, the function UnifyRow(mat,r) is used to normalize the row mat[r]'s first nonzero element in $GF(2^8)$ and returns the normalization coefficient b. Meanwhile, the function RowMul(row, r, b) is used to apply scale multiplication $row[r] \cdot b$ in $GF(2^8)$. Check() function is used to return the final list of linearly dependent row sets after Gaussian elimination in $GF(2^8)$. The matrix row_rec is used to record the coefficients of each row of linearly dependent row sets.

Algorithm 2 Gauss-Collector Algorithm over $GF(2^8)$

Input: A $M \times N$ matrix `MAT` over $GF(2^8)$, representing equations in AES
Output: A list `ROW_IND` of indices of linearly dependent rows in `MAT`

```
mat ← copy of matrix MAT
(rows, cols) ← shape of mat
row_idx ← 0
row_rec ← identity matrix of size rows × rows
for col = 0 to cols − 1 do
    pivot_row ← None
    for r = row_idx to rows − 1 do
        if mat[r][col] ≠ 0 then
            pivot_row ← r
            break
        end if
    end for
    if pivot_row is None then
        continue
    end if
    b ← UnifyRow(mat, pivot_row)                    ▷ Normalize pivot row
    RowMul(row_rec, pivot_row, b)
    Swap mat[row_idx] ↔ mat[pivot_row]
    Swap row_rec[row_idx] ↔ row_rec[pivot_row]
    for r = row_idx + 1 to rows − 1 do
        if mat[r][col] ≠ 0 then
            b ← UnifyRow(mat, r)
            mat[r] ← mat[r] ⊕ mat[row_idx]
            RowMul(row_rec, r, b)
            row_rec[r] ← row_rec[r] ⊕ row_rec[row_idx]
        end if
    end for
    row_idx ← row_idx + 1
end for
ROW_IND ← Check(mat, row_rec)
return ROW_IND
```

B Proof for Corollary 1

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 and $\mathbb{E}$ can still be a minimal constraint on key variables $\{k_0, k_1, \cdots, k_J\}$. And nonlinear equations are all of the form: $y_u = S(x_u)$. Therefore, we have:

$$\sum_i e_i = \sum_j k_j \oplus \sum_u (S(x_u) \oplus x_u) \oplus \sum_h v_h \oplus F = 0,$$

Here, F is a free variable, which cannot be included in $\sum v_h$ and cannot pair with any x_u or y_u in $\sum_u (S(x_u) \oplus x_u)$. 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. □

C Wilson Score Interval

The Wilson score interval estimates confidence intervals for proportions (e.g., success rates) based on the binomial distribution. It offers key improvements over the normal approximation by remaining reliable even with small sample sizes or skewed data. When applying the estimation method to compute the probability distribution of the differential trail, `Cons-Solver` adopts the Wilson score interval to compute the upper and lower bounds of the key space reduction ratio K_r. Let n denote the number of samples, and let $\hat{k}$ represent the observed average key space reduction ratio. The Wilson score interval is then given by:

$$K_r \in (k_-, k_+) = (1 + \frac{z^2}{n})^{-1} \cdot (\hat{k} + \frac{z^2}{2n} \pm \frac{z}{2n} \cdot \sqrt{4n\hat{k}(1-\hat{k}) + z^2})$$

For a 95% confidence level, the corresponding z-score is $z = 1.96$. In an experiment, the confidence level can be adjusted according to the specific situation. Meanwhile, for `Cons-Solver`, the number of key variable samples is $n = 2^{m_2}$.

D Time Consumption of `Cons-Collector` and `Cons-Solver`

In this section, we present the time consumption of `Cons-Collector` and `Cons-Solver` for all differential trail experiments. All experiments were performed on a laptop equipped

with an Intel Core i9-12900H 2.50 GHz processor, utilizing a single CPU core throughout the evaluations.

Table 4: Experimental time consumption for SKINNY, LBLOCK, TWINE and AES

Version	Rds	Reduced key space	Time Consumption Cons-Collector	Cons-Solver	Source
SKINNY	7	$2^{-7.3}$	2.1 s	204 s	Table 6 [DDH+21]
	10	0	3.3 s	0.8 s	Table 7 [DDH+21]
	10	1	3.1 s	1.3 s	Table 7 [PT22]
	13	2^{-4}	4.6 s	5.2 s	Table 8 [DDH+21]
	8	1	2.7 s	3.9 s	Table 16 [QDW+21]
	15	$2^{-6.2}$	9.5 s	2.1 s	Table 9 [DDH+21]
	11	0	7.8 s	20.6 s	Figure 2 [ZZ18]
	14	$2^{-7.7}$	9.2 s	4.5 h	Table 10 [DDH+21]
	16	$2^{-7.6} - 2^{-6.2}$	9.5 s	10 h	Table 11 [DDH+21]
	17	$2^{-8.37} - 2^{-6.21}$	10.7 s	7 h	Figure 6 [NGJE25]
LBLOCK	4	1	0.72 s	55 s	Table 10 Appendix J.2
	16	2^{-1}	4.8 s	6 h	Table 14 [ZZDX19]
TWINE	9	$0.99 - 1$	3.6 s	11 h	Table 13 Appendix J.3
	15	$2^{-4.7}$	6.2 s	4.5 h	Table 16 [ZZDX19]
AES	3	2^{-20}	1 s	10 s	Table 16 Appendix J.4
	2	2^{-4}	0.8 s	3.4 s	Table 14 Appendix J.4
	2	1	0.8 s	1.8 s	Table 15 Appendix J.4

