# Improved Differential and Linear Cryptanalysis on Round-Reduced SIMON

Chao Niu[1,3], Muzhou Li[2,3,4,5], Jifu Zhang[1,3], Meiqin Wang[1,3,4*]

[1]School of Cyber Science and Technology, Shandong University, Qingdao, China .
[2]School of Cryptologic Science and Engineering, Shandong University, Jinan, China .
[3]Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China .
[4]State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao, 266237, China .
[5]Quan Cheng Laboratory, Jinan, 250103, China .

*Corresponding author(s). E-mail(s): mqwang@sdu.edu.cn;
Contributing authors: niuchao@mail.sdu.edu.cn;
muzhouli@mail.sdu.edu.cn; zhangjifu@mail.sdu.edu.cn;

**Abstract**

SIMON is a lightweight block cipher proposed by the National Security Agency. According to previous cryptanalytic results on SIMON, differential and linear cryptanalysis are the two most effective attacks on it. Usually, there are many trails sharing the same input and output differences (resp. masks). These trails comprise the differential (resp. linear hull) and can be used together when mounting attacks. In ASIACRYPT 2021, Leurent *et al.* proposed a matrix-based method on SIMON-like ciphers, where only trails whose active bits stay in a $\boldsymbol{w}$-bit window are considered. The static window in each round is chosen to be $\boldsymbol{w}$ least significant bits. They applied this efficient framework on SIMON and SIMECK, and have obtained many better differentials and linear hulls than before. For SIMON, they also found that there seems to be some potential for improvement, which should be further investigated.

In this paper, we dynamically choose window for each round to achieve better distinguishers. Benefiting from these dynamic windows, we can obtain stronger differentials and linear hulls than previously proposed for almost all versions

of SIMON. Finally, we provided the best differential/linear attacks on SIMON48, SIMON64, and SIMON96 in terms of round number, complexity, or success rate.

**Keywords:** SIMON, Dynamic Window, Clustering Effect, Differential Attack, Linear Attack.

# 1 Introduction

The SIMON [1] cipher is a suite of lightweight block ciphers proposed by the National Security Agency in 2013. Due to its simple round function and good performance in hardware and software, SIMON has gained a lot of attention since its proposal. It follows a Feistel structure with a very simple round function:

$$f(x) = (\mathrm{S}^8(x) \wedge \mathrm{S}^1(x)) \oplus \mathrm{S}^2(x),$$

where $\mathrm{S}^a(x)$ denotes the $a$-bit left cyclic rotation of $x$.

Previous cryptanalytic results show that the best attacks on it were differential cryptanalysis or linear cryptanalysis [2–8]. Meanwhile, there can be many trails in a differential or a linear hull that could be used to mount attacks together. However, the expected probability of the differential (EDP) and the expected linear potential (ELP) of the linear hull are both hard to evaluate due to the massive amount of trails comprising them.

In ASIACRYPT 2021, Leurent *et al.* [9] proposed a novel framework to deal with this for SIMON-like ciphers, where only trails whose input differences stay in the least $w$ significant bits in each round are considered. In this paper, we refer to it as the $w$-bit *static window* (SW). Unlike previous methods to enumerate the good trails contributing to the differential or the linear hull, they efficiently compute the probability distribution by multiplication of the differential transition matrix or linear correlation matrix round by round, which can consider *all* trails with active bits staying in the $w$-bit SW. Using this method, they found longer distinguishers that are more conducive to key recovery attacks and improved the previous best attack for several rounds for both SIMON and SIMECK [10].

## 1.1 Motivations and Contributions.

To include as many good trails as possible, Leurent *et al.* [9] adopted the same $w$-bit SW for all rounds. In other words, the diffusion of the included input and output differences (resp. masks) for each round are restricted. The SW is chosen to contain the least significant bits. With the increase of the window size $w$, the results obtained with SW will gradually approach the lower bounds of EDP and ELP. Eventually, they will become stable. Of course, the bigger $w$ leads to a tighter lower bound of EDP and ELP. However, the size of window $w$ is restricted by the memory size of the computer resources one can exploit. In [9], the maximum of $w$ is 19 and the matrix size will be $2^{19\times 2}$ as the memory size for their used computer is 1TB. They claimed that the ELP for SIMON could not get stable when the window size $w$ is even set to be 19, while both

EDP and ELP for SIMECK could. By confining the active bits to the $w$ least significant bits window, the static window strategy constructs a differential transition or linear correlation matrix that contains *all* trails only with active bits in the fixed window for each round. In other words, the static window strategy discards all trails with active bits out of the $w$ least significant bits window. However, SIMON has a stronger diffusion property due to the larger rotation constant. It means that the static window strategy may lose some good trails even in the first few rounds for SIMON. Therefore, choosing a flexible window for each round can avoid losing too many trails and lead to better distinguishers. Then, how to choose these windows for SIMON is an interesting problem. In this paper, we focus on this problem and propose the dynamic window strategy. Our contributions are listed as follows.

**Table 1**: Summary of distinguishers on SIMON.

| Block | Type | R | EDP/ELP | $\Delta_{in}/\Gamma^{\dagger}_{in}$ | $\Delta_{out}/\Gamma^{\dagger}_{out}$ | Ref. |
|---|---|---|---|---|---|---|
| 48 | Diff. | 17 | -46.32 | $\{7\},\{1,5,9\}$ | $\{1,5,9\},\{7\}$ | [5][1] |
| | | 17 | -46.38 | $\{0\},\{2,18,22\}$ | $\{2,18,22\},\{0\}$ | [7] |
| | | **17** | **-45.49** | $\emptyset,\{0\}$ | $\{0\},\emptyset$ | **Sect. 4** |
| | Linear | 16 | -44.92 | $\{1,5,21\},\{23\}$ | $\{1,5\},\{23\}$ | [11] |
| | | **17** | **-45.19** | $\{23\},\emptyset$ | $\emptyset,\{23\}$ | **Sect. 4** |
| 64 | Diff. | 22 | -61.32 | $\{6,10\},\{7,11,12\}$ | $\{6,10\},\{8\}$ | [5] |
| | | 23 | -61.93 | $\{0\},\{2,30\}$ | $\{2,6,30\},\{4\}$ | [7] |
| | | **23** | **-61.50** | $\{0,4\},\{6\}$ | $\{6\},\{0,4\}$ | **Sect. 4** |
| | Linear | 21 | -62.53 | $\{20,24\},\{22\}$ | $\{22\},\{20,24\}$ | [12] |
| | | **23** | **-60.24** | $\{25\},\{27,31\}$ | $\{27,31\},\{25\}$ | **Sect. 4** |
| 96 | Diff. | 30 | -92.2 | $\{20\},\{6,14,18,22\}$ | $\{8,16\},\{6,10,14\}$ | [2] |
| | | **33** | **-94.10** | $\emptyset,\{0\}$ | $\{0\},\emptyset$ | **Sect. 4** |
| | Linear | 33 | -92.60 | $\{47\},\emptyset$ | $\emptyset,\{47\}$ | [9] |
| | | **33** | **-91.74** | $\{47\},\emptyset$ | $\emptyset,\{47\}$ | **Sect. 4** |
| | | **34** | **-93.74** | $\{47\},\emptyset$ | $\{47\},\{45\}$ | **Sect. 4** |
| 128 | Diff. | 41 | -123.74 | $\{12\},\{6,10,14\}$ | $\{6,10,14\},\{12\}$ | [7] |
| | | **41** | **-112.99** | $(\{6\},\{0,4,8\})$ | $(\{0,4,8\},\{6\})$ | **Sect. 4** |
| | | **41** | **-122.98** | $(\emptyset,\{0\})$ | $(\{0,6\},\emptyset)$ | **Sect. 4** |
| | Linear | 41 | -123.07 | $\{63\},\emptyset$ | $\emptyset,\{63\}$ | [9] |
| | | **43** | **-124.59** | $\{63,59\},\{61\}$ | $\emptyset,\{63,57\}$ | **Sect. 4** |

† The input and output differences (resp. masks) are denoted by $\{a_0,a_1,...,a_i\},\{b_0,b_1,...,b_j\}$ with $a_i$ being the active position in the left branch while $b_j$ denotes the active bits in the right branch. Note that $\emptyset$ means there is no active bit in this branch.

*Dynamic Window (DW) with Minimal Loss.* In Sect. 3, we show how to dynamically choose the window for each round based on two strategies: the MLW (Minimal Loss Window) strategy and the LWIM (Link Window in the Middle) strategy. The aim of MLW is to lose as few trails as possible, especially when the number of possibly activated bits is no bigger than the window size $w$. If the window size is run out, we will start to exclude some possibly active bits out of the window and restrict them as zero

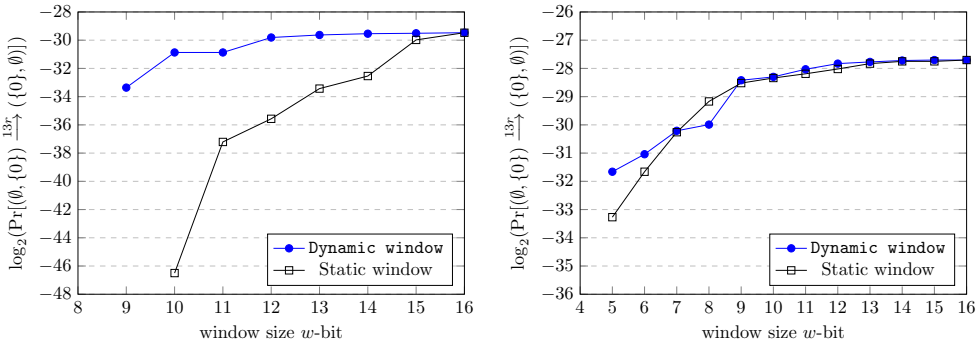


**Fig. 1**: Probability of a 13-round differential $(\emptyset, \{0\}) \rightarrow (\{0\}, \emptyset)$ evaluated under different windows for `SIMON32` (left) and `SIMECK32` (right).

differences/masks. To maintain a lower loss of trails, those excluded bits are chosen using a reasonable probability test. With `MLW`, we can determine the windows in each round according to the propagation property from input difference/mask.

To properly track the diffusion of differential for a longer number of rounds, we employ a meet in the middle like strategy, that is `LWIM`. For an $r(= 2i+1)$-round cipher, windows in the first $i$ rounds can be obtained with `MLW` using the input difference/-mask. For the last $i$ rounds, their windows are deduced by the output difference/mask from the backward direction using `MLW`. Then, we heuristically link the windows in the middle round deduced from two directions to lose fewer trails. Our experiments on `SIMON32` (Fig. 1) provide a stark illustration: a compact 9-bit `DW` achieves a tighter probability bound than a much larger 13-bit `SW`. This demonstrates that `DW` has superior efficiency and faster convergence over the `SW` approach. To better compare `DW` with `SW`, we also apply these two methods to `SIMON`-like ciphers with different rotation constants. Consequently, we observe that `DW` can not only approach the lower bound of EDP or ELP faster than `SW` but is also suitable for variants with some special rotation constants $(c > a, b)$, where `SW` cannot deal with.

*Improved Differential and Linear Hull.* Our goal is to find the differential or linear hull suitable for key recovery attack. Thus, the lower hamming weight for the input and output differences/masks and the higher EDP/ELP for the distinguisher are our objects. Using `DW`, we obtain a 17-round distinguisher for `SIMON48` with a single active bit in the input and output for the first time, which can be used to mount more rounds of key-recovery attack. Moreover, the previous longest distinguisher for `SIMON64` has 23 rounds and we improved the Hamming weight and probability for it. In addition, our identified 33-round distinguishers for `SIMON96` have higher EDP/ELP compared to those from the static window strategy, which enables us to extend to 34-round distinguisher at the first time. Finally, we get the improved Hamming weight and the EDP for the previous best differential distinguisher for `SIMON128`. We compare our new distinguishers with the best previous ones in Table 1. Detailed results are illustrated in Sect. 4.

