An Efficient Strategy to Construct a Better Differential on Multiple-Branch-Based Designs: Application to Orthros

Kazuma Taka[1], Tatsuya Ishikawa[2], Kosei Sakamoto[1], and Takanori Isobe[1]

[1] University of Hyogo, Kobe, Japan.
ad22c034@gsis.u-hyogo.ac.jp, k.sakamoto0728@gmail.com,
takanori.isobe@ai.u-hyogo.ac.jp
[2] WDB KOUGAKU Co.,Ltd, Tokyo, Japan. t.ishikawa037@gmail.com.

Abstract. As low-latency designs tend to have a small number of rounds to decrease latency, the differential-type cryptanalysis can become a significant threat to them. In particular, since a multiple-branch-based design, such as Orthros can have the strong clustering effect on differential attacks due to its large internal state, it is crucial to investigate the impact of the clustering effect in such a design. In this paper, we present a new SAT-based automatic search method for evaluating the clustering effect in the multiple-branch-based design. By exploiting an inherent trait of multiple-branch-based designs, our method enables highly efficient evaluations of clustering effects on this-type designs. We apply our method to the low-latency PRF Orthros, and show a best differential distinguisher reaching up to 7 rounds of Orthros with $2^{116.806}$ time/data complexity and 9-round distinguisher for each underlying permutation which is 2 more rounds than known longest distinguishers. Besides, we update the designer's security bound for differential attacks based on the lower bounds for the number of active S-boxes, and obtain the optimal differential characteristic of Orthros, Branch 1, and Branch 2 for the first time. Consequently, we improve the designer's security bound from 9/12/12 to 7/10/10 rounds for Orthros/Branch 1/Branch 2 based on a single differential characteristic.

Keywords: Differential cryptanalysis · Clustering effect · Multiple-branch-based designs · Orthros · SAT-based automatic search method.

1 Introduction

The design of lightweight cryptography is one of the prime topics in the field of symmetric cryptography, particularly since the emergence of the first lightweight block cipher PRESENT [9]. Many lightweight proposals tend to put effort into reducing the hardware circuit size as small as possible similar to PRESENT. Aside from minimizing the hardware circuit, minimizing the latency of the overall design has also become an area of emphasis. Since a quick response time of encryption is desirable for some applications, such as automotive communication,

memory bus encryption, and industrial control network, low-latency designs are recently getting more attention.

PRINCE, proposed by Borghoff et al. [10], is the first low-latency design that has reflection construction based on the substitution-permutation network (SPN). A low latency tweakable block cipher QARMA, proposed by Avanzi [2], follows this design strategy, and both PRINCE and QARMA realize very small latency. MIDORI, proposed by Banik et al. [4], is an SPN-based block cipher targeting low-energy applications, while its latency is quite small. Since SPN-based designs seem more promising in terms of latency than Feistel-based design, several other low-latency designs, such as Mantis [7], Orthros [5], SPEEDY [16] also have an SPN-based construction.

For these low-latency designs, a thorough security analysis is essential, as these designs typically feature a small number of rounds to achieve low latency. Among the variety of attack vectors, a differential-type cryptanalysis has emerged as the most significant threat for low-latency designs because the growth of the differential probability is not sufficient at the beginning of the rounds. In fact, the best attack on the first low-latency design PRINCE is a (multiple) differential cryptanalysis, and one variant of SPEEDY and MANTIS are broken by the differential cryptanalysis [11, 13]. Besides, the designers of Orthros and SPEEDY pay a lot of effort into ensuring a resistance against the differential cryptanalysis. Given these facts, a thorough security analysis of differential-type cryptanalysis is essential for such low-latency designs.

Among the low-latency designs, Orthros has an interesting construction in which the output is computed by summing the outputs of two keyed permutations. Such two-branch-based designs do not have a decryption function, namely, these designs are PRF not PRP, but they can still be applied into many popular modes, e.g., CTR, CMAC, and GCM. The advantage of a two-branch construction in terms of security is that it is difficult to add the key-recovery rounds for the attacker, as discussed in [5]. This means that additional rounds required for a security margin can be small in these designs, which directly results in a reduction in latency. Therefore, such multiple-branch-based designs seem promising for the construction of future ultra-low-latency PRFs.

A downside of such a two-branch-based construction is the difficulty in evaluating their security. Specifically, Orthros is based on two "weak" keyed permutations, i.e., each keyed permutation cannot be used as a standalone PRP by itself. This makes a discussion in the context of the provable security so hard that the authors of Orthros carefully investigated the security of the sum of permutations from the perspective of cryptanalysis [5]. In the designer's analysis, the most powerful attack on Orthros is the integral cryptanalysis, which can distinguish up to 7 rounds. For the differential cryptanalysis, they only presented the lower bound for the number of active S-boxes (AS) for each branch independently, and provide the lower bound for # AS as the sum of them. More specifically, they independently evaluate the lower bound for # AS in the first four rounds in a bit-wise level and the remaining eight rounds in a nibble-wise level for each Branch 1 and Branch 2. Then, they provide the lower bound for # AS of Orthros

as the sum of these independent four lower bounds. Hence, the provided security bound is rough in their work. Additionally, they only considered a single characteristic, not taking the clustering effect into consideration in their work. Given that the two-branch-based construction seems easy to happen the clustering effect due to a large space in its internal state, evaluating the clustering effect on such construction is of great importance.