Reduced key Space: refers to the proportion of the estimated valid key space for the differential trail.

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

In the following table, the uniformity of $S(x) \oplus x$ in different ciphers is 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, TWINE and AES 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 5: 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
AES	163	256	Non-Uniform

F 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 combines the linear equations, which have no impact on any variables, within the same linear layers. Thereby, the first two equations within the nonlinear constraint $\mathbb{E}_3$ are XORed with each other. Then the nonlinear constraints $\mathbb{E}_1$ and $\mathbb{E}_2$ are grouped together as they are related to each other.

Table 6: 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 7.

Table 7: Preprocessed constraint subsets from Table 6

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

G 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 8.

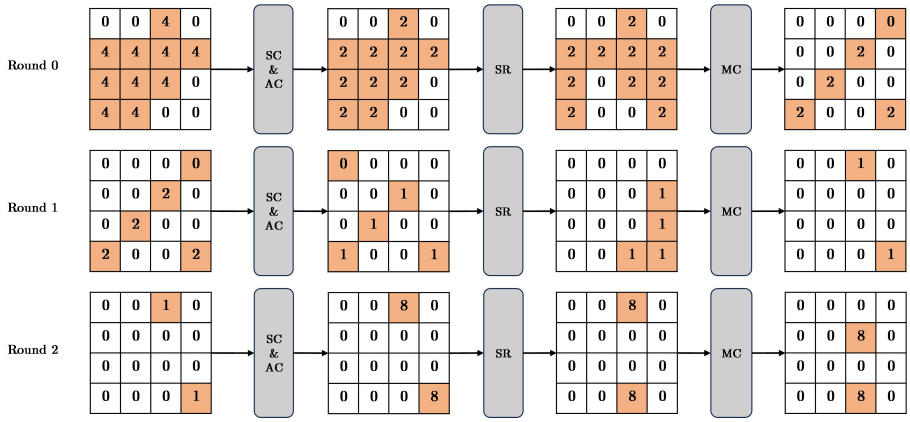


Figure 8: The first 3 rounds of the differential trail from Table 6 in [DDH+21], cells with orange color are active

The `Cons-Collector` first generalizes 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 the 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 3.5, 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 the differential trail. Based on this constraint, the valid space of key variable k_{13} is:

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

H Application to AES

AES is a block cipher proposed by Daemen and Rijmen [DR02] and standardized by the U.S. National Institute of Standards and Technology (NIST) in 2001. To support AES, we extend the Gaussian elimination-based Algorithm 1 to handle the $GF(2^8)$ linear layers. Cons-Collector adopt the modified algorithm, which is detailed in Algorithm 2 in Appendix A, to analyze differential trails of AES.

In our experiment, we first applied Trail-Estimator to the 2-round AES-128 trail T_{AES0} as shown in Table 14 in Appendix J.4. The Cons-Collector identified a composite linear constraint formed by four individual linear equations within T_{AES0}. The details of this linear constraint are provided in Table 28 in Appendix K.11. This constraint imposes restrictions on the key variables $(k_0^0, k_4^0, k_8^0, k_{12}^0)$, resulting in only a fraction 2^{-4} of the master keys being valid. The Cons-Solver solved the constraint and calculated the trail probability to be 2^{-31} for valid keys. We also find out this trail is a plateau characteristic with a weight of 35 and a height of 1, consistent with the findings shown in [DR07].

We further applied Trail-Estimator to the 2-round trail T_{AES1}, which has a stated probability of 2^{-30} and shown in Table 15 in Appendix J.4. The Cons-Collector identifies a single linear constraint, which has the same form as that of E_{AES0}, but with different values for the variables (detailed in Table 28 in Appendix K.11), and the Cons-Solver shows that this constraint imposes no restrictions on the values of the key variables. Consequently, the AES-128 trail T_{AES1} is also a plateau characteristic, with a weight of 30 and a height of 2, consistent with results presented in [DR07].