**Table 2**: Summary of attacks against SIMON.

| Variant | Round | Attacked | Type | Data | Time | Succ. rate | Ref. |
|---|---|---|---|---|---|---|---|
| 48/72 | 36 | 23 | Diff. | $2^{47}$ | $2^{63.25}$ | 0.48 | [13] |
| | | **25** | | $\mathbf{2^{47}}$ | $\mathbf{2^{67.89}}$ | **0.42** | **Sect. 5** |
| | | 24 | Linear | $2^{47.92}$ | $2^{69.92}$ | 0.91 | [4] |
| | | **26** | | $\mathbf{2^{47}}$ | $\mathbf{2^{65.06}}$ | **0.66** | **Sect. 6** |
| 48/96 | 36 | 24 | Diff. | $2^{48}$ | $2^{78.99}$ | 0.48 | [14] |
| | | **26** | | $\mathbf{2^{47}}$ | $\mathbf{2^{92.51}}$ | **0.42** | **Sect. 5** |
| | | 25 | Linear | $2^{47.92}$ | $2^{91.92}$ | 0.91 | [4] |
| | | **27** | | $\mathbf{2^{47}}$ | $\mathbf{2^{89.05}}$ | **0.66** | **Sect. 6** |
| 64/96 | 42 | 29 | Diff. | $2^{63}$ | $2^{86.94}$ | 0.48 | [14] |
| | | **31** | | $\mathbf{2^{63}}$ | $\mathbf{2^{91.43}}$ | **0.41** | **Sect. 5** |
| | | 30 | Linear | $2^{63.52}$ | $2^{89.53}$ | 0.48 | [4] |
| | | **32** | | $\mathbf{2^{63}}$ | $\mathbf{2^{88.10}}$ | **0.68** | **Sect. 6** |
| 64/128 | 44 | 30 | Diff. | $2^{63}$ | $2^{110.99}$ | 0.48 | [14] |
| | | **32** | | $\mathbf{2^{63}}$ | $\mathbf{2^{123.89}}$ | **0.41** | **Sect. 5** |
| | | 31 | Linear | $2^{63.53}$ | $2^{120}$ | 0.48 | [4] |
| | | **33** | | $\mathbf{2^{63}}$ | $\mathbf{2^{120.09}}$ | **0.68** | **Sect. 6** |
| 96/96 | 52 | 43 | Linear | $2^{94}$ | $2^{89.6}$ | 0.63 | [9] |
| | | **44** | | $\mathbf{2^{95}}$ | $\mathbf{2^{90.09}}$ | **0.65** | **Sect. 6** |
| 96/144 | 54 | 37 | Diff. | $2^{95}$ | $2^{132.25}$ | 0.46 | [13] |
| | | **42** | | $\mathbf{2^{96}}$ | $\mathbf{2^{140.53}}$ | **0.56** | **Sect. 5** |
| | | 45 | Linear | $2^{95}$ | $2^{136.67}$ | 0.63 | [9] |
| | | **45** | | $\mathbf{2^{95}}$ | $\mathbf{2^{135.77}}$ | **0.70** | **Sect. 6** |
| 128/128 | 68 | 49 | Diff. | $2^{127}$ | $2^{127}$ | 0.73 | [13] |
| | | **50** | | $\mathbf{2^{127}}$ | $\mathbf{2^{127.07}}$ | **0.60** | **Sect. 5** |
| | | 53 | Linear | $2^{127}$ | $2^{121}$ | 0.67 | [9]† |
| | | **53** | | $\mathbf{2^{127}}$ | $\mathbf{2^{120.72}}$ | **0.67** | **Sect. 6** |
| 128/192 | 69 | 49 | Diff. | $2^{127}$ | $2^{183.25}$ | 0.48 | [13] |
| | | **51** | | $\mathbf{2^{127}}$ | $\mathbf{2^{186.76}}$ | **0.60** | **Sect. 5** |
| 128/256 | 72 | 50 | Diff. | $2^{127}$ | $2^{247.25}$ | 0.48 | [13] |
| | | **52** | | $\mathbf{2^{127}}$ | $\mathbf{2^{250.76}}$ | **0.60** | **Sect. 5** |

† We recomputed the attack parameters using the distinguishers in [9] for better comparison.

*Improved Key Recovery Attacks on Round-Reduced SIMON.* Based on our identified distinguishers, we perform differential key recovery attacks in Sect. 5 and linear key recovery attacks in Sect. 6 for SIMON, respectively. As exploited by Leurent *et al.* in [9],

we adopt the dynamic-key guessing technique [13, 14] for differential cryptanalysis, and Fast Walsh Transform approach [15] for linear cryptanalysis. As a result, we provided the best differential/linear attacks on `SIMON48`, `SIMON64`, and `SIMON96` in terms of the number of rounds, complexity, or success rate. We compare our key recovery attacks with the best previous ones in Table 2.

To allow our results to be reproduced, all source code utilized in this paper is available in an anonymous repository at https://github.com/Improved-Simon.

## 2 Some Known Properties and Analyses of `SIMON`

`SIMON`$n/k$ [1] are Feistel ciphers with block size $n \in \{32, 48, 64, 96, 128\}$ and key size $k$. For each variant, their key size $k$ and the number of rounds $r$ are listed in Table 3.

**Table 3**: All variants of `SIMON`

| $n$ | 32 | 48 | | 64 | | 96 | | 128 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | 64 | 72 | 96 | 96 | 128 | 96 | 144 | 128 | 192 | 256 |
| $r$ | 32 | 36 | 36 | 42 | 44 | 52 | 54 | 68 | 69 | 72 |

For the $i$-th round ($0 \le i \le r-1$), we respectively denote $x^{(i+1)}$ and $x^{(i)}$ as the left $n/2$-bit and right $n/2$-bit of the $n$-bit input state. Due to its Feistel structure, the right part of its output state is also $x^{(i+1)}$. The left output part is then denoted as $x^{(i+2)}$ and obtained with $x^{(i+2)} = x^{(i)} \oplus f\left(x^{(i+1)}\right) \oplus k^{(i)}$. The function $f$ is a quadratic function designed to be $f(x) = (\mathrm{S}^8(x) \wedge \mathrm{S}^1(x)) \oplus \mathrm{S}^2(x)$, where $\mathrm{S}^a(x)$ denotes the $a$-bit left cyclic rotation of $x$, $\wedge$ as the bitwise AND, and $\oplus$ as the bit-wise exclusive or (XOR), respectively.

The key schedule of `SIMON` is linear. Let $c = 2^{n/2} - 4 =$ `0xff`$\cdots$`fc` and $z_j$ the constant sequence[1], the subkey of `SIMON`, with $m$ key words $k^{(i+m)}$ are generated by:

$$\begin{cases} c \oplus (z_j)_i \oplus k^{(i)} \oplus \left(I \oplus S^{-1}\right) S^{-3} k^{(i+1)}, & \text{if } m = 2 \\ c \oplus (z_j)_i \oplus k^{(i)} \oplus \left(I \oplus S^{-1}\right) S^{-3} k^{(i+2)}, & \text{if } m = 3 \\ c \oplus (z_j)_i \oplus k^{(i)} \oplus \left(I \oplus S^{-1}\right) \left(S^{-3} k^{(i+3)} \oplus k^{(i+1)}\right), & \text{if } m = 4 \end{cases}$$

for $0 \le i < r - m$, where $\mathrm{S}^{-a}(x)$ represents the $a$-bit right cyclic rotation of $x$. For more details of the constant sequence $z_j$, one can refer to [16].

Although `SIMON` uses a very simple quadratic round function, its differential and linear properties are hard to investigate due to the dependency between multiple active AND gates.

[1] The constant sequence $z_j$ uses as the round constants.

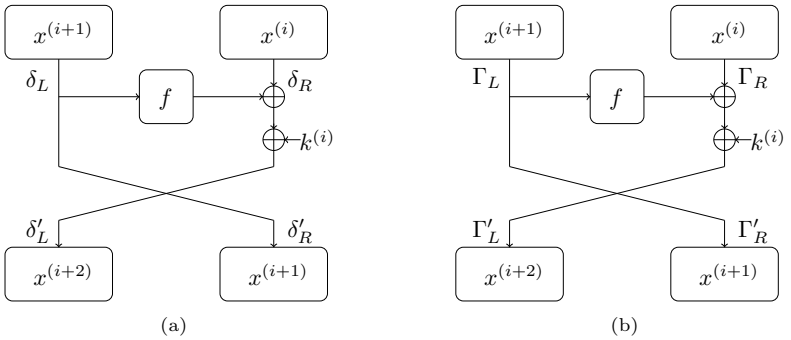


**Fig. 2**: Difference (left) and mask (right) propagation of one-round SIMON-like cipher.

***Analysis of SIMON-like Round Functions***

A foundational method for analyzing SIMON-like ciphers was introduced by Kölbl et al. at CRYPTO 2015 [5]. Their approach systematically studies the differential and linear properties of functions of the form $f(x) = (\mathrm{S}^a(x) \wedge \mathrm{S}^b(x)) \oplus \mathrm{S}^c(x)$, where $a$, $b$, and $c$ are integers. The key results of their work are briefly recalled below.

*Differential Properties.* As illustrated in Figure 2a, let $(\delta_L, \delta_R)$ and $(\delta'_L, \delta'_R)$ denote the input and output differences for the left and right halves of the round function, respectively. The probability of a single-round transition from $(\delta_L, \delta_R)$ to $(\delta'_L, \delta'_R)$ is given by:

$$\Pr\left[(\delta_L, \delta_R) \to (\delta'_L, \delta'_R)\right] = \begin{cases} 2^{-\dim(U_{\delta_L})}, & \text{if } \delta_L = \delta'_R \text{ and } \delta_R \oplus \delta'_L \in U_{\delta_L} \\ 0, & \text{otherwise} \end{cases}$$

Here, $U_{\delta_L}$ is a coset of the image of the linear function $g(x) = f(x) \oplus f(x \oplus \delta_L) \oplus f(\delta_L) \oplus f(0)$, and $\dim(U_{\delta_L})$ denotes its dimension. The probability of a complete $r$-round differential trail is the product of these single-round probabilities. Consequently, the total probability of a differential is the sum of the probabilities of all its constituent trails. However, since a differential can encompass a vast number of trails, enumerating and summing all of them is typically computationally infeasible.

*Linear Properties.* As illustrated in Figure 2b, let $(\Gamma_L, \Gamma_R)$ and $(\Gamma'_L, \Gamma'_R)$ represent the input and output masks for the left and right halves of the round function, respectively. The squared correlation of a single-round transition from $(\Gamma_L, \Gamma_R)$ to $(\Gamma'_L, \Gamma'_R)$ is given by:

$$\mathrm{Cor}^2\left[(\Gamma_L, \Gamma_R) \to (\Gamma'_L, \Gamma'_R)\right] = \begin{cases} 2^{-\dim(V_{\Gamma_R})}, & \text{if } \Gamma_R = \Gamma'_L \text{ and } \Gamma_L \oplus \Gamma'_R \in V_{\Gamma_R} \\ 0, & \text{otherwise} \end{cases}$$

Here, $V_{\Gamma_R}$ is the image of the linear map defined by the expression $((\Gamma_R \wedge \mathrm{S}^{b-a}(x)) \oplus (\mathrm{S}^{-(b-a)}(\Gamma_R \wedge x))) \ggg a$, where $x \ggg a$ denotes an $a$-bit right cyclic rotation of $x$. The squared correlation of an $r$-round linear trail is the product of the single-round squared

correlations. The Expected Linear Potential (ELP) of a linear hull is then calculated by summing the squared correlations of all its constituent trails [17]. However, this summation is often computationally infeasible due to the potentially large number of trails within the hull.

For a detailed account of the formal proofs and derivations, see [5].

***Equivalence between Differential and Linear Trail.***

This is another important property of SIMON-like ciphers which has been observed by several works [18–20]. Specifically, given a differential trail with probability $p$, which is

$$(\delta_L^0, \delta_R^0) \to (\delta_L^1, \delta_R^1) \to \cdots \to (\delta_L^r, \delta_R^r),$$

a linear trail covering the same number of rounds written as

$$(\overleftarrow{\delta_R^0}, \overleftarrow{\delta_L^0}) \to (\overleftarrow{\delta_R^1}, \overleftarrow{\delta_L^1}) \to \cdots \to (\overleftarrow{\delta_R^r}, \overleftarrow{\delta_L^r})$$

with $\overleftarrow{x}$ being the bit-reversed $x$. However, the squared correlation of this linear trail may not equal to $p$ [9] and shall be re-evaluated using the aforementioned method [5].

# 3 Dynamic Window for Transition Matrix

Given a differential or linear hull, it's hard to efficiently approximate its probability or ELP due to the massive number of trails contained. To tackle this, Leurent *et al.* [9] constructed an efficient framework for SIMON-like ciphers based on the difference transition matrix or linear correlation matrix $M$, which contains transition probability.