Our Contribution. In this paper, we study how to efficiently evaluate the clustering effect on multiple-branch-based designs such as Orthros. With the SAT-based automatic search tool for differential characteristics proposed by Sun et al. [19], we can efficiently evaluate the optimal differential characteristic. However, evaluating the clustering effect is challenging task, particularly for the designs with a large state size, such as multiple-branch-based designs. To address this issue, we propose a new method for efficiently evaluating the clustering effect on multiple-branch-based designs by exploiting an inherent trait of these designs. Our main contributions are as follows:

- We present a SAT-based automatic search method for evaluating the clustering effect on multiple-branch-based designs. This method can evaluate the clustering effect on a given pair of input and output differences, which is called *differential* in literature, not only two-branch-based designs such as Orthros, but also multiple-branch-based designs without limitation of the number of branches. A general approach to evaluate the clustering effect by automatic search tools is to count the differential characteristics of the entire construction under a given differential. The drawback of the general approach is that the computational cost will become heavy due to the large size of the internal state in a multiple-branch-based design. This drawback becomes more serious with the number of branches increasing. To address this issue, our method independently evaluates the clustering effect on each branch under a give differential. It allows us to efficiently obtain many differential characteristics that contribute to the probability of a given differential. While run-time is traditionally used as a metric to evaluate the efficiency of automatic search tools, this metric is highly dependent on the computational environment and mathematical solver used. Therefore, we introduce a new metric, "the number of invocations of a SAT solver ($\#SAT$)" to assess the efficiency of of the evaluation for the clustering effect by SAT. Since the evaluation of the clustering effect requires multiple invocations of a SAT solver, and these invocation dominates the most part of the evaluation, we can fairly assess the efficiency of each method by $\#SAT$ to a certain extent.
- We improve the designer's security bound of Orthros against the differential cryptanalysis. We first show the strict lower bound for # AS for the first time and update the designer's security bound based on # AS. More specifically, in the designer's evaluation, the 9-round Orthros is expected to resist differential cryptanalysis based on # AS, while we show that 8 rounds is enough. We also improve the designer's bound by 1 round for Branch 1 and Branch 2, both of which are the underlying keyed permutations of Orthros. Furthermore, we

reveal the optimal differential characteristics for up to 7 rounds of Orthros and full rounds of each branch for the first time. Our result shows that the distinguishing attack can be applied to 6/9/9 rounds of Orthros/Branch 1/Branch 2. Table 1 summarizes these results.

Table 1: Summary of our results for the AS-based evaluation and optimal differential characteristics to Orthros, Branch 1, and Branch 2.

Lower bounds for the number of active S-boxes													
Target \ Rounds	1	2	3	4	5	6	7	8	9	10	11	12 (full round)	Ref.
Branch 1	1	4	6	8	9	12	16	24	33	44	58	**68**	[5]
	1	4	6	8	11	18	28	37	48	58	**67**	80	Sect. 4.3
Branch 2	1	4	5	8	9	12	16	24	33	44	59	**68**	[5]
	1	4	5	8	10	16	26	36	49	58	**70**	80	Sect. 4.3
Orthros	2	8	12	16	18	24	36	56	**84**	88	117	136	[5]
	2	8	12	16	22	36	58	**79**	98	129	188	196	Sect. 4.3
Weight of optimal differential characteristics													
Branch 1	2	8	14	19	29	41	61	91	113	**142**	160	181	Sect. 4.3
Branch 2	2	8	13	19	26	38	58	82	117	**136**	163	180	Sect. 4.3
Orthros	4	16	29	42	59	90	**136**	-	-	-	-	-	Sect. 4.3

– We apply our method to 7 rounds of Orthros whose the probability of the optimal differential characteristic is 2^{-136}. To demonstrate the efficiency of our method, we compare our method with the general one. As a result, our method yields a significant improvement, raising the probability of a differential corresponding to the optimal differential characteristic from 2^{-136} to $2^{-116.806}$, whereas the conventional method can only achieve $2^{-127.395}$. Moreover, our method improves # SAT and a practical run-time 93.6% and 99.5% in comparison to the general method, respectively. It should be mentioned that our result is the best distinguishing attack to Orthros. Table 2 shows the result of our method in comparison with the previous distinguishing attack to Orthros.

Table 2: Summary of the distinguishing attacks to Orthros, Branch 1, and Branch 2.

Target	Round	Method	Time/Data	Ref.
Branch1	7	Integral	$2^{127.0}$	[5]
	9	Differential	$2^{113.0}$	Sect. 4.3
Branch2	7	Integral	$2^{127.0}$	[5]
	9	Differential	$2^{117.0}$	Sect. 4.3
Orthros	7	Integral	2^{127}	[5]
	7	Differential	$2^{116.8}$	Sect. 4.6

As a multiple-branch-based design can dramatically decrease latency, it is a promising approach for the development of ultra-low-latency designs. Therefore, we believe that our method has the potential to be widely utilized in future multiple-branch-based designs and aid in the examination of the behavior of a differential in such designs.

Outline. The organization of this paper is as follows: In Sect. 2, we provide a brief explanation of differential cryptanalysis and the SAT-based automatic evaluation for differential characteristics and differentials. In Sect. 3, we first describe our target construction. We then introduce a new metric # SAT for assessing the efficiency of the evaluation of the clustering effect. Subsequently, we elaborate our SAT-based automatic method for evaluating the clustering effects in multiple-branch-based designs. In Sect. 4, we first evaluate the lower bound fot # AS for Orthros and each branch in Orthros, and search for the optimal differential characteristics for them. Then, we apply our and the general method to Orthros and compare the efficiency and probability. Additionally, we discuss the good parameters in our method and further improve the probability with a found good parameter. Finally, we conclude this paper in Sect. 5.