For the 3-round AES-128 trail T_{AES2} as shown in Table 16 in Appendix J.4, the Cons-Collector identified 5 linear constraints, as shown in Table 29 in Appendix K.11. The Cons-Solver then calculated the valid key space size based on these constraints, and found that only 2^{-20} keys are valid. This result was also verified by the estimation probability calculation method, where we sampled 2^{30} random master keys and integrated the key scheduling into the solving phase. Among these, only 1011 keys satisfy the imposed constraints, indicating that merely a fraction of approximately $2^{-20.02}$ of master keys are valid.

I Details of the Monte-Carlo Simulation

We run a Monte Carlo simulation on the predicted distribution for the trail in our paper. Suppose the frequency F of right plaintext pairs follows a Binomial distribution: $F \sim Binomial(N, p)$, where N represents the total number of plaintext pairs and p is the probability of the differential trail. By the Central Limit Theorem, this Binomial distribution can be approximated by a Gaussian distribution: $F \sim \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(Np, Np(1-p))$. Since p is much smaller than 1, the distribution of F can be further approximated as $\mathcal{N}(Np, Np)$. Assume that we have f right plaintext pairs when choosing a master key, the corresponding predicted probability can be computed by $P = f/N$.

To validate the accuracy of our prediction, we randomly selected 1000 valid master keys. For each master key, we generated $N = 2^{31} - 2^{38}$ pairs of plaintexts to test the probability under each key, depending on the details of the trails. Furthermore, due to the computational limitation, we were only able to perform Monte Carlo simulations for differential trails with probabilities greater than 2^{-38}.

J Differential Trails

J.1 Differential Trails of SKINNY-64

In the verification experiment, the SKINNY-64 trail T_{S_0} is the first 3-round extracted from the trail in Table 6 [DDH+21]. The differential trail is shown in Table 8:

Table 8: 3-round SKINNY-64 differential trail T_{S_0} derived from Table 6 [DDH+21]

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

J.2 Differential Trails of LBLOCK

For LBLOCK, the trail T_{L_0} is the first 4 rounds extracted from the 16-round trail in [ZZDX19], as shown in Table 10. Meanwhile, the trail T_{L_1} in Table 11 has the totally same structure as T_{L_0}, with only the 6-th output nibble value different. The original 16-round trail T_{LBLOCK} is shown in Table 12.

Table 10: 4-round differential trail of LBLOCK: T_{L_0}

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

Table 11: 4-round differential trail of LBLOCK: T_{L_1}

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

Table 12: 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

J.3 Differential Trail of TWINE

Table 13 shows the optimal 9-round differential trail for TWINE, which is generated using the MILP solver.

Table 13: 9-Round differential trail T_{TW_1} 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

J.4 Differential Trail of AES-128

In this section, we present the detailed information for AES-128 trails T_{AES_0}, T_{AES_1} and T_{AES_2}. T_{AES_0} and T_{AES_1} are detailed in Table 14 and Table 15, respectively, and both are also depicted in Figure 9. T_{AES_2} is presented in Table 16, and depicted in Figure 10

Table 14: 2-Round differential trail T_{AES_0} of AES with probability of 2^{-35}

Round	ΔI_s	ΔO_s
0	01000000000000000000000000000000	01000000000000000000000000000000
1	02000000010000000100000003000000	47000000630000000C20000008A000000

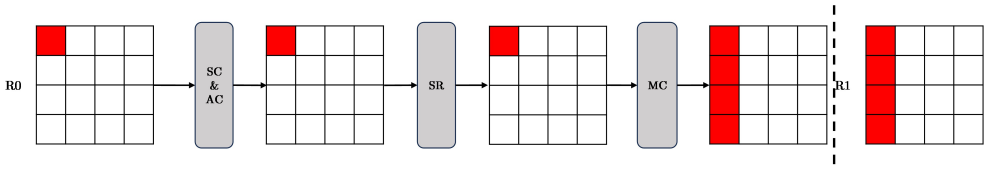


Figure 9: The structure of 2-round AES differential trail T_{AES_0} and T_{AES_1}

Table 15: 2-Round differential trail T_{AES_1} of AES with probability of 2^{-30}

Round	ΔI_s	ΔO_s
0	01000000000000000000000000000000	1F000000000000000000000000000000
1	3E0000001F0000001F00000021000000	D1000000A3000000A30000009E000000

Table 16: 3-Round differential trail T_{AES_2} of AES with probability of 2^{-147}

Round	ΔI_s	ΔO_s
0	01000000000000000000000000000000	48000000000000000000000000000000
1	90000000480000004800000d8000000	79000000040000000E400000081000000
2	F281E40C798137087998D3048B19E404	D7F7418405760458079292E5D92C56