Recall the definition of differential transition matrix. Let $\Delta_A = \{\delta_1, \delta_2, \ldots, \delta_N\}$ be a pre-selected ordered set of $N$ differential values. For a given round $r$ of the cipher, the differential transition matrix $M^{(r)}$ is an $N \times N$ square matrix that quantifies the propagation probabilities between these differences. Each element $M_{ij}^{(r)}$, corresponding to the $i$-th row and $j$-th column, is defined as the conditional probability that an input difference $\delta_i \in \Delta_A$ at the beginning of round $r$ results in an output difference $\delta_j \in \Delta_A$ at the end of that same round. Formally, the element $M_{ij}^{(r)}$ is given by:

$$M_{ij}^{(r)} := \Pr[\Delta X^{(r)} = \delta_j \mid \Delta X^{(r-1)} = \delta_i]$$

where: $i, j \in \{1, 2, \ldots, N\}$. $\Delta X^{(r-1)}$ denotes the input difference to round $r$. $\Delta X^{(r)}$ denotes the output difference from round $r$. $\Pr[\cdot \mid \cdot]$ denotes the conditional probability.

A key aspect of this approach is that the matrices only consider differential trails where all active bits are confined to a fixed $w$-bit window, specifically the $w$ least significant bits (LSBs) of each round. This technique yielded longer differential characteristics and linear hulls than previously reported for both SIMON and SIMECK. For SIMON in particular, the authors noted the potential for further improvement and encouraged future research on the cipher.

In this section, we follow the similar framework as Leurent *et al.* [9], however, a dynamic way to choose the $w$-bit window for each round is used. Firstly, we propose

the MLW (short for minimal loss window) strategy and LWIM (short for link window in the middle) strategy to help us determine which bits shall be included in the window for each round in Sect. 3.1. Based on these dynamic windows, a modified algorithm for computing EDP or ELP is given at Sect. 3.2. We applied this modified framework on SIMON and SIMECK. Compared with the static windows adopted in [9], we can achieve better differentials and linear hulls for SIMON. Detailed application results are shown in Sect. 4.

## 3.1 New Strategies for Window Choosing

To make it clear, we first introduce our strategies for choosing the window from the perspective of constructing the differential transition matrix for a differential.

For the linear hull, we determine the window for each round by leveraging the well-known **structural duality** between differential and linear trails in SIMON-like ciphers. Specifically, for any given linear trail with specific active masks in each round, there exists a *unique* corresponding differential trail that has the same active bit positions (after bit-reversal and reversing the round order). However, as correctly noted in [9] and our paper, the probability of the differential trail is not necessarily equal to the squared correlation of the linear trail. Our framework respects this distinction by re-evaluating the probabilities/correlations independently. We use the structural duality *only* to determine the window for the linear hull.

The process is as follows: to find the windows for a linear hull with input mask $(\Gamma_L^0, \Gamma_R^0)$ and output mask $(\Gamma_L^r, \Gamma_R^r)$, we consider its dual differential, which has an input difference of $(\overleftarrow{\Gamma_R^r}, \overleftarrow{\Gamma_L^r})$ and an output difference of $(\overleftarrow{\Gamma_R^0}, \overleftarrow{\Gamma_L^0})$, where $\overleftarrow{x}$ denotes the bit-reversed $x$. We then apply our strategies to this dual differential to obtain its dynamic windows. If the window for the $j$-th round of the dual differential comprises the bit positions $\{t_0^j, t_1^j, \ldots, t_{w-1}^j\}$, then the corresponding window for the $j$-th round of the original linear hull is composed of the bit-reversed positions $\{(\frac{n}{2}-1)-t_0^j, \ldots, (\frac{n}{2}-1)-t_{w-1}^j\}$.

### 3.1.1 MLW: A Window with Minimal Loss.

By restricting the difference to the $w$-bit window, Leurent *et al.* construct a differential transition matrix that contains *all* trails only active at this window. In other words, they try to control the diffusion of the input difference by discarding all trails that diffuse out of the window. Windows in every involved rounds are chosen to be the $w$ least significant bits. Due to the strong diffusion of non-linear part $w+5$ (for SIMECK) or $w+8$ (for SIMON) cyclic rotation, this strategy may lose many valuable trails even before the number of diffused bits exceed the window size, especially for SIMON. We show in Table 4 as an illustration where the input difference for SIMON48 is set to be $(\emptyset, \{0\})$ and the window size is 17. Note that the static window chosen by Leurent *et al.* is in the range between 0 and 16. Hence, the 17-th and 18-th bits in the fourth round are discarded. However, we run out of the window only after the fifth round. Thus, it will be better if we only discard trails after running out of the window size.

In our MLW strategy, we keep all trails before the window size is running out. When the number of diffused bits exceeds the window size, we discard some bits with the aim of losing less trails according to the Minimal Active Probability Test.

**Table 4**: Comparison of the bit loss on SIMON48 with different 17-bit window, where bits colored are discarded by our dynamic window. Note that the static window is composed of bits 0, 1, $\cdots$, 16.

| Round | Diffusion bit position with input difference $(\emptyset, \{0\})$ | Ours | [9] |
|---|---|---|---|
| 0 | [] | 0 | 0 |
| 1 | [0] | 0 | 0 |
| 2 | [1,2,8] | 0 | 0 |
| 3 | [0,2,3,4,9,10,16] | 0 | 0 |
| 4 | [0,1,3,4,5,6,8,10,11,12,17,18] | 0 | 2 |
| 5 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,16,18,19,20] | 2 | 3 |
| 6 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22] | 6 | 6 |
| 7(full) | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23] | 7 | 7 |
| 8 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23] | 7 | 7 |

*Minimal Active Probability Test (Considering Dependency).* Assume that the input difference of the differential is $\triangle_{in}$. We respectively denote $\mathcal{B}^{(j)}$ and $\mathcal{D}^{(j)}$ as the number of diffused bits and discarded bits in the $j$-th round. Therefore, we have $\binom{\mathcal{B}^{(j)}}{\mathcal{D}^{(j)}}$ cases of the possible discarded bits. In the following, we focus on how to determine the window in the $j$-th round. Whatsmore, we only need to determine the window of the left part of the $j$-round output state because the right window is the same as the window of the left part of the input state.

To get a window with minimal loss of trails, we consider the probability $\Pr_c$ of the number of kept trails among all possible trails in the differential covering the previous $j$ rounds. Note that $\Pr_c$ is equal to the probability of a truncated differential with a fixed input difference $\triangle_{in}$, while the output difference is a truncated difference $\triangle^{\star}_{out}$. This truncated difference $\triangle^{\star}_{out}$ has zero differences on these $\mathcal{D}^{(i)}$ bits while other bits can be active or not. Note that $\triangle^{\star}_{out}$ takes all zero differences on bits that will never be diffused. The probability $\Pr_c$ can be reasonably estimated as follows. We randomly construct $N$ plaintext pairs satisfying the input difference $\triangle_{in}$, and then calculate how many pairs fulfilling the truncated difference $\triangle^{\star}_{out}$. Then, we formally define the estimate of $\Pr_c$ as

$$\Pr_c \approx \frac{1}{N}\#\{x \in \{0,1\}^n : \mathrm{E}(x) \oplus \mathrm{E}(x \oplus \triangle_{in}) = \triangle^{\star}_{out}\}.$$

For each case of those $\mathcal{D}^{(i)}$ possible discarded bits, we can obtain its corresponding $\Pr_c$, and then choose the case that with the maximal $\Pr_c$ to discard these $\mathcal{D}^{(i)}$ bits.

Take SIMON48 for an example. When the input difference is $(\emptyset, \{0\})$, the diffused bits in each round is shown in Table 4. As one can see, we run out the window size 17 at the fifth round. Hence, we need to determine which bits to be discarded from the

19 bits, which leads to $\binom{19}{2} = 171$ cases. Similarly, there are $\binom{23}{6} \approx 2^{16.62}$, $\binom{24}{7} \approx 2^{18.4}$ and $\binom{24}{7} \approx 2^{18.4}$ cases for the 6-th, 7-th and 8-th round, respectively. For each case, we obtain its $\Pr_c$ utilizing $N = 2^{24}$ plaintext pairs. To gain an accurate estimation, $N$ is chosen to be $2^{24}$ here since it can ensure that the estimated value of $\Pr_c$ is stable. As a result, those bits bolded in Table 4 are discarded. Compared with the static window, our dynamic window have the less chance of losing trails.

However, for variants with a larger state, the dependency-aware test described above cannot be executed, as the number of cases for discarded bits becomes computationally impractical. More precisely, for `SIMON64`, the number of combinations of bits to discard reaches $\binom{20}{3} \approx 2^{10.15}$, $\binom{26}{9} \approx 2^{21.58}$, $\binom{30}{13} \approx 2^{26.84}$, and $\binom{32}{15} \approx 2^{29.08}$ at the 5-th, 6-th, 7-th, and 8-th rounds, respectively. Even with a sample size of $N = 2^{24}$ plaintexts, the total complexity to test all combinations would be infeasibly large (e.g., $2^{24} \times \binom{26}{9} \approx 2^{45.58}$ operations for the 6-th round).

*Minimal Active Probability Test (Independent).* To overcome this computational barrier, we introduce a heuristic approach for larger variants: we assume that each bit in the same round activates independently of the others. Under this assumption, we can simply test the activation probability of each individual bit using a large number of plaintext pairs satisfying the fixed input difference $\triangle_{in}$. We then discard the $\mathcal{D}^{(i)}$ bits that exhibit the lowest activation probabilities. This independence assumption is a necessary simplification, but its effectiveness can be validated. For `SIMON48`, where the full dependency-aware test is still feasible, we observed that the 17-bit window deduced from this independence-based test is **identical** to the one derived from the full test. This result strongly suggests that the independence assumption is a reasonable and effective approximation, at least in the early rounds where the test is applied.

The choice of the sample size, $N = 2^{24}$ plaintext pairs, was determined empirically to ensure a **stable estimation**. We define a stable estimation as the point at which further increasing the sample size $N$ does not change the outcome of the test—specifically, it does not alter the *ranking* of which bits are to be discarded. We observed that for values of $N \geq 2^{24}$, the set of bits identified for exclusion remained consistent. This gives us confidence that our choice of which bits to discard is robust and not an artifact of insufficient sampling. However, we also note that as the number of rounds grows, the activation probability of each bit approaches 1/2, limiting the effectiveness of this test to the rounds before full diffusion is reached.

### 3.1.2 `LWIM`: A Combined Window Location Strategy

As noted previously, the `MLW` strategy is most effective in the initial rounds of the cipher, before full diffusion is reached and activation probabilities converge towards 1/2. To construct effective dynamic windows for the entire distinguisher, we introduce the `LWIM` **(Link Window in the Middle)** strategy. This technique avoids applying `MLW` in later rounds where it is ineffective, and instead relies on constraints imposed by both the input and output of the distinguisher.

The `LWIM` strategy operates as follows. For a target $r$-round distinguisher, where $r$ is typically odd (e.g., $r = 2i+1$), we divide the task into three parts as shown in Fig. 3:

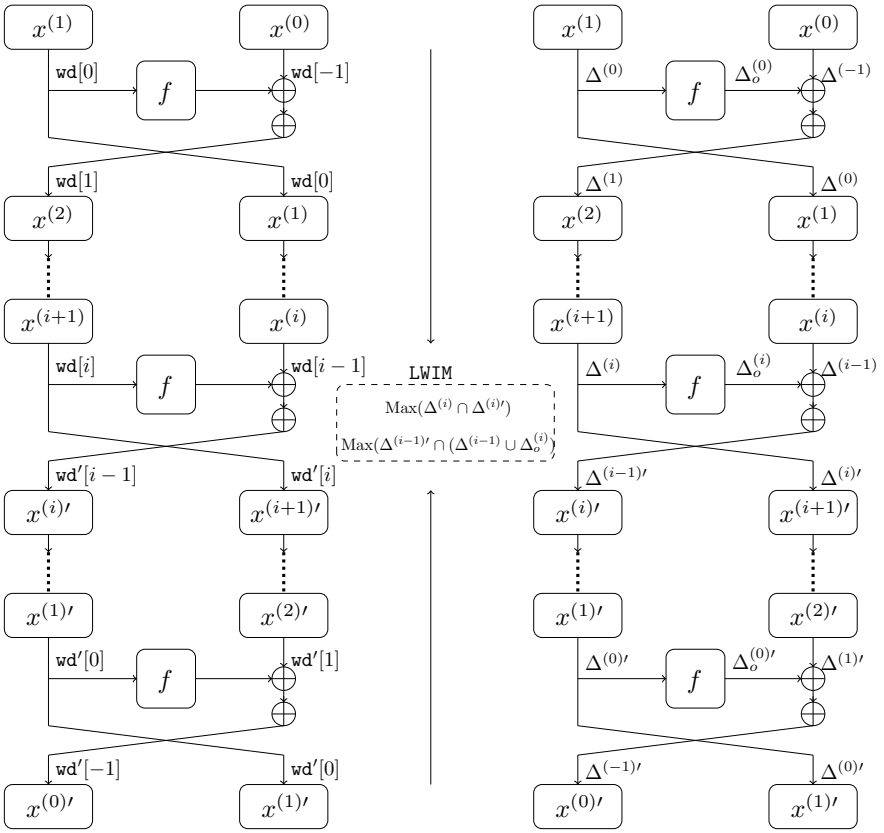


**Fig. 3**: `LWIM` strategy. $\Delta^{(i)}$ (resp. $\Delta^{(i)'}$) is the set containing all differences where only bits in the window `wd[`$i$`]` (resp. `wd'[`$i$`]`) can take non-zero differences.