2 Preliminary

2.1 Differential Cryptanalysis

The differential cryptanalysis, proposed by Biham and Shamir, is one of the most powerful cryptanalysis techniques for symmetric-key primitives [8]. In the differential cryptanalysis, the attacker attempts to find a pair of input and output differences with a high probability, i.e., $E_K(\Delta P) = \Delta C,\ (\Delta C = C \oplus C', \Delta P = P \oplus P')$ occurs with high probability on a symmetric-key primitives E_k, where (P, P') and (C', C) denote a pair of plaintexts and ciphertexts, respectively. A pair of input and output differences $(\Delta P, \Delta C)$ is called a *differential* in the differential cryptanalysis. The probability of a differential, called a *differential probability*, is calculated by investigating all pairs of plaintext following $\Delta P = P \oplus P'$ on E_K. We define a differential and its probability on a symmetric-key primitive E_K as follows.

Definition 1 (Differential) *A differential is a pair of input and output differences. The probability of a differential* $(\Delta P, \Delta C)$ *is calculated as follows:*

$$DP(\Delta P \xrightarrow{E_K} \Delta C) = \Pr_P(E_K(P) \oplus E_K(P \oplus \Delta P) = \Delta C),$$

where P *are chosen from a uniformly distributed random variable.*

Generally, calculating such a probability is computationally infeasible in most symmetric-key primitives. Therefore, a *differential characteristic* is usually employed to estimate a differential probability. Let E_K be a r-round iterated block cipher as $E_K(\cdot) = f_r(\cdot) \circ f_{r-1}(\cdot) \circ \cdots \circ f_1(\cdot)$. A differential characteristic can

be defined as a sequence of differences over all rounds in E_K, and its probability can be estimated as a product of differential probabilities of each round under the well-known Markov cipher assumption [15]. We give the definition of a differential characteristic and its probability on a block cipher E_K as follows.

Definition 2 (Differential characteristic) *A differential characteristic is a sequence of differences over all rounds in a block cipher E_K as follows:*

$$\boldsymbol{C} = (\boldsymbol{c_0} \xrightarrow{f_1} \boldsymbol{c_1} \xrightarrow{f_2} \cdots \xrightarrow{f_r} \boldsymbol{c_r}) \coloneqq (\boldsymbol{c_0}, \boldsymbol{c_1}, \cdots, \boldsymbol{c_r}),$$

where $(\boldsymbol{c_0}, \boldsymbol{c_1}, \cdots, \boldsymbol{c_r})$ denotes differences of the output of each round, i.e., $\boldsymbol{c_0}$ and $\boldsymbol{c_r}$ denote differences of a plaintext and ciphertext, respectively. The probability of a differential characteristic $\boldsymbol{C}$ is estimated as follows:

$$DP(\boldsymbol{C}) = \prod_{i=1}^{r} DP(\boldsymbol{c_{i-1}} \xrightarrow{f_i} \boldsymbol{c_i}).$$

From the attacker aspect, the attacker is interested in only a differential, that is, information about internal differences is not necessary. Hence, the attacker can construct a differential by gathering the differential characteristics sharing the same $(\boldsymbol{c_0}, \boldsymbol{c_r})$ and try to enhance the probability of a differential $(\boldsymbol{c_0}, \boldsymbol{c_r})$. Such an endeavor is called "considering the *clustering effect*". In that case, we can view a differential $(\boldsymbol{c_0}, \boldsymbol{c_r})$ as a bunch of multiple differential characteristics. Therefore, the probability of $(\boldsymbol{c_0}, \boldsymbol{c_r})$ can be calculated by sum of probabilities of all differential characteristics constructing $(\boldsymbol{c_0}, \boldsymbol{c_r})$ as follows:

$$DP(\boldsymbol{c_0} \xrightarrow{E_K} \boldsymbol{c_r}) \approx \sum_{\boldsymbol{C} \in \boldsymbol{C_{all}}} DP(\boldsymbol{C}),$$

where $\boldsymbol{C_{all}}$ denotes the set of all differential characteristics constructing a differential $(\boldsymbol{c_0}, \boldsymbol{c_r})$.

From the viewpoint of the designer, guaranteeing the upper bound of $DP(\boldsymbol{C})$ is enough instead of showing the optimal differential characteristic. Many modern block ciphers take an approach to constructing non-linear layers only by an S-box. Let DP_s be the maximum differential probability of an S-box, we can estimate the upper bound of $DP(\boldsymbol{C})$ by the lower bound for # AS, i.e., $2^{-(DP_s \times \#AS)} \leq 2^{-n}$ is sufficient to resist against the distinguishing attack, where n denotes the block size. Nowadays, it is common to evaluate the optimal differential characteristic and the lower bound for # AS with automatic search tools by MILP, SAT/SMT, and CP.

Finally, We define "weight" which is frequently used to express the probability of a differential characteristic and a differential in this paper.

Definition 3 (Weight) *A weight w is a negative value of the binary logarithm of the differential probability DP defined as follows:*

$$w = -\log_2 DP.$$

2.2 Automatic Search Tools for Differential Characteristics and Differentials

Automatic search tools by MILP, SAT/SMT, and CP have been very popular for evaluating a differential characteristic and differential [1, 14, 17–20]. The advantage of such automatic search tools compared to conventional Matsui's algorithm is the simplicity of implementation and its efficiency. As the procedure of implementing these automatic search tools, we first convert the differential propagation over all operations in a cipher into their languages, such as linear inequalities and a Conjunctive Normal Form (CNF), and then the minimum weight can be obtained by minimizing the objective function.

Several previous works on automatic search tools try to find a better differential not only the optimal differential characteristic [1,19,21,22]. To construct a better differential, these works first search for the optimal differential characteristic and then construct a differential based on it. This strategy comes from the observation that the most contributing differential characteristics to increasing the probability of a differential are the optimal one. As mentioned in Sect. 2.1, since a differential can be seen as a bunch of multiple differential characteristics sharing the same input and output differences, we enumerate these differential characteristics by automatic search tools. Thus, the probability of such a differential constructed by multiple differential characteristics depends on the number of differential characteristics and their probabilities (weights). Because The number of differential characteristics that we can find highly depends on the efficiency of a solver and how to count such differential characteristics, sophisticating a counting strategy is important for constructing a differential.