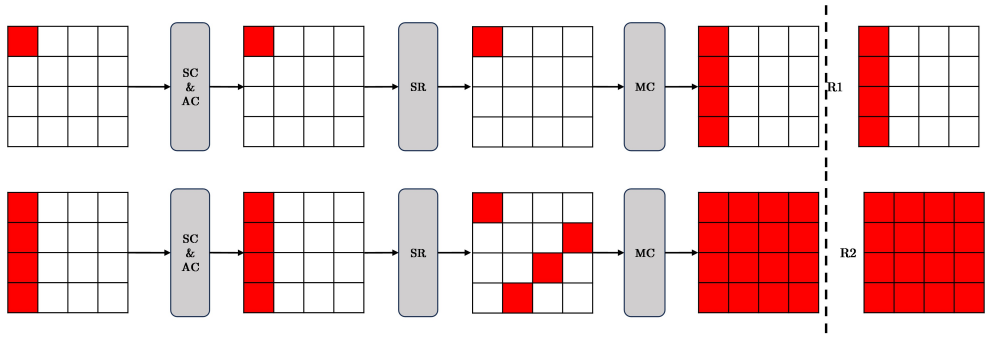


Figure 10: The structure of 3-round AES differential trail T_{AES_2}

K Important Nonlinear and Linear Constraints in Differential Trails

In this section, we show important nonlinear and linear constraints in differential trails of different ciphers. In these constraints, k_i^r stands for the i-th subkey in round r, x_i^r and y_i^r are the input and output variable of ith Sbox in round r

K.1 Infeasible Nonlinear and Linear Constraints of 3-round SKINNY-128 Trail from Figure 2 [ZZ18]

In this part, two nonlinear constraint subsets and one linear constraint subset within 3-5 rounds 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 17: Infeasible nonlinear constraints and linear constraint within Figure 2 [ZZ18]

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

K.2 Nonlinear Constraints Found in First 3 Rounds of SKINNY-64 Trail in 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 18: Nonlinear constraint within first 3 rounds of the trail in Table 6 [DDH+21]

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

K.3 Other Solvable Nonlinear Constraints of SKINNY-64 Trail from Table 6 [DDH+21]

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

Table 19: Nonlinear constraints within Table 6 [DDH+21]

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

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

In Table 20, a linear constraint that causes infeasibility in the 10-round trail from Table 7 [DDH+21].

Table 20: Infeasible linear constraint within Table 7 [DDH+21]

Infeasible Linear Constraint
$y_0^0 \oplus [y_{10}^0] \oplus [y_{13}^0] \oplus [x_0^1] \oplus k_0 = 0$
$y_0^0 \oplus [x_4^1] \oplus k_0 = 0$

K.5 New Solvable Nonlinear Constraint in 14-round SKINNY-128 Trail

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

Table 21: Nonlinear constraint $\mathbb{E}_7$

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

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

Two new solvable nonlinear constraints are detected in the 16-round `SKINNY-128` trail, as shown in Table 22.

Table 22: New solvable nonlinear constraints in Table 11 [DDH+21]

Nonlinear Constraint $\mathbb{E}_8$	**Nonlinear Constraint** $\mathbb{E}_9$
$[y_1^2] \oplus y_{11}^2 \oplus [y_{14}^2] \oplus x_1^3 \oplus k_1^2 = 0$	$y_0^3 \oplus y_{10}^3 \oplus [y_{13}^3] \oplus [x_0^4] \oplus k_0^3 = 0$
$[y_1^2] \oplus y_{11}^2 \oplus [x_{13}^3] \oplus k_1^2 = 0$	$y_0^3 \oplus y_{10}^3 \oplus x_{12}^4 \oplus k_0^3 = 0$
$y_1^3 \oplus [y_{11}^3] \oplus [x_{13}^4] \oplus k_1^3 = 0$	$[y_3^4] \oplus [y_9^4] \oplus y_{12}^4 \oplus [x_3^5] \oplus k_3^4 = 0$
$S(x_1^3) \oplus y_1^3 = 0$	$S(x_{12}^4) \oplus y_{12}^4 = 0$

K.7 Linear and Solvable Nonlinear Constraints of SKINNY-128 trail in Figure 6 [NGJE25]

The 17-round trail in [DDH+21] is invalid due to a typo. For the 6th 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. In Figure 6 [NGJE25], this typo is corrected, and we applied the `Trail-Estimator` to this modified trail.

And `Trail-Estimator` identified 7 new solvable nonlinear constraints. The interdependent constraints are merged into two nonlinear constraints. And the final linear and nonlinear constraints are shown below.

Table 23: Linear and solvable nonlinear constraint subsets for the 17-round SKINNY-128 trail in Figure 6 ([NGJE25]).