1. **Forward Propagation:** We apply the `MLW` strategy to the input difference $(\delta_L^{(0)}, \delta_R^{(0)})$ to determine the dynamic windows `wd[`$i$`]` for the first $i$ rounds. This captures the forward diffusion from the input.
2. **Backward Propagation:** Crucially, we then apply the `MLW` strategy in the reverse direction, starting from the output difference $(\delta_L^{(r)}, \delta_R^{(r)})$. This allows us to determine the dynamic windows `wd'[`$i$`]` for the last $i$ rounds by analyzing the backward diffusion.
3. **Linking in the Middle:** The final step is to heuristically link these two sets of windows in the middle round $(i+1)$.

For the middle round, we must ensure that the windows chosen for the states $x^{(i+1)}$, $x^{(i)}$, $x^{(i)'}$, and $x^{(i+1)'}$ are compatible to minimize trail loss. We denote $\Delta^{(i)}$ (resp. $\Delta^{(i)'}$) as the set of all differences constrained to the window `wd[`$i$`]` (resp. `wd'[`$i$`]`). To preserve as many trails as possible, we aim to maximize the size of the intersections $\Delta^{(i)} \cap \Delta^{(i)'}$ and $\Delta^{(i-1)'} \cap (\Delta^{(i-1)} \cup \Delta_0^{(i)})$, where $\Delta_0^{(i)}$ is the set of output differences from the round function $f$ in the middle round. The sizes of these intersections are determined by

the input and output differences of the target differential. A straightforward way to maximize them is to focus on symmetric distinguishers where $\delta_L^{(0)} = \delta_R^{(r)}$ and $\delta_R^{(0)} = \delta_L^{(r)}$. In this case, the forward-derived windows $\texttt{wd}[i]$ will be identical to the backward-derived windows $\texttt{wd}'[i]$, which simplifies the intersections to $\Delta^{(i)} \cap \Delta^{(i)'} = \Delta^{(i)}$ and $\Delta^{(i-1)'} \cap (\Delta^{(i-1)} \cup \Delta_0^{(i)}) = \Delta^{(i-1)}$. This symmetric construction is a powerful heuristic for finding high-probability distinguishers, though the `LWIM` strategy is also applicable to asymmetric cases.

### 3.2 Modified Algorithm for Computing Transition Probability

Recall that when using the static windows, the difference sets $\Delta^{(j)}$ and $\Delta^{(j)'}$ are equal for all rounds $j$. In other words, with the notation used in Fig. 3,

$$\Delta^{(-1)} = \Delta^{(0)} = \cdots = \Delta^{(i)} = \cdots = \Delta^{(0)'} = \Delta^{(-1)'}.$$

Therefore, the differential propagation matrix (resp. linear transition matrix) in each round are indexed with the same difference (resp. mask) values. However, when dynamically choosing the window for each round, the matrices will have different indexes since the difference sets in each round are different. Note that in our dynamic window strategy, we only restrict that $\Delta^{(j)} = \Delta^{(j)'}$ while no restrictions on these $\Delta^{(j)}$ or $\Delta^{(j)'}$ in each round are applied. To deal with these different indexes, we established a hash table to store the index of each matrix.

Next, we will detail our modified algorithm that evaluates EDP for a differential as follows and focus on the one-round propagation. We denote $\delta_L^{(i)}$ ($\delta_R^{(i)}$) and $\Delta_L^{(i)}$ ($\Delta_R^{(i)}$) as the input differences and its set of the left (right) branch of the $i$-th round, respectively. At first, we generate all possible $\delta_L^{(i)}$ and $\delta_R^{(i)}$ according to the corresponding $w$-bit dynamic windows. Specifically, they can only have non-zero differences on bits contained in the window. Secondly, we compute the output difference $\gamma$ of the $f$ function and check whether $\gamma \in U_{\delta_L^{(i)}}$. If so, we can obtain $\delta_L^{(i+1)}$ and then check whether keep it according to its window. Similar as [9, Algorithm 1], the complexity of the algorithm is bounded by $r \times 2^{2w} \times \max_{\delta_L^{(i)} \in \Delta_L^{(i)}} |U_{\delta_L^{(i)}}|$ elementary operations. We assume that the time complexity of the hash lookup table is negligible compared to floating-point arithmetic. With the help of hash tables, we can efficiently update the differential propagation matrix indexed by $(\delta_R^{(i)} \oplus \gamma, \delta_L^{(i)})$ and finally obtain EDP of the differential. Details of the above procedure have been described in Algorithm 1.

## 4 Application of the Dynamic Window Strategy

We applied the dynamic window strategy on evaluating EDP and ELP for almost all variants of `SIMON` and obtained better distinguishers. The largest window size we can choose is 17-bit, which consumes 256GB of memory size. All of our distinguishers were evaluated using a 17-bit dynamic window. We also note that all the tests in this section are implemented on a server with AMD EPYC 7302 16-Core Processor, and the RAM size is 256GB.

**Algorithm 1** Computation of Probability Using Transition Matrix

**Input:** An $r$-round differential $(\delta_L^{(0)}, \delta_R^{(0)}) \to (\delta_L^{(r)}, \delta_R^{(r)})$ with its dynamic windows
**Output:** EDP (Expected Differential Probability) of the differential

Initialize $X \leftarrow [0 \text{ for } i \in \Delta_L^{(0)},\ j \in \Delta_R^{(0)}]$ as the distribution matrix storing the transition probability
Set $X[\delta_L^{(0)}, \delta_R^{(0)}] \leftarrow 1$ to initialize the matrix with the input difference
**for** $i = 1$ to $r - 1$ **do**
    Set $\Delta_L^{(i)} \leftarrow \delta_L^{(i)}$ restricted by the window
    Set $\Delta_R^{(i)} \leftarrow \delta_R^{(i)}$ restricted by the window
    **for** each $\delta_L^{(i)} \in \Delta_L^{(i)}$ **do**
        **for** each $\delta_R^{(i)} \in \Delta_R^{(i)}$ **do**
            **for** each $\gamma \in U_{\delta_L^{(i)}}$ **do**
                **if** $(\delta_R^{(i)} \oplus \gamma)$ fulfills its window constraint **then**
                $$Y[\delta_R^{(i)} \oplus \gamma, \delta_L^{(i)}] \leftarrow Y[\delta_R^{(i)} \oplus \gamma, \delta_L^{(i)}] + 2^{-\dim(U_{\delta_L^{(i)}})} X[\delta_L^{(i)}, \delta_R^{(i)}]$$
                **end if**
            **end for**
        **end for**
    **end for**
    Update $X \leftarrow Y$ for the next iteration
**end for**
    **return** $X[\delta_L^{(r)}, \delta_R^{(r)}]$ as the computed EDP

## 4.1 A Heuristic Setting for DW to Reach a Local Optimum

As explained in Sect. 2, `DW` strategy is designed to minimize the loss of high-probability trails by tracking the diffusion of active bits. The core mechanism for this is the `MLW` (Minimal Loss Window), which is highly effective in the initial rounds of a cipher. However, a fundamental challenge arises after full diffusion is reached. In these later rounds, determining the optimal dynamic window that connects a given intermediate state to a desired final output difference is an extremely difficult problem, and can be shown to be computationally intractable (akin to an NP-hard problem). A purely one-directional, forward-propagating `DW` strategy would inevitably fail to efficiently find a path to a specific, low Hamming weight output difference.

Our solution is a meet-in-the-middle approach `LWIM`. Instead of attempting to bridge the entire round count in one direction, we proceed as follows:

1. Apply the `MLW` strategy to generate a `DW` for the first few rounds (e.g., 8 rounds for `SIMON`) in the forward direction from the input difference.
2. Apply `MLW` to generate a `DW` for the last few rounds in the backward direction from the desired output difference.
3. Use the `LWIM` strategy to link these two shorter, more manageable window heuristics in the middle.

This meet-in-the-middle methodology provides two significant advantages. First, it makes the problem of finding a path to a specific output difference computationally feasible. Second, by starting from both a desired input and a desired output (typically with low Hamming weight), it naturally biases the search towards finding distinguishers that are key-recovery friendly. For example, this process allows us to construct what we term a local optimal 17-round DW for a differential such as $(\emptyset, \{0\}) \rightarrow (\{0\}, \emptyset)$.

*Application on SIMON48.* With the MLW and LWIM strategies, we divided the cipher into two parts, and obtained the local optimal 17-bit DW for a 17-round differential $(\emptyset, \{0\}) \rightarrow (\{0\}, \emptyset)$. We show the DW in Appendix Table 12. The input and output differences are denoted by $(\{a_0, a_1, \cdots\}, \{b_0, b_1, \cdots\})$ with $a_i$ being the active position in the left branch while $b_j$ denotes the active bits in the right branch. Note that $\emptyset$ means there is no active bit in this branch. The 17-round differential has the probability of $2^{-45.49}$. Meanwhile, we obtained the ELP of the 17-round linear hull $(\{23\}, \emptyset) \rightarrow (\emptyset, \{23\})$ whose DW can be determined due to the duality between differential and linear trails, as explained in Sect. 3.1. The ELP of this linear hull is $2^{-45.19}$. We also compare DW with the SW in Fig. 4.

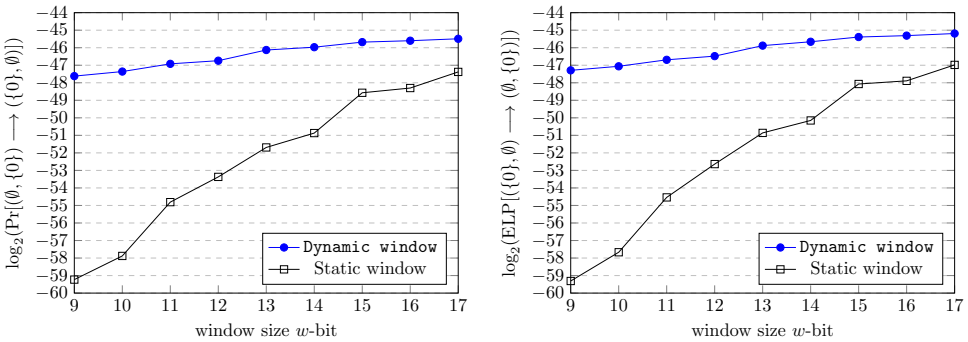


**Fig. 4**: EDP (left) and ELP (right) of the 17-round differential and linear hull for SIMON48.

## 4.2 Construct DW for Longer Differentials

As aforementioned, the 17-round DW for the differential $(\emptyset, \{0\}) \rightarrow (\{0\}, \emptyset)$ is a local optimal one with single active bit of input-output difference for SIMON. To obtain DW for differentials covering more rounds ($> 17$) and fully utilize this local optimal differential, we adopt two approaches. Firstly, one can append an optimal trail covering a short number of rounds before and after the 17-round one. In this case, a differential with multi-active bits of input-output difference can be obtained. Note that DW in these added top and bottom rounds can also be determined with MLW. Given this DW, EDP of this multi-active bits differential can be evaluated using Algorithm 1. Hence, DW discovers an effective multi-active bits differential covering more than 17 rounds. This is also one of the advantages of DW. The second approach utilizes the 17-round local

optimal differential $(\emptyset, \{0\}) \to (\{0\}, \emptyset)$, as well as the differential $(\emptyset, \{0\}) \to (\emptyset, \{0\})$ covering its top 16 rounds and the differential $(\{0\}, \emptyset) \to (\{0\}, \emptyset)$ covering its bottom 16 rounds. EDPs of these two 16-round differentials are the same as that of the 17-round one. Thus, one can construct a longer differential by cascading these two or three differentials. EDP of this longer differential can be computed with Algorithm 1, where its DW can be constructed with DW for these three differentials.