2.3 SAT-Based Automatic Search for Differential Characteristics

Satisfiability problem A formula consisting of only AND($\wedge$),OR($\vee$),NOT($\neg$) is called Boolean formulas. In a SAT problem, we judge whether a given Boolean formula is "SAT", which means there is an assignment of Boolean variables satisfying a given Boolean formula, or not. A SAT problem is widely known as NP-complex [12], however, nowadays many SAT solvers can solve a SAT problem efficiently thanks to numerous studies on a SAT.

In a Boolean formula, we call a Boolean variable x and its negation $\neg x$ as a *literal*. These Boolean variables construct CNF by the conjunction ($\wedge$) of the disjunction ($\vee$) on themselves such as $\bigwedge_{a=0}^{i}(\bigvee_{b=0}^{j_a} c_{i,j})$, where $c_{i,j}$ is Boolean variables. We call each disjunction $\bigvee_{b=0}^{j_a} c_{i,j}$ in a Boolean formula a *clause*. It is known that any Boolean formulas can be expressed by CNF.

Overview of SAT modeling Since our method is implemented as the real SAT method rather than an SMT method, we construct SAT models to depict a differential propagation over the basic operations outlined in the work of Sun et al. [20]. A SAT model of Orthros can be divided into 4bit S-box (nonlinear

transformation), Matrix Multiplication (linear transformation) and Boolean cardinality constraints. Therefore, we only describe SAT models (clauses) of these operations.

S-box Let $(a_0, a_1, ..., a_{i-1})$ and $(b_0, b_1, ..., b_{i-1})$ be the input and output differences of an i-bit S-box, respectively. To express the weight through an S-box, we need to introduce additional binary variables $\boldsymbol{w} = (w_0, w_1, ..., w_{j-1})$ where j is the maximum weight of the differential propagation in an S-box. With the above variables, we introduce a function g as follows:

$$g(a, b, w) = \begin{cases} 1 & \text{if } Pr(a \to b) = 2^{-\sum_{q=0}^{j-1} w_q}, \\ 0 & \text{otherwise.} \end{cases}$$

Then, we extract the set A that contains all vectors satisfying $f(x, y, z) = 0$ as follows:

$$A = \{(x, y, z) \in \mathbb{F}_2^{2i+j} | f(x, y, z) = 0\}.$$

Since A is the set of invalid patterns in the S-box model, it is excluded from the set of constituent clauses by the following formula:

$$\bigvee_{c=0}^{i-1} (a_c \oplus x_c) \vee \bigvee_{d=0}^{i-1} (b_d \oplus y_d) \vee \bigvee_{e=0}^{j-1} (w_e \oplus z_e) = 1, \ (x, y, z) \in A.$$

The remaining vectors are the same set of valid patterns as $\bar{A}$. Thus, these clauses extract differential propagations with corresponding weights on i-bit S-boxes. Here, $|A|$ denotes the number of vectors in the set A, and the solution space of the clause $|A|$ for (a, b, w) in the above equation is identical to the solution space of the function h below:

$$h(a, b, w) = \bigwedge_{\eta=0}^{|A|-1} \left(\bigvee_{c=0}^{i-1} (a_c \oplus x_{c^\eta}) \vee \bigvee_{d=0}^{i-i} (b_d \oplus y_{d^\eta}) \vee \bigvee_{e=0}^{j-1} (w_e \oplus z_{e^\eta}) \right) = 1.$$

The above equation can be reformulated into a product-of-sum expression and then the minimum number of clauses can be extracted using a specific software, such as Logic Friday[3]. Thus, the clauses to represent the differential propagation considering the weight of the S-box are as follows:

$$h(a, b, w) = \bigwedge_{(x,y,z) \in \mathbb{F}_2^{2i+j}} \left(g(x, y, z) \vee \bigvee_{c=0}^{i-1} (a_c \oplus x_{c^\eta}) \vee \bigvee_{d=0}^{i-i} (b_d \oplus y_{d^\eta}) \vee \bigvee_{e=0}^{j-1} (w_e \oplus z_{e^\eta}) \right).$$

$$\mathcal{C}_{sbox_{DC}} \leftarrow \min(h(a, b, w))$$

When we evaluate the lower bound for # AS, we only need to determine whether an S-box is active or not. Therefore, we introduce a binary variable $s \in \{0, 1\}$ instead of $\boldsymbol{w}$. The rest of procedure is the same as in that for a probability model.

[3] `https://web.archive.org/web/20131022021257/http://www.sontrak.com/`

Matrix Multiplication We first give the clauses to represent an XOR operation since the matrix operation can be decomposed into multiple XOR operations.

XOR operation Let $(a_0, a_1, ..., a_{i-1})$ and b be the input and output of an i input XOR operation, respectively, i.e., $a_0 \oplus a_1 \oplus \cdots \oplus a_{i-1} = b$. Additionally, let X be a set satisfying $\{(x_0, x_1, ..., x_i) \in \mathbb{F}_2^{i+1} | (x_0 \oplus x_1 \oplus \ldots x_i) = 1\}$. The clauses to represent the differential propagation of the i-input XOR operation are as follows:

$$\mathcal{C}_{xor} \leftarrow (a_0 \oplus x_0) \vee (a_1 \oplus x_1) \vee \ldots (a_{i-1} \oplus x_{i-1}) \vee (b \oplus x_i) \text{ for all } (x_0, x_1, \ldots, x_i) \in X$$

For a matrix multiplication, we can decompose it into several XOR operations. For example, the binary matrix used in Orthros can be decomposed as follows:

$$\begin{pmatrix} 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \oplus x_2 \oplus x_3 \\ x_0 \oplus x_2 \oplus x_3 \\ x_0 \oplus x_1 \oplus x_3 \\ x_0 \oplus x_1 \oplus x_2 \end{pmatrix}.$$

Since we can view a matrix multiplication as several XOR operations from the above example, the clauses to represent a matrix operation are as follows:

$$\mathcal{C}_{matrix} \leftarrow \mathcal{C}_{xor} \; for \; all \; XORs \; decomposed \; from \; a \; matrix.$$

Boolean Cardinality Constraints To evaluate the lower bound for # AS and the total weight of a differential characteristic, we need to sum all variables to express the weight or AS over an entire model. Boolean cardinality constraints are widely used to implement such a function.