13 Linear Constraints	Nonlinear Constraint 1	Nonlinear Constraint 2
$[y_1^6] \oplus [y_{11}^6] \oplus [x_{13}^7] \oplus k_1^6 = 0$	$[y_1^1] \oplus y_{11}^1 \oplus [y_{14}^1] \oplus x_1^2 \oplus k_1^1 = 0$	$[y_7^3] \oplus [y_{10}^3] \oplus x_8^4 \oplus k_7^3 = 0$
$[y_0^5] \oplus [x_4^6] \oplus k_0^5 = 0$	$[y_1^1] \oplus y_{11}^1 \oplus [x_{13}^2] \oplus k_1^1 = 0$	$[y_5^4] \oplus y_8^4 \oplus [x_{10}^5] \oplus k_5^4 = 0$
$[y_3^4] \oplus [x_7^5] \oplus k_3^4 = 0$	$y_1^2 \oplus [y_{11}^2] \oplus [x_{13}^3] \oplus k_1^2 = 0$	$[y_2^4] \oplus y_8^4 \oplus [x_{14}^5] \oplus k_2^4$
$[y_2^4] \oplus [x_6^5] \oplus k_2^4 = 0$	$S(x_1^2) = y_1^2$	$[y_2^4] \oplus y_8^4 \oplus [y_{15}^4] \oplus x_2^5 \oplus k_2^4 = 0$
$xy_1^4] \oplus [x_5^5] \oplus k_1^4 = 0$		$y_2^5 \oplus [y_8^5] \oplus [x_{14}^6] \oplus k_2^5 = 0$
$[y_3^3] \oplus [y_9^3] \oplus [x_{15}^4] \oplus k_3^3 = 0$		$S(x_8^4) = y_8^4$
$[y_3^3] \oplus [x_7^4] \oplus k_3^3 = 0$		$S(x_2^5) = y_2^5$
$[y_1^3] \oplus [x_5^4] \oplus k_1^3 = 0$		
$[y_0^3] \oplus [y_{10}^3] \oplus [x_{12}^4] \oplus k_0^3 = 0$		
$[y_0^3] \oplus [x_4^4] \oplus k_0^3 = 0$		
$[y_1^1] \oplus [x_5^2] \oplus k_1^1 = 0$		
$[y_2^0] \oplus [y_8^0] \oplus [x_{14}^1] \oplus k_2^0 = 0$		
$[y_2^0] \oplus [x_6^1] \oplus k_2^0 = 0$		

K.8 The Only Constraint in LBLOCK Trail T_{L_0} and T_{L_1}

Since the active cell positions in T_{L_0} and T_{L_1} are all the same, so these two trails have the same nonlinear constraint structure. However, the value of the constrained variable x_6^3 is different, which results in totally different probability distributions for these two trails.

Table 24: The only nonlinear constraint found in T_{L_0} and T_{L_1}

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

K.9 Solvable Nonlinear and Linear Constraint of 16-round LBLOCK 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 25.

K.10 Solvable Nonlinear and Linear Constraints in TWINE

In Table 26 and Table 27, we show the solvable nonlinear minimal constraint subsets identified in TWINE. For T_{TW_1}, we identified 3 long constraints, all of which span over 10 rounds. While in T_{TW_2}, there are 4 nonlinear constraints and 4 linear constraints which can be solved by Cons-Solver.

Table 25: 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** $\mathbb{E}_{L3}$
$[x_4^1] \oplus x_{12}^2 \oplus k_4^1 = 0$	$[x_2^7] \oplus x_{10}^8 \oplus k_2^7 = 0$	$[x_4^{10}] \oplus x_{12}^{11} \oplus k_4^{10} = 0$
$x_{12}^2 \oplus y_4^2 \oplus [x_6^3] \oplus k_6^2 = 0$	$x_{10}^8 \oplus [y_5^8] \oplus x_4^9 \oplus k_4^8 = 0$	$x_{12}^{11} \oplus [y_4^{11}] \oplus x_6^{12} \oplus k_6^{11} = 0$
$x_4^2 \oplus x_{12}^3 \oplus k_4^2 = 0$	$x_4^9 \oplus x_{12}^{10} \oplus k_4^9 = 0$	$[x_5^{11}] \oplus x_{13}^{12} \oplus k_5^{11} = 0$
$x_{12}^3 \oplus [y_4^3] \oplus [x_6^4] \oplus k_6^3 = 0$	$x_{12}^{10} \oplus [y_4^{10}] \oplus [x_6^{11}] \oplus k_6^{10} = 0$	$x_{13}^{12} \oplus y_6^{12} \oplus [x_7^{13}] \oplus k_7^{12} = 0$
$S(x_4^2) \oplus y_4^2 = 0$		$S(x_6^{12}) = y_6^{12}$
Nonlinear Constraint $\mathbb{E}_{L4}$	**Nonlinear Constraint** $\mathbb{E}_{L5}$	
$[x_7^7] \oplus x_{15}^8 \oplus k_7^7 = 0$	$[x_5^2] \oplus x_{13}^3 \oplus k_5^2 = 0$	
$x_{15}^8 \oplus [y_3^8] \oplus x_1^9 \oplus k_1^8 = 0$	$[x_7^2] \oplus x_{15}^3 \oplus k_7^2 = 0$	
$x_1^9 \oplus x_9^{10} \oplus k_1^9 = 0$	$x_{15}^3 \oplus y_3^3 \oplus [x_1^4] \oplus k_1^3 = 0$	
$x_9^{10} \oplus y_2^{10} \oplus x_3^{11} \oplus k_3^{10} = 0$	$x_{13}^3 \oplus [y_6^3] \oplus x_7^4 \oplus k_7^3 = 0$	
$x_2^{10} \oplus x_{10}^{11} \oplus k_2^{10} = 0$	$x_3^3 \oplus x_{11}^4 \oplus k_3^3 = 0$	
$[x_7^{10}] \oplus x_{15}^{11} \oplus k_7^{10} = 0$	$x_{11}^4 \oplus y_7^4 \oplus x_5^5 \oplus k_5^4 = 0$	
$x_1^{11}5 \oplus y_3^{11} \oplus [x_1^{12}] \oplus k_1^{11} = 0$	$x_5^5 \oplus x_{13}^6 \oplus k_5^5 = 0$	
$x_{10}^{11} \oplus [y_5^{11}] \oplus [x_4^{12}] \oplus k_4^{11} = 0$	$x_{13}^6 \oplus [y_6^6] \oplus [x_7^7] \oplus k_7^6 = 0$	
$S(x_2^{10}) \oplus y_2^{10} = 0$	$S(x_3^3) \oplus y_3^3 = 0$	
$S(x_3^{11}) \oplus y_3^{11} = 0$	$S(x_7^4) \oplus y_7^4 = 0$	