*Application on SIMON64.* Based on the 17-round differential, we follow the first approach where three rounds are respectively added before and after it to construct 23-round multi-active bits differential $(\{0, 4\}, \{6\}) \to (\{6\}, \{0, 4\})$ for SIMON64. The DW for this 23-round differential is shown in Appendix. With Algorithm 1, its EDP is $2^{-61.50}$. Similarly, using the DW for the differential and the duality of linear trails, we obtained the 23-round $(\{25\}, \{27, 31\}) \xrightarrow{23r} (\{27, 31\}, \{25\})$ linear hull with ELP $2^{-60.24}$. For comparison, we also use a 17-bit SW to re-evaluate the EDP and ELP of these newly discovered distinguishers. As a result, EDP of the 23-round differential is $2^{-62.88}$, while the ELP for the linear hull is $2^{-61.36}$. Both are worse than those evaluated under 17-bit DW.

*Application on SIMON96.* We constructed a 33-round differential $(\emptyset, \{0\}) \xrightarrow{33r} (\{0\}, \emptyset)$ following the second approach, which is the same differential proposed by [9]. However, with our Algorithm 1, this differential can have a higher EDP than theirs due to the local optimal DW. More precisely, its EDP is $2^{-94.10}$. The distinguisher is constructed as $(\emptyset, \{0\}) \xrightarrow{16r} (\emptyset, \{0\}) \xrightarrow{1r} (\{0\}, \emptyset) \xrightarrow{16r} (\{0\}, \emptyset)$. We show the DW in Appendix. Meanwhile, we can obtain a linear hull $(\{47\}, \emptyset) \xrightarrow{33r} (\emptyset, \{47\})$ with ELP $2^{-91.74}$. Similar to [9], we can append one round after the distinguisher with the loss of ELP $2^{-2}$. Hence, a 34-round linear hull $(\{47\}, \emptyset) \xrightarrow{34r} (\{47\}, \{45\})$ can be achieved with ELP $2^{-93.74}$.

*Application on SIMON128.* For SIMON128, we utilized a 33-round local optimal DW and appended eight rounds of SW after the 33-round DW to construct a 41-round one. Finally, we obtained a 41-round differential $(\emptyset, \{0\}) \xrightarrow{41r} (\{0, 6\}, \emptyset)$ with EDP of $2^{-122.98}$. Then, with the 41-round DW, we obtained a linear hull $(\{63\}, \emptyset) \xrightarrow{41r} (\emptyset, \{57, 63\})$ with ELP $2^{-120.59}$. Similarly, we append two rounds on the top of the linear hull with the loss of ELP $2^{-4}$ to get a 43-round one whose input mask is $(\{63, 59\}, \{61\})$.

***Correspondence Between Theoretical Estimates and Experimental Results.*** Experimentally verifying the distinguishers' exact probabilities is computationally infeasible for the full ciphers. However, prior work [9] has demonstrated through tests on small-state versions of SIMECK and SIMON that their distinguishers follow a Poisson distribution. Extending the framework of Leurent et al., our method constructs varied transition matrices from different window bits to improve the estimation of the clustering effect. This approach yields more accurate probability estimates for distinguishers while maintaining the same underlying theoretical distribution.

# 5 Key-Recovery Attacks Using Differential Cryptanalysis

In this section, we detail differential key-recovery attacks against SIMON using dynamic key-guessing technique [13, 14]. We denote the involved key as $k_p, k_f$ on the plaintext side, and $k_b, k_c$ on the ciphertext side (shown in Fig. 5), and the number of the key bits as $\kappa_p, \kappa_f$ ($\kappa_b, \kappa_c$). The total number of the involved key bits is denoted as $\kappa_g = \kappa_p + \kappa_f + \kappa_b + \kappa_c$. Then, we briefly recall this technique and introduce our attacks in Sect. 5.2.

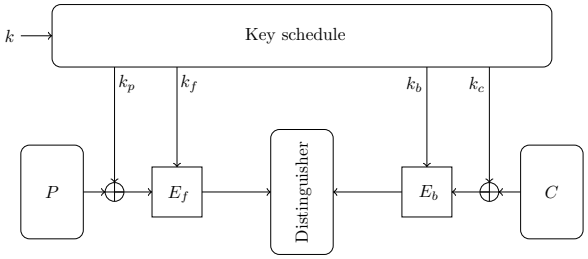


**Fig. 5**: General description of a cipher.

## 5.1 Dynamic Key-Guessing Technique [8, 14]

The main idea of dynamic key-guessing is to remove the redundancy of the guessed subkey according to the immediate values of a given differential path, and the process of the attack is as follows.

*Sufficient Conditions.* Based on the $R$-round distinguisher $\Delta_i \to \Delta_o$, we append $r_f$ rounds before and $r_b$ rounds after and build the extended differential for our key recovery attacks. Then we can identify the sufficient bit conditions as shown in Table 5.

*Data Collection.* First, we build structures of plaintexts where the bits with a fixed difference in round one are identical for all the plaintexts in $S_1$ and $S_2$, where $S_1$ and $S_2$ denote two structures with fixed differences. Each structure is composed of $2^{n-l_1}$ plaintexts, where $l_1$ denote the number of the fixed bits at round one. For each structure, we encrypt the plaintexts and obtain the corresponding ciphertexts, totally $D \times 2^{n-l_1}$ encryption, where $D$ denotes the data complexity. Then, we can pair these ciphertexts and filter them at the output according to the fixed difference in $C$, and $2^{2(n-l_1)-l_{r-1}}$ pairs remain in each pair of structure.

*Key Recovery.* Then, we need to associate partial key guesses to each of these pairs and make sure they can validate the difference of distinguisher. From the outer to the inner, for each sufficient bit condition, we associate to each pair the possible combinations of key bits that lead to the desired input-output difference of the distinguisher. For each pair, we increment the corresponding counters of the combination of key bits that lead to $\Delta_i$ and $\Delta_o$. In total, we have incremented $\lambda_W \times 2^{\kappa_g}$ counters on average, with $\lambda_W$ the average value of a counter for a wrong key guess. After processing

all pairs, we set a threshold $s$, and for each counter greater than $s$, we store them as the key candidate. Finally, we exhaustively search the remaining key bits not guessed under these key candidates and construct the master key.

*Complexity and Success Probability.* We denote the average value of counters for the right key $\lambda_R$ for the wrong key $\lambda_W$, and then we estimate them for computing the complexity and the success probability. For each plaintext $P_1$ in $S_1$ and for each key guess, we compute $P_2 = E_f^{-1}(E_f(P_1) \oplus \Delta_i)$ such that $E_f(P_1) \oplus E_f(P_2) = \Delta_i$. So, for each key guess, two plaintexts form a pair, and we have $D/2$ pairs with the desired difference at the input of our distinguisher. For the right key guess, if the distinguisher probability is $p$, we have $\lambda_R = p \times D/2$ pairs that satisfy $\Delta_o$. By construction, all these pairs belong to the structures and pass the filters. For the wrong key guesses, the probability that they have a fixed difference $\Delta_o$ at the output is $1/2^n$, and the counter is expected to be $\lambda_W = D/2^{n+1}$ on average.

Similar to [9, 13, 14], we apply the Poisson distribution as the statistical model and denote the corresponding cumulative distribution function as $F_W$ and $F_R$. The probability that a counter associated with the wrong key guess is greater than threshold $s$ is $1 - F_W(s)$, and the expected number of counters greater than $s$ is $2^{\kappa_g}(1 - F_W(s))$. Due to the linear key schedule of SIMON, we can reconstruct the master key candidates from any subkey bits using linear algebra. The cost of reconstructing the master keys is $2^{\kappa_g} \cdot (1 - F_W(s)) \times 2^{\kappa - \kappa_g} = 2^{\kappa} \cdot (1 - F_W(s))$. The time complexity and success probability are:

$$C_1 = D + 2^{\kappa_g} \cdot \lambda_W + 2^{\kappa} \cdot (1 - F_W(s))$$
$$P_S = 1 - F_R(s)$$

## 5.2 25-Round Key-Recovery on SIMON48/72

We apply the method with the differential $(\emptyset, \{0\})$ to $(\{0\}, \emptyset)$ covering 17 rounds with probability $p = 2^{-45.49}$, and append three rounds before and five rounds after it. We show the detail of the bits to guess round by round in Table 6. From the outer to the inner, we use the sufficient bit conditions from Table 5 to guess the key bits that lead to the desired differences. When possible, we use the guessed key bits to deduce other key bits using the key schedule. In the rightmost column, we detail the time complexity of each step starting from $2^t$ pairs. In total, the complexity of guessing the key bits leading to $\Delta_i$ and $\Delta_o$, and incrementing the corresponding counters is $2^{t+32}$. During this step, $\kappa_g = 69$ bits, from the first subkeys to the last subkeys are guessed.

*Attack Parameters.* If the data $D = 2^{47}$ is taken, knowing that $l_1 = 41$ and $l_{24} = 18$, we split the data into $2^{40}$ structures of $2^7$ plaintexts and after constructing our pairs of structures and filtering the ciphertexts $C$, there remain $2^{40-1} \times 2^{7\times 2}/2^{18} = 2^{35}$ pairs. So $t = 35$ and the time complexity for the counter incrementing part is $2^{t+32} = 2^{67}$. The average value for the counter of the right key guess is $\lambda_R = p \times D/2 = 2^{0.51}$. For a bad key guess, we expect the counter to be close to $\lambda_W = D/2^{48+1} = 2^{-2}$. We choose the subkeys whose counts are greater than 1, the complexity is $2^{67} + 2^{66.76} \approx 2^{67.89}$ with a success probability of 42%.

We show parameters for different variants of SIMON in Table 7.

**Table 5**: Extended path for 25 rounds of SIMON48/72, where bold bits represent the sufficient bit conditions.

| $r$ | Differential path | | $l_i$ |
|---|---|---|---|
| 0 | 0000000*00000**00001**01 | 00000**0000***0*01***0** | 32 |
| 1 | 000000000000000*000001*0 | 0000000*00000**00001**01 | 41 |
| 2 | 00000000000000000000001 | 000000000000000*000001*0 | 46 |
| 3 | 00000000000000000000000 | 00000000000000000000001 | 48 |
| | 17-round differential ($3 \rightarrow 20$) | | |
| 20 | 00000000000000000000001 | 00000000000000000000000 | 48 |
| 21 | 000000000000000*000001*0 | 00000000000000000000001 | 46 |
| 22 | 0000000*00000**00001**01 | 000000000000000*000001*0 | 41 |
| 23 | 00000**0000***0*01***0** | 0000000*00000**00001**01 | 32 |
| 24 | 000***0*0*************** | 00000**0000***0*01***0** | 18 |
| 25 | 0********************** | 000***0*0*************** | 6 |

**Table 6**: Details of the bits to guess round by round and the corresponding complexity when starting from $2^t$ pairs.

| R | Bits to guess | total | #cond. | Complexity |
|---|---|---|---|---|
| 23 | $k^{(24)}_{0,3,4,5,6,7,10,11,12,13,16,17,18,19,20,21,22,23}$, | 18 | 14 | $2^t \cdot 2^{18-14} = 2^{t+4}$ |
| 22 | $k^{(23)}_{2,3,4,9,10,11,16,17,19,20,21,23}$, $k^{(24)}_{1,2,8,9,15}$ | 17 | 9 | $2^{t+4} \cdot 2^{17-9} = 2^{t+12}$ |
| 21 | $k^{(24)}_{14}$, $k^{(23)}_{0,1,6,7,8,13,14,18}$, $k^{(22)}_{1,8,9,15,18,19}$ | 15 | 5 | $2^{t+12} \cdot 2^{15-5} = 2^{t+22}$ |
| 20 | $k^{(23)}_{5,15,22}$, $k^{(22)}_{5,6,16,23}$, deduced $k^{(21)}_{7,17}$ | 7 | 2 | $2^{t+22} \cdot 2^{7-2} = 2^{t+27}$ |
| 2 | $k^{(0)}_{1,8,9,15,18,19}$ | 6 | 5 | $2^{t+27} \cdot 2^{6-5} = 2^{t+28}$ |
| 3 | $k^{(0)}_{5,6,16,23}$, $k^{(1)}_{7,17}$ | 6 | 2 | $2^{t+28} \cdot 2^{6-2} = 2^{t+32}$ |