Let $X_n = (x_0, x_1, ..., x_{n-1})$ where $x_i \in \{0, 1\}$ be a sequence of literals, in which 1 and 0 denote true and false, respectively. The following equation is called a Boolean cardinality constraint on X_n:

$$\sum_{i=0}^{n-1} x_i \leq k,$$

where k is an integer value.

We employ Totalizer [3] to realize Boolean cardinality constraints. In this paper, we use $\mathcal{C}_{sum(k)}$ as the clauses to represent $\sum_{i=0}^{n-1} x_i \leq k$. Besides, we use $\mathcal{C}_{sum(\overline{k})}$ as the clauses to represent $\sum_{i=0}^{n-1} x_i \geq k$.

Joint SAT models We need to remove the obvious differential propagation such that all input differences are zero. Let $(a_0, a_1, ..., a_{i-1})$ be Boolean variables to express the input differences. We can remove such a differential propagation by the following clauses:

$$\mathcal{C}_{input} \leftarrow a_0 \vee a_1 \vee \cdots \vee a_{i-1}.$$

With the clauses to represent each operation described so far, we can construct an entire SAT model $\mathcal{M}_{SAT}$ as follows:

$$\mathcal{M}_{SAT} \leftarrow (\mathcal{C}_{sbox_{DC}}, \mathcal{C}_{matirx}, \mathcal{C}_{sec}, \mathcal{C}_{input}, \mathcal{C}_{sum(k)}).$$

If a solver returns "UNSAT", there are no assignments satisfying $\mathcal{M}_{SAT}$, i.e., the lower bound for # AS or the minimum weight outnumbers k. In this case, we increment k and repeat this procedure until a solver returns "SAT". If a solver returns "SAT", there are assignments satisfying $\mathcal{M}_{SAT}$, i.e., we find the lower bound for # AS or the minimum weight k.

2.4 Clustering Effect

As described in Sect. 2.1, we need to gather multiple differential characteristics sharing the same input and output differences to evaluate the clustering effect. Sun et al. show the easy way to realize such enumeration by a SAT [19].

Let $(a_{j,0}, a_{j,1}, ..., a_{j,i-1})$ be Boolean variables to express the differences in the input of the j-th round, where i is the position of bits. With an r-round differential characteristics $C = (c_0, c_1, ..., c_r)$, where $c_m = (c_{m,0}, c_{m,1}, ..., c_{m,i-1})$, we can fix the input and output differences to c_0 and c_r, respectively, by the following clauses:

$$\mathcal{C}_{clust} \leftarrow \begin{cases} a_{0,n} \oplus \overline{c_{0,n}} & for\ 0 \le n \le i-1. \\ a_{r,n} \oplus \overline{c_{r,n}} & for\ 0 \le n \le i-1. \end{cases}$$

To avoid solving a SAT model with the same internal differential propagation $(c_1, c_2, ..., c_{r-1})$ multiple times during the evaluation of the clustering effect, we add the following clauses to a SAT model:

$$\mathcal{C}_{\overline{clust}} \leftarrow \bigvee_{x=1}^{r-1} \bigvee_{y=0}^{i-1} (a_{x,y} \oplus c_{x,y})$$

These clauses will be repeatably added to a SAT model, wherever we find another internal differential propagation.

3 Efficient Strategy to Evaluate the Clustering Effect for a Multiple-Branch-Based Design

In differential cryptanalysis, a differential is more important than a single differential characteristic. Generally, to search for a differential with a high probability, we evaluate the clustering effect, i.e., finding multiple differential characteristics sharing the same input and output differences.

A generic strategy to evaluate the clustering effect is to count the number of differential characteristics that share the same input and output differences while simultaneously eliminating identical internal differences whenever a differential

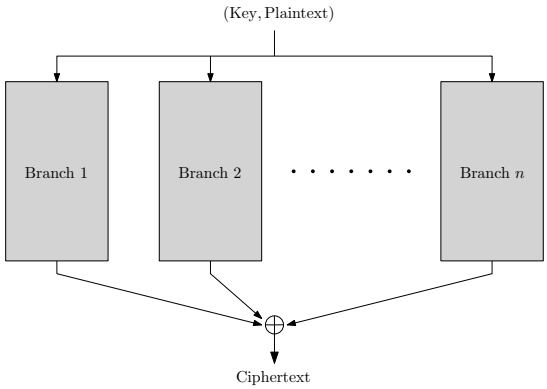

Fig. 1: Overview of n-branch-based design.

characteristic is found. As can been seen in the previous works [1, 19, 21, 22], this strategy works well on a single-branch-based design. In contrast, when considering a multiple-branch-based design, such as Orthros, the internal state size increases proportionately to the number of branches, which makes the computational cost of the evaluation expensive.

To address this issue, we propose an efficient search strategy for evaluating the clustering effect on the multiple-branch-based designs. The underlying concept is to independently evaluate the clustering effect of each branch and then construct differential characteristics for the entire construction.

In the reminder of this section, we first define our target construction and give a new metric for fairly comparing a cost of our method with that of the general one. Then, we provide an overview of our strategy and a detailed method.