Table 26: Solvable nonlinear constraint subsets within T_{TW_1}

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

Table 27: Solvable nonlinear constraint subsets within T_{TW_2}

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

K.11 Identified Linear Constraints in AES-128

In this section, we show the linear constraints identified in AES-128 trails: T_{AES_0}, T_{AES_1}, and T_{AES_2}.

Table 28: The only linear constraint found in T_{AES_0} and T_{AES_1}

Linear Constraint E_{A0}
$2 \cdot [y_0^0] \oplus 3 \cdot y_5^0 \oplus 1 \cdot y_{10}^0 \oplus 1 \cdot y_{15}^0 \oplus 1 \cdot [x_0^1] \oplus 1 \cdot k_0^0 = 0$
$1 \cdot [y_0^0] \oplus 2 \cdot y_5^0 \oplus 3 \cdot y_{10}^0 \oplus 1 \cdot y_{15}^0 \oplus 1 \cdot [x_4^1] \oplus 1 \cdot k_4^0 = 0$
$1 \cdot [y_0^0] \oplus 1 \cdot y_5^0 \oplus 2 \cdot y_{10}^0 \oplus 3 \cdot y_{15}^0 \oplus 1 \cdot [x_8^1] \oplus 1 \cdot k_8^0 = 0$
$3 \cdot [y_0^0] \oplus 1 \cdot y_5^0 \oplus 1 \cdot y_{10}^0 \oplus 2 \cdot y_{15}^0 \oplus 1 \cdot [x_{12}^1] \oplus 1 \cdot k_{12}^0 = 0$
$\downarrow$
$23 \cdot (2 \cdot [y_0^0] \oplus 3 \cdot y_5^0 \oplus 1 \cdot y_{10}^0 \oplus 1 \cdot y_{15}^0 \oplus 1 \cdot [x_0^1] \oplus 1 \cdot k_0^0) = 0$
$47 \cdot (1 \cdot [y_0^0] \oplus 2 \cdot y_5^0 \oplus 3 \cdot y_{10}^0 \oplus 1 \cdot y_{15}^0 \oplus 1 \cdot [x_4^1] \oplus 1 \cdot k_4^0) = 0$
$246 \cdot (1 \cdot [y_0^0] \oplus 1 \cdot y_5^0 \oplus 2 \cdot y_{10}^0 \oplus 3 \cdot y_{15}^0 \oplus 1 \cdot [x_8^1] \oplus 1 \cdot k_8^0) = 0$
$145 \cdot (3 \cdot [y_0^0] \oplus 1 \cdot y_5^0 \oplus 1 \cdot y_{10}^0 \oplus 2 \cdot y_{15}^0 \oplus 1 \cdot [x_{12}^1] \oplus 1 \cdot k_{12}^0) = 0$
$\downarrow$
$95 \cdot [y_0^0] \oplus 23 \cdot [x_0^1] \oplus 47 \cdot [x_4^1] \oplus 246 \cdot [x_8^1] \oplus 145 \cdot [x_{12}^1] \oplus 23 \cdot k_0^0 \oplus 47 \cdot k_4^0 \oplus 246 \cdot k_8^0 \oplus 145 \cdot k_{12}^0 = 0$