# 6 Key-Recovery Attacks Using Linear Cryptanalysis

In this section, we detail our key recovery attack using linear cryptanalysis. We follow the description of a last-round key recovery given by Matsui's Algorithm 2 [21] and consider a linear approximation $P' \cdot \alpha \oplus C' \cdot \beta$ where $P'$ and $C'$ the intermediate values after a few rounds of encryption/decryption. Given a set of $D$ known plaintext/ciphertext pairs $(P, C)$, we can compute the intermediate values $P'$ and $C'$ for each partial key guess $\kappa_g = (\kappa_p, \kappa_f, \kappa_b, \kappa_c)$ for the first and/or last rounds, and compute the experimental correlation of the linear approximation

$$\begin{aligned} q\left(k_p, k_f, k_b, k_c\right) &= \frac{1}{D}(\#\left\{P, C : P' \cdot \alpha \oplus C' \cdot \beta = 0\right\} \\ &\quad - \#\left\{P, C : P' \cdot \alpha \oplus C' \cdot \beta = 1\right\}) \\ &= \frac{1}{D} \sum_{P,C} (-1)^{P' \cdot \alpha \oplus C' \cdot \beta} \end{aligned}$$

**Table 7**: Attack parameters for differential attacks. $P_S$ is success probability. We set the threshold $s = 1$.

| Variant | Rounds | $\kappa_g$ | $D$ | $\lambda_R$ | $\lambda_W$ | $P_S$ | Time |
|---|---|---|---|---|---|---|---|
| 48/72 | $25 = 3 + 17 + 5$ | 69 | $2^{47.5}$ | $2^{1.01}$ | $2^{-1.5}$ | 0.60 | $2^{68.58}$ |
| 48/72 | $25 = 3 + 17 + 5$ | 69 | $2^{47}$ | $2^{0.51}$ | $2^{-2}$ | 0.42 | $2^{67.89}$ |
| 48/96 | $26 = 4 + 17 + 5$ | 94 | $2^{47.5}$ | $2^{1.01}$ | $2^{-1.5}$ | 0.60 | $2^{93.14}$ |
| 48/96 | $26 = 4 + 17 + 5$ | 94 | $2^{47}$ | $2^{0.51}$ | $2^{-2}$ | 0.42 | $2^{92.51}$ |
| 64/96 | $31 = 4 + 23 + 4$ | 92 | $2^{63.5}$ | $2^{1}$ | $2^{-1.5}$ | 0.59 | $2^{92.20}$ |
| 64/96 | $31 = 4 + 23 + 4$ | 92 | $2^{63}$ | $2^{0.5}$ | $2^{-2}$ | 0.41 | $2^{91.43}$ |
| 64/128 | $32 = 4 + 23 + 5$ | 124 | $2^{63.5}$ | $2^{1}$ | $2^{-1.5}$ | 0.59 | $2^{124.58}$ |
| 64/128 | $32 = 4 + 23 + 5$ | 124 | $2^{63}$ | $2^{0.5}$ | $2^{-2}$ | 0.41 | $2^{123.89}$ |
| 96/144 | $42 = 4 + 33 + 5$ | 124 | $2^{96}$ | $2^{0.9}$ | $2^{-1}$ | 0.56 | $2^{140.53}$ |
| 128/128 | $50 = 4 + 42 + 4$ | 105 | $2^{127}$ | $2^{1.02}$ | $2^{-2}$ | 0.60 | $2^{127.07}$ |
| 128/192 | $51 = 5 + 42 + 4$ | 159 | $2^{127.5}$ | $2^{1.52}$ | $2^{-1.5}$ | 0.78 | $2^{187.67}$ |
| 128/192 | $51 = 5 + 42 + 4$ | 159 | $2^{127}$ | $2^{1.02}$ | $2^{-2}$ | 0.60 | $2^{186.76}$ |
| 128/256 | $52 = 5 + 42 + 5$ | 208 | $2^{127}$ | $2^{1.02}$ | $2^{-2}$ | 0.60 | $2^{250.76}$ |

The value of $P' \cdot \alpha$ and $C' \cdot \beta$ is computed as a function of the partial key guess and some bits in plaintext and ciphertext, and we denote the masked bits as $\chi_p(P)$ and $\chi_c(C)$. Then, we have:

$$P' \cdot \alpha = f\left(k_f, k_p \oplus \chi_p(P)\right)$$
$$C' \cdot \beta = g\left(k_b, k_c \oplus \chi_c(C)\right)$$

## 6.1 FWT Approach [22, 23]

Since the value of $P' \cdot \alpha$ and $C' \cdot \beta$ do not depend on the full plaintext/ciphertext, we can compress the dataset using a distillation phase [21] where we only count the number of plaintext/ciphertext pairs that reach each value of $P'$ and $C'$, then we correlate $q\left(k_p, k_f, k_b, k_c\right)$

$$= \frac{1}{D} \sum_{P,C} (-1)^{f(k_f, k_p \oplus \chi_p(P)) \oplus g(k_b, k_c \oplus \chi_c(C))}$$
$$= \frac{1}{D} \sum_{i \in \mathbb{F}_2^{\kappa_p}} \sum_{j \in \mathbb{F}_2^{\kappa_c}} \#\left\{P, C : \chi_p(P) = i, \chi_c(C) = j\right\}$$
$$\times (-1)^{f(k_f, k_p \oplus i) \oplus g(k_b, k_c \oplus j)}$$

where $i$ and $j$ denote the masked value of plaintext and ciphertext. Then, we can find that the expression above is a convolution:

$$\begin{aligned}&\frac{1}{D}\sum_{i,j}\phi(i,j)\times\psi_{k_f,k_b}\left(k_p\oplus i,k_c\oplus j\right)\\&=\frac{1}{D}\left(\phi*\psi_{k_f,k_b}\right)\left(k_p,k_c\right)\end{aligned}$$

where

$$\begin{aligned}\phi(x,y)&=\#\left\{P,C:\chi_p(P)=x,\chi_c(C)=y\right\}\\\psi_{k_f,k_b}(x,y)&=(-1)^{f(k_f,x)\oplus g(k_b,y)}\end{aligned}$$

Therefore, for a given $k_f$, $k_b$, we can evaluate $q(k_p,k_f,k_b,k_c)$ for all $k_p$, $k_c$ with complexity $\tilde{\mathcal{O}}(2^{\kappa_p+\kappa_c})$ using a Fast Walsh Transform, which is first observed in [22], and then generalized in [23]. The time complexity of the analysis phase is reduced to $\tilde{\mathcal{O}}(D+2^{\kappa_g})$.

## 6.2 Statistical Models to Estimate Success Probability

We follow the work of Blondeau and Nyberg [24, 25] to estimate the success probability of the attack. Similarly, the sampling model with a factor $B$ depends on the type of attack: $B=1$ if the plaintexts are randomly chosen with repetition, and $B=(2^n-D)/(2^n-1)$ if they are distinct (we assume distinct plaintext in the rest of this attack).

When using a single linear hull with empirical ELP, the correlations for the right and wrong keys follow normal distributions with parameters:

$$\begin{aligned}\mu_R&=0, & \sigma_R^2&=B/D+\mathrm{ELP},\\\mu_W&=0, & \sigma_W^2&=B/D+2^{-n}.\end{aligned}$$

The distributions are both centered on zero, and the variance for the right key is greater. Then, we can sort the keys according to the absolute value of the measured correlation, and expect a larger value for the right key than for the wrong key. More precisely, using a threshold $s=\sigma_W\Phi^{-1}\left(1-2^{-a-1}\right)$ on the absolute value of the correlation, the success probability is given by [24, Theorem 2]:

$$P_S=2-2\Phi\left(\frac{\sigma_W}{\sigma_R}\Phi^{-1}\left(1-2^{-a-1}\right)\right),$$

where $a$ denotes the advantage of the attack, and $\Phi$ is the cumulative distribution function of the standard normal distribution, respectively.

## 6.3 Attack Parameters

We compute the attack parameters against different variants of `SIMON` and summarize in Table 8. Then, we explain one attack in detail. Due to the rotation invariant of

SIMON, we can repeat the attack by using the rotated distinguisher to improve the success probability, according to [9]. We expect the attack to succeed after $\frac{1}{P_S}$ attempts with average time complexity $C_1 \times \frac{1}{P_S}$, where $C_1$ is the time complexity to run a single time. To estimate the success probability with a binomial distribution, we have the success probability after $\frac{1}{P_S}$ attempts is $1-(1-P_S)^{\frac{1}{P_S}}$.

**Table 8**: Attack parameters for linear attacks on SIMON with advantage $a = 10$, $D = 2^{n-1}$. All the attacks used the distinguishers in Table 1.

| Variant | Rounds | Source | $\kappa_p, \kappa_f, \kappa_b, \kappa_c$ | Capacity | $C_1$ | $P_S$ | Time | $P$ |
|---|---|---|---|---|---|---|---|---|
| 48/72 | 26=5+17+4 | 17-bit DW | 29,7,2,16 | $2^{-45.19}$ | $2^{54}+2^{62}$ | 0.12 | $2^{65.06}$ | 66% |
| 48/96 | 27=5+17+5 | 17-bit DW | 29,7,7,29 | $2^{-45.19}$ | $2^{72}+2^{86}$ | 0.12 | $2^{89.05}$ | 66% |
| 64/96 | 32=5+23+4 | 17-bit DW | 43,10,2,24 | $2^{-60.24}$ | $2^{79}+2^{86}$ | 0.24 | $2^{88.10}$ | 68% |
| 64/128 | 33=5+23+5 | 17-bit DW | 43,10,10,43 | $2^{-60.24}$ | $2^{106}+2^{118}$ | 0.24 | $2^{120.09}$ | 68% |
| 96/96 | 44=5+34+5 | 17-bit DW | 30,7,9,38 | $2^{-93.74}$ | $2^{84}+2^{86}$ | 0.07 | $2^{90.09}$ | 65% |
| 96/144 | 45=6+33+6 | 17-bit DW | 47,18,18,47 | $2^{-91.74}$ | $2^{130}+2^{134}$ | 0.31 | $2^{135.77}$ | 70% |
| 128/128 | 53=5+43+5 | 17-bit DW | 47,14,12,43 | $2^{-124.59}$ | $2^{116}+2^{118}$ | 0.19 | $2^{120.72}$ | 67% |
| 128/256 | 56=6+43+7 | 17-bit DW | 64,33,55,78 | $2^{-124.59}$ | $2^{230}+2^{246}$ | 0.19 | $2^{248.40}$ | 67% |

Search Time: the time of searching linear hull utilized in the attack;
$C_1$: the time complexity to run the attack a single time;
$P_S$: success probability to run the attack a single time;
Time= $C_1 \times \frac{1}{P_S}$: the average time by assuming that the attack is repeated $\frac{1}{P_S}$ times;
$P$: success probability after $\frac{1}{P_S}$ attempts.

*Key recovery attack on 26-round SIMON48/72.* We apply the FWT approach to SIMON48/72, with the 17-round linear approximation $(\{23\},\emptyset) \rightarrow (\emptyset,\{23\})$ with capacity $2^{-45.19}$, and add five rounds before the distinguisher and four rounds after. Following [9, Algorithm 3], $x_{23}^{(6)}$ can be computed from $\kappa_p = 29$ bits of $P$, $\kappa_p = 29$ bits of the whitening key $k^{(0)}||k^{(1)}$, and $\kappa_f = 7$ additional key bits. Similarly, $x_{23}^{(r-4)}$ can be computed from $\kappa_c = 16$ bits of $C$, $\kappa_c = 16$ bits of the whitening key $k^{(r-1)}||k^{(r-2)}$, and $\kappa_b = 2$ additional key bits:

$$
\begin{aligned}
k_p &= k^{(0)}_{[22,21,20,19,\ldots,9,7,5,4,3]}, k^{(1)}_{[23,22,20,19,18,15,13,12,11,6,5]} \\
k_f &= k^{(2)}_{[21,20,14,13,7]}, k^{(3)}_{[22,15]} \\
k_b &= k^{(r-3)}_{[22,15]} \\
k_c &= k^{(r-2)}_{[21,20,14,13,7]}, k^{(r-1)}_{[23,22,20,19,18,15,13,12,11,6,5]}
\end{aligned}
$$

We ignore bits that have a linear effect because they only flip the sign of the imbalance. The attack is decomposed in three phases:

*Distillation Phase.* Compute $\phi(x,y) = \#\{P,C : \chi_p(P) = x, \chi_c(C) = y\}$ for $0 \leq x < 2^{\kappa_p}, 0 \leq y < 2^{\kappa_c}$.

This step only requires setting up $2^{\kappa_p+\kappa_c}$ counters and to iterate over the $D$ available plaintext/ciphertext pairs.

*Analysis Phase.* For each $k_f, k_b$, compute $\psi_{\kappa_f,\kappa_b}(x,y) = (-1)^{f(\kappa_f,x)\oplus g(\kappa_b,y)}$ for $0 \le x < 2^{\kappa_p}, 0 \le y < 2^{\kappa_c}$, then evaluate the convolution $\left(\phi * \psi_{\kappa_f,\kappa_b}\right)$ using the Fast Walsh Transform.