3.1 Target Construction

We define the round function of a multiple-branch-based design. We extend the construction of Orthros straightforwardly and define the n-branch-based design. Figure 1 shows the overview of the n-branch-based design.

Let $E_{K_i}(\cdot)$, K, and M be any cryptographic function under key K_i, which is called "branch i" in this work, a secret key, and plaintext, respectively. The encryption algorithm of n-branch-based design $E(\cdot)$ is defined as follows:

$$E(K, M) := \bigoplus_{i=1}^{n} E_{K_i}(K, M).$$

We do not give details about a key scheduling since it does not affect to all evaluations in this work.

3.2 How to Assess the Efficiency of the Evaluation of the Clustering Effect

Generally, the efficiency of automatic search methods is measured by their practical run-time during evaluations. However, a practical run-time highly depends on the computational environment and the efficiency of solvers. In particular, for a automatic search tools based on a SAT, we have many choices of excellent SAT solvers owing to numerous dedicated works on a SAT. Thus, it seems important to introduce a new metric for automatic search tools based on a SAT.

In the evaluation of the clustering effect, we need to solve a SAT problem multiple times as explained in Sect. 2.3 This entails repeatedly invoking a SAT solver, which constitutes the majority of the cost associated with the evaluation of the clustering effect. The cost of such a single invocation, of cause, depends on the total number of clauses and Boolean variables in a solved SAT problem. Generally, the total number of clauses and variables does not vary significantly among different evaluation methods for the same target design, as the majority of clauses and variables are those that express the propagation of internal differences and weight in non-linear operations, both of which are typically common across different evaluation methods for the same target design.[4]

Hence, we introduce the number of invocations of a SAT solver to evaluate the clustering effect as a new metric as follows.

Definition 4 (the number of invocations of a SAT solver "$\#SAT$") *The number of invocations of a SAT solver $\#SAT$ is defined as the total number of "SAT" and "UNSAT" that a solver returns during the evaluation of the clustering effect.*

Note that $\#SAT$ does not contain the invocation for obtaining a differential characteristic that is used as a starting point for evaluating the clustering effect.

Suppose that we evaluate the clustering effect on a specific differential corresponding to the optimal differential characteristic with weight W_{min} by the general method. In this approach, we first enumerate the differential characteristic with weight W_{min} and repeat this procedure with incrementing weight. To increase the probability of this differential to $2^{-Wmin+\alpha}$, $\#SAT$ must be at least $2^{\alpha}+1$. Specifically, a solver returns "SAT" 2^{α} times, which indicates the existence of 2^{α} differential characteristics with weight W_{min}, and "UNSAT" once, which indicates the absence of further differential characteristics with weight W_{min}. It must be mentioned that this is the best case of the general strategy because it assumes that the differential is constructed solely by the optimal differential characteristics (it usually hardly ever happens).

We emphasis that this metric should be employed only when evaluating the efficiency of the evaluation of the clustering effect. This is because that our

[4] The number of clauses and variables in our method is smaller than those in the general method, since our method essentially evaluate the clustering effect on each branch not an entire design while the general method evaluate it on an entire design. Therefore, a practical run-time can be short in our method even if the number of solved SAT problems is the same as that of the general method.

assumption that a practical run-time depends on the number of clauses and variables in a SAT problem works only when evaluating the clustering effect, as we fix the input and output differences. In contrast, when evaluating optimal differential characteristics, the practical run-time is also influenced by other factors in many cases.

In addition to $\#SAT$, we also employ a runtime of the entire evaluation as a metric of the efficiency, similar to previous works.

3.3 Our Strategy

Let N_{cha} be the total number of differential characteristics that contribute to the probability of a differential. For one-branch-based designs, we can at most obtain a one differential characteristic by solving a one SAT problem, i.e., we can obtain differential characteristics followed by $N_{cha} = \mathcal{O}(N_{sat})$ when $\#SAT = N_{sat}$. This is also observed in the case of multiple-branch-based designs. This natural observation is the basis for most works considering the clustering effect, and it works well in their works. We call this strategy the "general strategy" in this work.

A drawback of the general strategy in the case of a multiple-branch-based design is that the computational cost becomes expensive as the number of branches increases, as the number of clauses and variables increase linearly in multiple-branch-based designs. Consequently, evaluating the clustering effect with the general strategy can get challenging when the number of branches exceeds two and the number of rounds is large.

To address this issue, we introduce a new strategy for evaluating the clustering effect on multiple-branch-based designs. The essence of our strategy is to independently evaluate the clustering effect in each branch and then construct differential characteristics for an entire design using these results. This strategy leverages the inherent trait of multiple-branch-based designs in which each differential characteristic in each branch corresponds to all differential characteristics in other branches under the same input and output differences. This can significantly increase the number of characteristics that contribute to the probability of a differential and ultimately decrease $\#SAT$ in the overall evaluation. Suppose we evaluate the clustering effect of an n-branch-based design with a pre-found optimal differential characteristic, we can obtain $N_{cha} = \mathcal{O}((N_{sat})^n)$ differential characteristics of the entire design when $\#SAT = N_{sat}$ in each branch[5]. We illustrate our strategy for enumerating the differential characteristics in Fig. 2. In Fig. 2, we search for differential characteristics in parallel based on each branch containing red, blue, and green lines, and then we can construct the differential characteristic of an entire design using the found differential characteristics in each branch. Moreover, the computational cost of solving a single SAT problem becomes small since we independently evaluate the clustering effect for every single branch.

[5] In practical, $\#SAT$ in each branch is different since it depends on various factors, such as their structure. We here assume $\#SAT$ in each branch is the same for the sake of argument.