Table 29: Linear constraints found in T_{AES2}

5 **Linear Constraints in** T_{AES_2}
$95 \cdot [y_0^0] \oplus 23 \cdot [x_0^1] \oplus 47 \cdot [x_4^1] \oplus 246 \cdot [x_8^1] \oplus 145 \cdot [x_{12}^1] \oplus 23 \cdot k_0^0 \oplus 47 \cdot k_4^0 \oplus 246 \cdot k_8^0 \oplus 145 \cdot k_{12}^0 = 0$
$64 \cdot [y_0^1] \oplus 173 \cdot [x_0^2] \oplus 246 \cdot [x_4^2] \oplus 109 \cdot [x_8^2] \oplus 118 \cdot [x_{12}^2] \oplus 173 \cdot k_0^1 \oplus 246 \cdot k_4^1 \oplus 109 \cdot k_8^1 \oplus 118 \cdot k_{12}^1 = 0$
$170 \cdot [y_{12}^1] \oplus 194 \cdot [x_1^2] \oplus 19 \cdot [x_5^2] \oplus 141 \cdot [x_9^2] \oplus 246 \cdot [x_{13}^2] \oplus 194 \cdot k_1^1 \oplus 19 \cdot k_5^1 \oplus 141 \cdot k_9^1 \oplus 246 \cdot k_{13}^1 = 0$
$96 \cdot [y_8^1] \oplus 214 \cdot [x_2^2] \oplus 77 \cdot [x_6^2] \oplus 118 \cdot [x_{10}^2] \oplus 141 \cdot [x_{14}^2] \oplus 214 \cdot k_2^1 \oplus 77 \cdot k_6^1 \oplus 118 \cdot k_{10}^1 \oplus 141 \cdot k_{14}^1 = 0$
$163 \cdot [y_4^1] \oplus 204 \cdot [x_3^2] \oplus 136 \cdot [x_7^2] \oplus 145 \cdot [x_{11}^2] \oplus 118 \cdot [x_{15}^2] \oplus 204 \cdot k_3^1 \oplus 136 \cdot k_7^1 \oplus 145 \cdot k_{11}^1 \oplus 118 \cdot k_{15}^1 = 0$

L Probability Distribution of Differential Trails

This section presents the detailed probability distribution figures for differential trails of different ciphers. In section L.3, we show the comparison between experimental and predicted probability distributions in the Monte-Carlo simulation of `LBLOCK` trails.

L.1 Probability Distribution of the `SKINNY-64` Trails

This section shows the probability distributions of the `SKINNY-64` trails. For the 7-round `SKINNY-64` trail from [DDH+21], the `Cons-Solver` calculated 3 solvable nonlinear constraints and 3 linear constraints, and got the resulting distribution as shown in Figure 11.

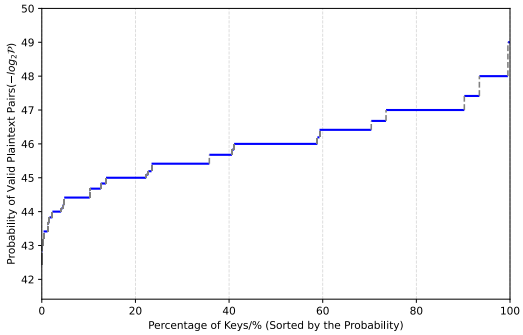


Figure 11: Predicted probability distribution of the 7-round `SKINNY-64` trail from [DDH+21]

For the 15-round `SKINNY-64` trail from [DDH+21], based on linear constraints and one solvable nonlinear constraint, the `Cons-Solver` predicted the probability distribution shown in Figure 12.

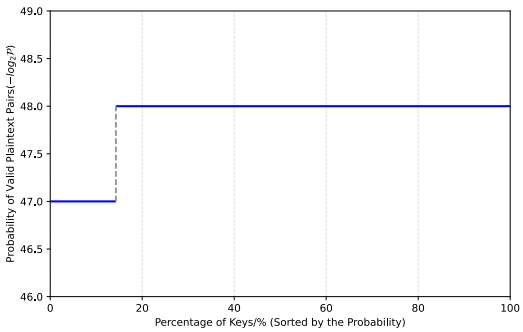


Figure 12: Predicted probability distribution of the 15-round `SKINNY-64` trail from [DDH+21]

L.2 Probability Distribution of LBLOCK Differential Trails

The probability distribution of LBLOCK trail T_{LBLOCK}, calculated by Cons-Solver using the accurate probability calculation method, is shown in Figure 13.