For each $\kappa_f, \kappa_b$, this requires $2^{\kappa_p+\kappa_c}$ evaluations of $f$ and $g$ to generate $\psi_{\kappa_f,\kappa_b}$, and $3(k_p+k_c)\cdot 2^{\kappa_p+\kappa_c}$ additions and $2^{\kappa_p+\kappa_c}$ multiplications to evaluate the convolution, according to [23]. Assuming that the cost of $\kappa_p+\kappa_c$ additions and the cost of multiplication are comparable to the cost of one encryption, the total complexity of the analysis phase is $\mathcal{O}(2^{\kappa_g})$ using a memory of size $2^{\kappa_p+\kappa_c}$.

*Search Phase.* For all keys with $q(k_p,k_f,k_b,k_c) \ge s$, exhaustively try all master keys corresponding to the subkey candidates. Due to the linear key schedule of `SIMON`, we can reconstruct the master key from the $2^{\kappa_g-a}$ candidates using linear algebra. With a threshold $s = F_W^{-1}(1-2^{-a})$, we expect a fraction $2^{-a}$ of the keys to remain. Then, we exhaustively search the remaining key bits, with a complexity of $2^{\kappa_g-a}\times 2^{\kappa-\kappa_g} = 2^{\kappa-a} \approx 2^{72-a}$.

Using the Walsh transform pruning technique of [23] (and partially precomputing the Walsh transform of $\psi$), the complexity of the analysis phase is reduced to[2]:

$$\begin{aligned}
&\rho_A\left(\kappa_p+\kappa_c\right)2^{\kappa_p+\kappa_c} + 2\rho_M 2^{\kappa_p+\kappa_f+\kappa_b+\kappa_c-l_{12}}\\
&+\rho_A 2^{\kappa_f+\kappa_b+\kappa_c-l_{12}}\left(2^{\kappa_p}+\left(\kappa_p-l_0-1\right)2^{\kappa_p-l_0}\right)\\
&+\rho_A 2^{\kappa_p-l_0+\kappa_f+\kappa_b-l_{12}}\left(2^{\kappa_c}+\left(\kappa_c-l_3-1\right)2^{\kappa_c-l_3}\right)\\
&=45\rho_A 2^{45}+2\rho_M 2^{54}+\rho_A 2^{25}(2^{29}+27\times 2^{28})\\
&+\rho_A 2^{38}(2^{16}+15\times 2^{16}) \approx 2\rho_M 2^{54},
\end{aligned}$$

with $\rho_A$ the cost of an addition, and $\rho_M$ the cost of a multiplication. Assuming that two multiplications correspond to roughly one evaluation of the cipher, we end up with a complexity of $2^{\kappa_g}$. This variant uses a memory of $2^{\kappa_p+\kappa_c}+2^{\kappa_p+\kappa_f}+2^{\kappa_c+\kappa_b} = 2^{45}+2^{36}+2^{18}\approx 2^{45}$ elements.

Following the above attack parameter, we obtain the complexity of the analysis phase is $2^{54}$. With an advantage of $a=10$, the complexity of the search phase is $2^{62}$, and the success probability $P_S = 0.12$. After $\frac{1}{P_S} = 8.3$ attempts, the success probability is $1-(1-P_S)^{\frac{1}{P_S}} \approx 66\%$, and the average time complexity of this attack is $2^{65.06}$.

# 7 Discussion and Perspective

As shown in Fig.1, we observe that the `DW` of the `SIMECK32` is almost the same as the `SW`. For better compare `DW` to `SW`, we also evaluated other `SIMON`-like ciphers with different rotate constant.

*`SIMON`-like ciphers with different rotate constant.* In Crypto 2015, Kölbl *et al.* explored `SIMON`-like ciphers with different rotation constants. After the analysis of

[2] With the notation adopted by [23], we have $k_0 = 29, k_1 = 7, k_2 = 2, k_3 = 16, l_{12} = 0, l_0 = 1, l_3 = 0$.

**Table 9**: Probability of a 17-round differential $(\emptyset, \{0\}) \rightarrow (\{0\}, \emptyset)$ evaluated under different windows for $\texttt{SIMON48}_{(12,5,3)}$ and $\texttt{SIMON48}_{(7,0,2)}$.

| $\texttt{SIMON48}n_{(12,5,3)}$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $w$-bit | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| EDP(SW) | -inf | -inf | -inf | -inf | -66.89 | -66.49 | -64.63 | -59.89 | -59.65 |
| EDP(DW) | -inf | -inf | -48.13 | -47.47 | -47.34 | -47.14 | -47.02 | -46.91 | -45.99 |
| $\texttt{SIMON48}n_{(7,0,2)}$ | | | | | | | | | |
| $w$-bit | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| EDP(SW) | -inf | -inf | -inf | -inf | -51.95 | -50.70 | -49.05 | -48.10 | -47.62 |
| EDP(DW) | -inf | -inf | -50.86 | -49.01 | -48.78 | -47.75 | -47.48 | -46.70 | -46.26 |

the diffusion, differential, and linear properties of these ciphers, they screened three candidate rotation constants: $(12, 5, 3)$, $(1, 0, 2)$, and $(7, 0, 2)$. Next, we will call the cipher with rotation constants $(a, b, c)$ and block size $n$ as $\texttt{SIMON}n_{(a,b,c)}$. To further compare DW with SW, we show in Table 9 the probability of the 17-round differential $(\emptyset, \{0\})$ to $(\{0\}, \emptyset)$ for $\texttt{SIMON48}_{(12,5,3)}$ and $\texttt{SIMON48}_{(7,0,2)}$ for different window sizes. Whatsmore, for the variant $\texttt{SIMON}n_{(1,0,2)}$, which can not be evaluated by SW due to the rotation constant $(c > a, b)$, DW can adjust the output difference or mask and trace the diffusion of the input to solve this problem.

Finally, for $\texttt{SIMON48}_{(1,0,2)}$, we rotate the output difference of the left branch to $(1 << ((r-1) \times 2)\%\frac{n}{2})$ and get a 19-round differential $(\emptyset, \{0\})$ to $(\{12\}, \emptyset)$ with EDP $2^{-43.13}$. We also get a 21-round differential $(\emptyset, \{0\})$ to $(\{16\}, \emptyset)$ with EDP $2^{-47.54}$. Impressively, we observe that the advantage of DW over SW is more significant when applied to large rotation constants $(12, 5, 3)$. Compared to SW, DW better exploits the diffusion property of the cipher and guarantees the minimal loss of trails in the first few rounds.

*Compared to Dinur et al.'s work [26].* Recently, Dinur *et al.* proposed new, faster generic black-box algorithms for finding certain statistical properties (high-probability differentials, linear biases, boomerangs) in block ciphers [26]. All algorithms are based on "surrogate differentiation". According to [26, Table 1], the time complexity of finding the differential with EDP $\geq p$ is bounded by $\tilde{\mathcal{O}}(2^{n/2}p^{-1})$ costing memory $\tilde{\mathcal{O}}(2^{n/2}p^{-1})$. They also presented a memoryless version, while time complexity is increased to $\tilde{\mathcal{O}}(2^{n/2}p^{-2})$.

When applied to SIMON-like ciphers, our DW methodology exhibits several benefits over the technique presented in [26]:

1. DW **can detect low probability distinguisher.** The algorithm presented in [26, Algorithm 1] relies on a probability testing mechanism that is less adept at identifying low probability distinguishers due to a large amount of required data complexity.
2. DW **can find a key recovery friendly distinguisher.** When applied to SIMON-like ciphers, a distinguisher with low hamming weight input-output can mount a longer key recovery attack due to fewer key candidates. In [26], they detect all differential with probability greater than $p$, which may cause an unsatisfied distinguisher.

3. `DW` **has a constant growth in time complexity.** Furthermore, the time complexity of our approach is solely dependent on the window size $w$, and as demonstrated in Fig. 4, a small $w$ suffices to achieve satisfactory accuracy. For variants with larger block sizes, the time complexity of the method proposed in [26] enlarges exponents due to the increased value of $n$ and the decreased probability $p$. In contrast, the growth in complexity of our method is constant, due to the extended number of rounds $r$.

With high applicability, the methods developed by Dinur *et al.* are well-suited for constructing Differential Distribution Tables (DDT), Linear Approximation Tables (LAT), and Boomerang Connectivity Tables (BCT) of large Sbox. When applying dataset experiments, these structures are more likely to be revealed due to higher probability/correlation.

*Application to other kinds of ciphers.* For symmetric ciphers with different linear layers, for instance, SPN structures with MDS/almost MDS (Maximum Distance Separable) linear layers or Addition-RX algorithms, further research on the `DW` approach is required. Even for `SIMON`-like ciphers with larger rotation constant parameters, their diffusive properties are much weaker than those of SPN ciphers. For example, one active bit causes at most three active ones in each round.

The SPN cipher typically exhibits more rapid diffusion than the `SIMON`-like cipher, which is also because the latter is designed with an extreme focus on minimal hardware area. For SPN cipher like `AES`, which utilizes an MDS matrix as the linear layer, full diffusion occurs by the second round. This implies that a significant portion of active bits must be discarded early on with the dynamic window (`DW`) strategy, which can lead to bad results.

For those with an almost MDS matrix, the `DW` approach can reduce the searching space of the differential or linear hull. Once the input difference/mask is fixed, we can utilize `MLW` in Section 3.1.1 to narrow the search space by forcing the discarded bits to be zero. Meanwhile, it should be noted that determining a `DW` for these ciphers is not an easy task. The `MLW` works around full diffusion round, which means that we need to use `LWIM` to combine several `MLW` and form a `DW` with more rounds. For these ciphers, we believe combining `MLW` with SAT problems could yield better results, and we defer this task to future work.

# 8 Conclusion

In this paper, we follow Leurent *et al.*'s framework but adopt a dynamic way of choosing windows for each round. To determine these dynamic windows (`DW`), we propose the `MLW` (minimal loss window) and `LMIM` (link window in the middle) strategies. In our `MLW` strategy, the window is chosen to contain all possibly active bits as the number of them is no greater than the window size $w$. If the number of active bits is greater than $w$, we heuristically exclude some possibly active bits out of the window and restrict them as zero differences/masks. To maintain a lower loss of trails, these excluded bits are chosen using minimal active probability test. However, the test can only apply to the first few rounds (around the full diffusion). To solve this problem the `LWIM` strategy is proposed to link two short `MLW`s and form a full `DW`. With these

two strategies, we obtain the $w$-bit DW for SIMON and SIMECK. These windows are very different from the SW for SIMON, while being similar to those SW for SIMECK. Benefiting from these dynamic windows, we observed stronger differentials and linear hulls than previously proposed for almost all versions of SIMON. With these stronger distinguishers, we mounted the best key recovery attacks against SIMON. Moreover, we improved the previous analysis results on the small version of SIMON.

**Acknowledgements.** We thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work is supported by the National Key R&D Program of China (Grant No. 2024YFA1013000), the National Natural Science Foundation of China (Grant No. U2336207), and the National Cryptologic Science Fund of China (Grant No. 2025NCSF01013), and Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM114). Muzhou Li is also supported by the National Natural Science Foundation of China (Grant No. 62502277), and the Natural Science Foundation of Shandong Province. China (Grant No. ZR2024QF218).