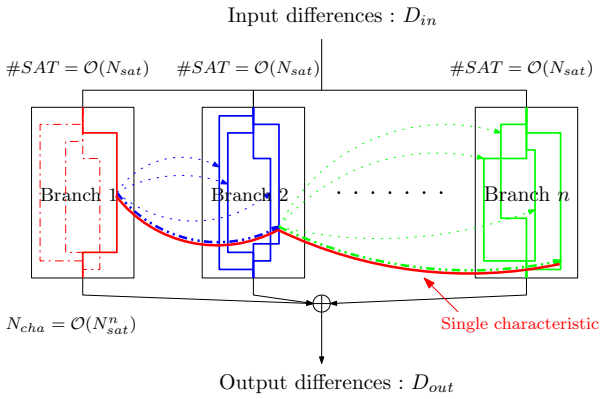

Fig. 2: Overview of our strategy to efficiently count the differential characteristics in a multiple-branch-based design.

3.4 Efficient Method to Evaluate the Clustering Effect

With the strategy outlined in Sect. 3.3, we present an efficient method for evaluating the clustering effect on a multiple-branch-based design. Our method require a specific differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ corresponding to the optimal differential characteristics which can be identified by a SAT-based automatic search tool proposed by Sun et al. [20] in advance.[6] Our method follows a five-step approach, the procedure of which is detailed step-by-step as follows:

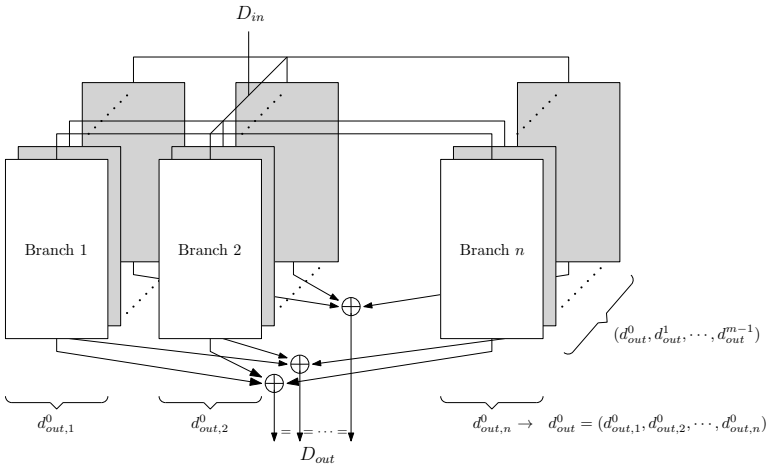

Fig. 3: Overview of Step 1.

[6] Strictly speaking, A differential characteristic do not need to be optimal, but the optimal one is the best choice for our method.

Step 1. Search for all sets of output differences $(\boldsymbol{d^0_{out}}, \boldsymbol{d^1_{out}}, \ldots, \boldsymbol{d^{m-1}_{out}})$ in each branch under a given differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ with the minimum weight W_{min}, where $\boldsymbol{d^i_{out}} = (\boldsymbol{d^i_{out,1}}, \boldsymbol{d^i_{out,2}}, \ldots, \boldsymbol{d^i_{out,n}})$, i.e., $\boldsymbol{d^i_{out,1}} \oplus \boldsymbol{d^i_{out,2}} \oplus \cdots \oplus \boldsymbol{d^i_{out,n}} = \boldsymbol{D_{out}}$. Note that m depends on some factors, such as the construction of the target and the number of rounds. After completing this step, we have multiple differentials for each branch, i.e., $\{(\boldsymbol{D_{in}}, \boldsymbol{d^i_{out,1}}), (\boldsymbol{D_{in}}, \boldsymbol{d^i_{out,2}}), (\boldsymbol{D_{in}}, \boldsymbol{d^i_{out,n}})\}$ for $0 \leq i \leq m-1$. Figure 3 illustrates the overview of Step 1.

Step 2. Count the number of differential characteristics for a differential $(\boldsymbol{D_{in}}, \boldsymbol{d^i_{out,j}})$. This procedure is virtually equivalent to evaluating the clustering effect on $(\boldsymbol{D_{in}}, \boldsymbol{d^i_{out,j}})$. Suppose that we count the number of differential characteristics for each $(\boldsymbol{D_{in}}, \boldsymbol{d^i_{out,j}})$, we will obtain a list $\boldsymbol{N^i} = (\boldsymbol{N^i_1}, \boldsymbol{N^i_2}, \ldots, \boldsymbol{N^i_n})$ where $\boldsymbol{N^i_k} = (N^i_{k,\alpha}, N^i_{k,\alpha+1}, \ldots, N^i_{k,\alpha+W_\alpha-1})$ for $\boldsymbol{d^i_{out}}$, in which each $N^i_{k,l}$ stores the number of the differential characteristics with $(\boldsymbol{D_{in}}, \boldsymbol{d^i_{out,k}})$ corresponding to weight l. Note that α and W_α can be set arbitrary. Figure 4 illustrates the overview of Step 2.

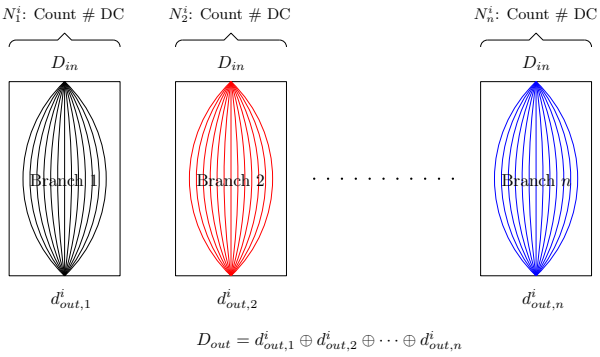

Fig. 4: Overview of Step 2. DC denotes a differential characteristic.