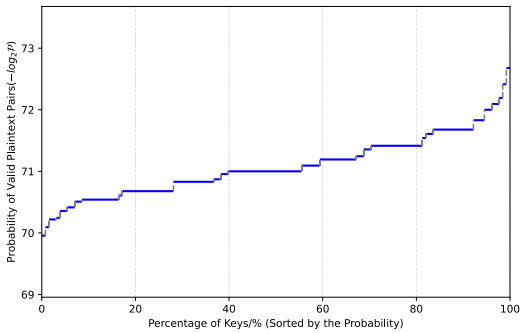
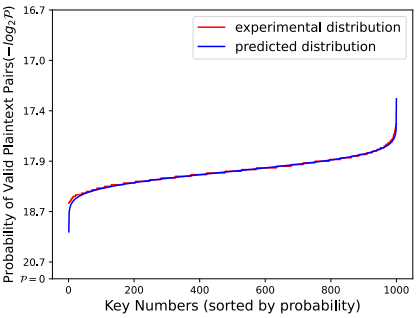


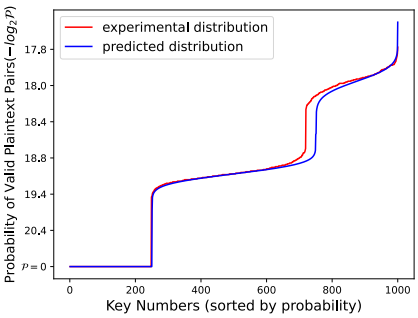
Figure 13: Predicted probability distribution of the 16-round LBLOCK trail from [ZZDX19]

L.3 Comparison between Experimental and Predicted Probability Distribution in the Monte-Carlo Simulation

Figure 14 shows the predicted and experimental probability distribution for the 4-round LBLOCK trails T_{L_0} and T_{L_1}. Due to the bit-wise key scheduling, we cannot represent the subkeys with master keys in a word-wise manner. Therefore, we assume the subkeys of LBLOCK are independent. In Figure 14b, a slight discrepancy appears around key number 700, which can be attributed to the effects of the key schedule. This discrepancy arises from the assumption that subkeys are mutually independent, introducing a degree of imprecision.



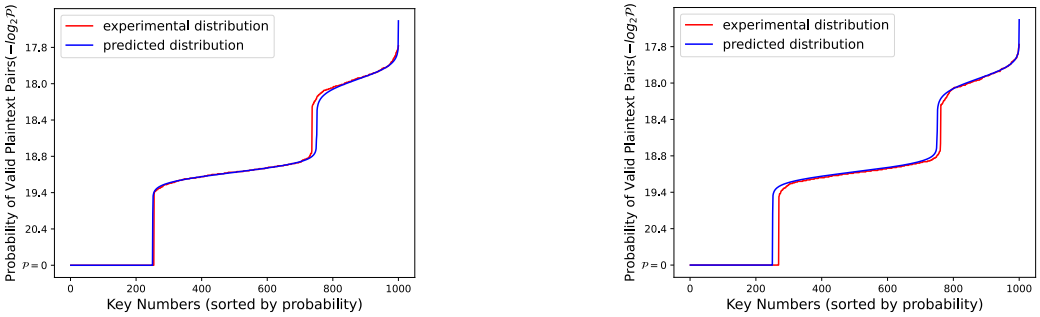
(a) Probability distribution of T_{L_0}



(b) Probability distribution of T_{L_1}

Figure 14: Predicted and experimental probability distributions of 4-round LBLOCK 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 the prediction and experimental probability distribution. In Figure 15a, the prediction is lower than the real probability around key number 750, while we got opposite results in Figure 15b, which shows the discrepancy is caused by randomness in master key selection.

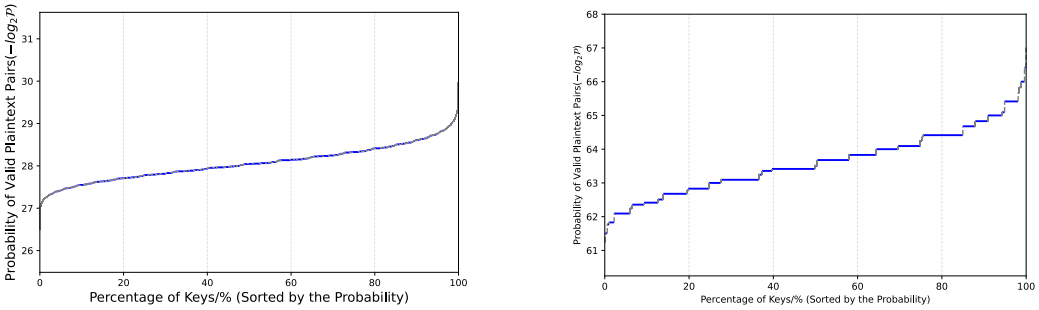


(a) Case 1 (b) Case 2

Figure 15: The predicted and experimental probability distribution of the 4-round trail T_{L_1} of LBLOCK without key scheduling

L.4 Probability Distribution of TWINE Differential Trails

For TWINE trails T_{TW_1} and T_{TW_2}, the detailed probability distributions are shown in Figure 16a and Figure 16b, respectively.



(a) Probability distribution of T_{TW_1} (b) Probability distribution of T_{TW_2}

Figure 16: Predicted probability distributions of two TWINE differential trails