# References

[1] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015, pp. 175–11756 (2015). https://doi.org/10.1145/2744769.2747946

[2] Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced simon and speck. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8540, pp. 525–545 (2014). https://doi.org/10.1007/978-3-662-46706-0_27

[3] Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8540, pp. 546–570 (2014). https://doi.org/10.1007/978-3-662-46706-0_28

[4] Chen, H., Wang, X.: Improved linear hull attack on round-reduced simon with dynamic key-guessing techniques. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 428–449 (2016). https://doi.org/10.1007/978-3-662-52993-5_22

[5] Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 161–185 (2015). https://doi.org/10.1007/978-3-662-47989-6_8

[6] Kölbl, S., Roy, A.: A brief comparison of simon and simeck. In: Bogdanov, A. (ed.) Lightweight Cryptography for Security and Privacy - 5th International Workshop, LightSec 2016, Aksaray, Turkey, September 21-22, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10098, pp. 69–88 (2016). https://doi.org/10.1007/978-3-319-55714-4_6

[7] Liu, Z., Li, Y., Wang, M.: Optimal differential trails in simon-like ciphers. IACR Trans. Symmetric Cryptol. **2017**(1), 358–379 (2017)

[8] Wang, X., Wu, B., Hou, L., Lin, D.: Automatic search for related-key differential trails in simon-like block ciphers based on MILP. In: Chen, L., Manulis, M., Schneider, S.A. (eds.) Information Security - 21st International Conference, ISC 2018, Guildford, UK, September 9-12, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11060, pp. 116–131 (2018). https://doi.org/10.1007/978-3-319-99136-8_7

[9] Leurent, G., Pernot, C., Schrottenloher, A.: Clustering effect in simon and simeck. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13090, pp. 272–302 (2021). https://doi.org/10.1007/978-3-030-92062-3_10

[10] Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The simeck family of lightweight block ciphers. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9293, pp. 307–329 (2015). https://doi.org/10.1007/978-3-662-48324-4_16

[11] Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. IACR Cryptol. ePrint Arch., 747 (2014)

[12] Abdelraheem, M.A., Alizadeh, J., Alkhzaimi, H.A., Aref, M.R., Bagheri, N., Gauravaram, P., Lauridsen, M.M.: Improved linear cryptanalysis of reduced-round simon. IACR Cryptol. ePrint Arch., 681 (2014)

[13] Wang, N., Wang, X., Jia, K., Zhao, J.: Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. Sci. China Inf. Sci. **61**(9), 098103–10981033 (2018)

[14] Qiao, K., Hu, L., Sun, S.: Differential security evaluation of simeck with dynamic key-guessing techniques. In: Camp, O., Furnell, S., Mori, P. (eds.) Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016, pp. 74–84 (2016).

https://doi.org/10.5220/0005684400740084

[15] Qin, L., Chen, H., Wang, X.: Linear hull attack on round-reduced simeck with dynamic key-guessing techniques. In: Liu, J.K., Steinfeld, R. (eds.) Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9723, pp. 409–424 (2016). https://doi.org/10.1007/978-3-319-40367-0_26

[16] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptol. ePrint Arch., 404 (2013)

[17] Nyberg, K.: Linear approximation of block ciphers. In: Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. Lecture Notes in Computer Science, vol. 950, pp. 439–444 (1994). https://doi.org/10.1007/BFb0053460

[18] Alizadeh, J., AlKhzaimi, H., Aref, M.R., Bagheri, N., Gauravaram, P., Kumar, A., Lauridsen, M.M., Sanadhya, S.K.: Cryptanalysis of SIMON variants with connections. In: Saxena, N., Sadeghi, A. (eds.) Radio Frequency Identification: Security and Privacy Issues - 10th International Workshop, RFIDSec 2014, Oxford, UK, July 21-23, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8651, pp. 90–107 (2014). https://doi.org/10.1007/978-3-319-13066-8_6

[19] Blondeau, C., Nyberg, K.: New links between differential and linear cryptanalysis. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 388–404 (2013). https://doi.org/10.1007/978-3-642-38348-9_24

[20] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4727, pp. 450–466 (2007). https://doi.org/10.1007/978-3-540-74735-2_31

[21] Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. Lecture Notes in Computer Science, vol. 765, pp. 386–397 (1993). https://doi.org/10.1007/3-540-48285-7_33

[22] Collard, B., Standaert, F., Quisquater, J.: Improving the time complexity of matsui's linear cryptanalysis. In: Nam, K., Rhee, G. (eds.) Information Security and Cryptology - ICISC 2007, 10th International Conference, Seoul, Korea, November 29-30, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4817, pp. 77–88 (2007). https://doi.org/10.1007/978-3-540-76788-6_7

[23] Flórez-Gutiérrez, A., Naya-Plasencia, M.: Improving key-recovery in linear attacks: Application to 28-round PRESENT. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 221–249 (2020). https://doi.org/10.1007/978-3-030-45721-1_9

[24] Blondeau, C., Nyberg, K.: Improved parameter estimates for correlation and capacity deviates in linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2016**(2), 162–191 (2016)

[25] Blondeau, C., Nyberg, K.: Joint data and key distribution of simple, multiple, and multidimensional linear cryptanalysis test statistic and its impact to data complexity. Des. Codes Cryptogr. **82**(1-2), 319–349 (2017)

[26] Dinur, I., Dunkelman, O., Keller, N., Ronen, E., Shamir, A.: Efficient Detection of High Probability Statistical Properties of Cryptosystems via Surrogate Differentiation. Cryptology ePrint Archive, Paper 2023/288. https://eprint.iacr.org/2023/288 (2023). https://eprint.iacr.org/2023/288

# A Different Window in This Paper

# B Discard Bits Determined by the MLW Strategy

**Table 10**: 16-bit window for 13-round `SIMECK32` given by `DW`.

| $R$ | Size(bits) | Rank |
|---|---|---|
| 0 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 1 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 2 | 16 | [1, 0, 5, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 3 | 16 | [0, 2, 1, 6, 5, 10, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15] |
| 4 | 16 | [1, 3, 2, 7, 5, 0, 6, 11, 10, 15, 4, 8, 9, 12, 13, 14] |
| 5 | 16 | [2, 6, 1, 0, 4, 3, 8, 7, 5, 12, 10, 11, 15, 9, 13, 14] |
| 6 | 16 | [3, 1, 7, 2, 5, 4, 9, 6, 0, 8, 11, 13, 12, 10, 15, 14] |
| 7 | 16 | [2, 6, 1, 0, 4, 3, 8, 7, 5, 12, 10, 11, 15, 9, 13, 14] |
| 8 | 16 | [1, 3, 2, 7, 5, 0, 6, 11, 10, 15, 4, 8, 9, 12, 13, 14] |
| 9 | 16 | [0, 2, 1, 6, 5, 10, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15] |
| 10 | 16 | [1, 0, 5, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 11 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 12 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 13 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |

**Table 11**: 16-bit window for 13-round `SIMON32` given by `DW`.

| $R$ | Size(bits) | Rank |
|---|---|---|
| 0 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 1 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 2 | 16 | [2, 8, 1, 0, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15] |
| 3 | 16 | [4, 0, 10, 3, 9, 2, 1, 5, 6, 7, 8, 11, 12, 13, 14, 15] |
| 4 | 16 | [6, 8, 1, 5, 12, 11, 4, 2, 10, 3, 0, 7, 9, 13, 14, 15] |
| 5 | 16 | [8, 0, 4, 3, 7, 14, 13, 9, 10, 6, 12, 2, 5, 11, 1, 15] |
| 6 | 16 | [10, 2, 8, 12, 9, 0, 5, 1, 15, 11, 6, 4, 14, 7, 13, 3] |
| 7 | 16 | [8, 0, 4, 3, 7, 14, 13, 9, 10, 6, 12, 2, 5, 11, 1, 15] |
| 8 | 16 | [6, 8, 1, 5, 12, 11, 4, 2, 10, 3, 0, 7, 9, 13, 14, 15] |
| 9 | 16 | [4, 0, 10, 3, 9, 2, 1, 5, 6, 7, 8, 11, 12, 13, 14, 15] |
| 10 | 16 | [2, 8, 1, 0, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15] |
| 11 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 12 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
| 13 | 16 | [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |

**Table 12**: 17-bit window for 17-round SIMON48 that generated by DW.

| $R$ | Size(bits) | Rank |
|---|---|---|
| 0 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 1 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 2 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 3 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 4 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,17,18] |
| 5 | 17 | [0,2,3,4,5,6,7,8,9,10,12,13,14,16,18,19,20] |
| 6 | 17 | [1,2,3,4,5,8,9,10,11,12,14,15,16,17,18,21,22] |
| 7 | 17 | [0,2,3,6,7,8,9,10,11,12,13,14,16,17,18,20,23] |
| 8 | 17 | [1,2,4,5,8,9,11,12,13,14,15,16,18,19,20,21,22] |
| 9 | 17 | [0,2,3,6,7,8,9,10,11,12,13,14,16,17,18,20,23] |
| 10 | 17 | [1,2,3,4,5,8,9,10,11,12,14,15,16,17,18,21,22] |
| 11 | 17 | [0,2,3,4,5,6,7,8,9,10,12,13,14,16,18,19,20] |
| 12 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,17,18] |
| 13 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 14 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 15 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 16 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 17 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |

**Table 13**: 17-bit window for 23-round SIMON64 given by DW (By appending three rounds on both sides).

| $R$ | Size(bits) | Rank |
|---|---|---|
| -3 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| -2 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| -1 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,16,20] |
| 0 | 17 | [1,2,4,5,6,8,9,10,11,12,14,15,16,17,18,21,22] |
| 1 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 2 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 3 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 4 | 17 | [0,1,3,4,5,6,8,10,11,12,13,14,15,16,17,18,24] |
| 5 | 17 | [0,2,3,4,5,6,7,8,9,10,12,13,14,16,19,20,26] |
| 6 | 17 | [1,2,4,5,8,9,10,11,12,14,15,16,17,18,20,21,22] |
| 7 | 17 | [0,3,6,7,8,9,10,11,12,13,14,16,17,18,20,23,24] |
| 8 | 17 | [1,4,5,8,9,11,12,13,14,15,16,18,19,20,21,25,26] |
| 9 | 17 | [0,3,6,7,8,9,10,11,12,13,14,16,17,18,20,23,24] |
| 10 | 17 | [1,2,4,5,8,9,10,11,12,14,15,16,17,18,20,21,22] |
| 11 | 17 | [0,2,3,4,5,6,7,8,9,10,12,13,14,16,19,20,26] |
| 12 | 17 | [0,1,3,4,5,6,8,10,11,12,13,14,15,16,17,18,24] |
| 13 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 14 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 15 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 16 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 17 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 17+1 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 17+2 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 17+3 | 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |

**Table 14**: 17-bit window for 33-round SIMON96 given by DW (By iterating two times). 17-bit window for first 33-round of SIMON128.

| $R$ | Rank |
|---|---|
| 0 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 1 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 2 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 3 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 4 | [0,1,3,4,5,6,8,10,11,12,13,14,15,16,17,18,24] |
| 5 | [0,2,3,4,5,6,7,8,9,10,12,13,14,16,19,20,26] |
| 6 | [1,2,4,5,8,9,10,11,12,14,15,16,17,18,20,21,22] |
| 7 | [0,3,6,7,8,9,10,11,12,13,14,16,17,18,20,23,24] |
| 8 | [1,4,5,8,9,11,12,13,14,15,16,18,19,20,21,25,26] |
| 9 | [0,3,6,7,8,9,10,11,12,13,14,16,17,18,20,23,24] |
| 10 | [1,2,4,5,8,9,10,11,12,14,15,16,17,18,20,21,22] |
| 11 | [0,2,3,4,5,6,7,8,9,10,12,13,14,16,19,20,26] |
| 12 | [0,1,3,4,5,6,8,10,11,12,13,14,15,16,17,18,24] |
| 13 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 14 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 15 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 16 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 17 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 18 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 19 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 20 | [0,1,3,4,5,6,8,10,11,12,13,14,15,16,17,18,24] |
| 21 | [0,2,3,4,5,6,7,8,9,10,12,13,14,16,19,20,26] |
| 22 | [1,2,4,5,8,9,10,11,12,14,15,16,17,18,20,21,22] |
| 23 | [0,3,6,7,8,9,10,11,12,13,14,16,17,18,20,23,24] |
| 24 | [1,4,5,8,9,11,12,13,14,15,16,18,19,20,21,25,26] |
| 25 | [0,3,6,7,8,9,10,11,12,13,14,16,17,18,20,23,24] |
| 26 | [1,2,4,5,8,9,10,11,12,14,15,16,17,18,20,21,22] |
| 27 | [0,2,3,4,5,6,7,8,9,10,12,13,14,16,19,20,26] |
| 28 | [0,1,3,4,5,6,8,10,11,12,13,14,15,16,17,18,24] |
| 29 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 30 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 31 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 32 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |
| 33 | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16] |

captionComparison of the discard bits on SIMON64 using the 17-bit window, where bit position red labeled discarded by us. Note the DW for SIMON96 is the same.

| Round | Diffusion bit position with input difference $(0, 1)$ | #Discard Bits | #Combi. |
|---|---|---|---|
| 0 | [] | 0 | 0 |
| 1 | [0] | 0 | 0 |
| 2 | [1,2,8] | 0 | 0 |
| 3 | [0,2,3,4,9,10,16] | 0 | 0 |
| 4 | [1,3,4,5,6,8,10,11,12,17,18,24] | 0 | 0 |
| 5 | [0,2,3,4,5,6,7,8,9,10,11,12,13,14,16,18,19,20,25,26] | 3 | $\binom{20}{3}$ |
| 6 | [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,26,27,28] | 9 | $\binom{26}{9}$ |
| 7 | [0,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30] | 13 | $\binom{30}{13}$ |
| 8(full) | [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31] | 15 | $\binom{32}{15}$ |