Step 3. Construct the differential characteristics with $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ by combining the differential characteristics found for each branch in Step 2. For $\boldsymbol{d^i_{out}}$, each differential characteristic in each branch corresponds to all differential characteristics in all branches, namely, all possible combinations of a differential characteristic of each branch bring a differential characteristic with $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$. Suppose that the sum of all elements in $\boldsymbol{N^i_k}$ is c^i_k, we can construct $(c^i_1 \times c^i_2 \times \cdots \times c^i_n)$ differential characteristics with $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ for each $\boldsymbol{d^i_{out}}$, and their probability can be calculated by the product of the probabilities of differential characteristics in each branch that compose them, that is, $\prod_{b=1}^{n} DP(\boldsymbol{C_b})$ where $\boldsymbol{C_b}$ denotes a differential characteristic of branch b. This is based on the strategy outlined in Sect. 3.3. As the output differences in each branch in $\boldsymbol{d^i_{out}}$ follow $\boldsymbol{d^i_{out,1}} \oplus \boldsymbol{d^i_{out,2}} \oplus \cdots \oplus \boldsymbol{d^i_{out,n}} = \boldsymbol{D_{out}}$, all differential characteristics constructed in this step belong to a differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$. Figure 5 illustrates the overview of Step 3.

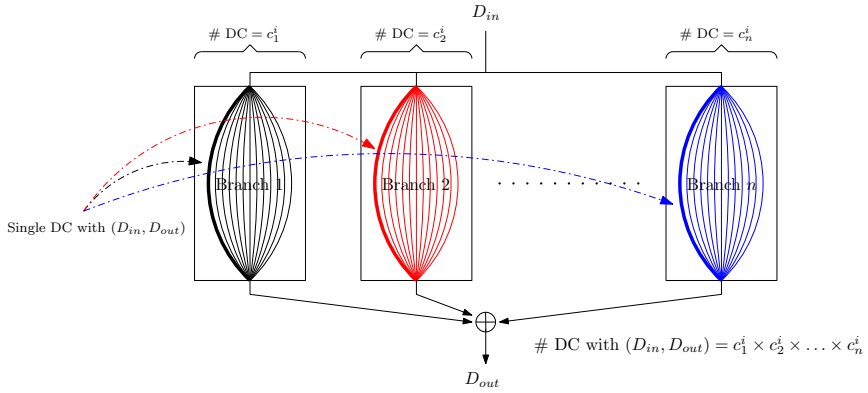

Fig. 5: Overview of Step 3. DC denotes a differential characteristic.

Step 4. Calculate the probability of a differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$. The probability can be calculated by a sum of the probability of all differential characteristics constructed in Step 3.

Step 5. Repeat steps 1 – 4 with incrementing the weight W_{min} given in step 1.

The detailed algorithm of our method is given in Algorithm 1. We describe Algorithm 1 line by line as follows:

Input: Give a differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$, the number of branches B_n, the number of rounds r, the weight W_{min} of the optimal differential characteristics corresponding to $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$, and two thresholds W_α and W_c as input. W_c specifies the range of weight in Step 5. For example, when $W_c = 3$, we conduct Step 1–4 from W_{min} to $W_{min} + 2$. W_α specifies the range of a weight related to the evaluation of the clustering effect on each branch in Step 2, i.e., the size of list $\boldsymbol{N_m^i}$ becomes $\alpha + W_\alpha - 1$. Note that α can be set arbitrarily, such as the minimum weight of the optimal differential characteristic of each branch. In our work, α is always set to a weight that a solver first returns "SAT".

Output: Return the probability of the differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$.

Lines 2–3 : Initialize P which is the probability of the differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ and $\boldsymbol{D}$ that stores $\boldsymbol{d}_i$ including $(\boldsymbol{d_{out}^0}, \boldsymbol{d_{out}^1}, \cdots, \boldsymbol{d_{out}^{m-1}},)$ for weight i.

Lines 4–12: Repeat Step 1–4 with increasing weight.

Line 5: Obtain all $(\boldsymbol{d_{out}^0}, \boldsymbol{d_{out}^1}, \cdots, \boldsymbol{d_{out}^{m-1}},)$ for weight i.

Lines 6–7: Check the overlap of $\boldsymbol{d_{out}^j}$ for all former weights. If an identical $\boldsymbol{d_{out}^j}$ has been already evaluated in another weight, it will be removed in this weight.

Lines 8–14: Count the number of differential characteristics in each weight.

Line 9: Initialize $\boldsymbol{N}$ which stores the number of differential characteristics with weight α to $\alpha + W_\alpha - 1$ for each branch. $\boldsymbol{N}$ denotes $\boldsymbol{N^i}$ in Step 2.

Algorithm 1: Evaluating the clustering effect in a design based on multiple branches.

input : $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}}), B_n, r, W_{min}, W_\alpha, W_c$
output: P

1 **begin**
2 $P \leftarrow 0$
3 $\boldsymbol{D} \leftarrow (\boldsymbol{d_0}, \boldsymbol{d_1}, \ldots, \boldsymbol{d_{W_c-1}})$
4 **for** $i = W_{min}$ **to** $W_{min} + W_c - 1$ **do**
5 $\boldsymbol{d_{i-W_{min}}} \leftarrow \mathtt{SAT}_{\mathtt{all.out}}((\boldsymbol{D_{in}}, \boldsymbol{D_{out}}), N_b, r, i)$
6 **if** $i \neq W_{min}$ **then**
7 $\mathtt{CHECK}_{\mathtt{overlap}}(\boldsymbol{D})$
8 **for** *all elements in* $\boldsymbol{d_{i-W_{min}}}$ **do**
9 $\boldsymbol{N} \leftarrow (\boldsymbol{N_1}, \boldsymbol{N_2}, \cdots, \boldsymbol{N_{B_n}})$
10 /* $\boldsymbol{N}$ denotes $\boldsymbol{N}^i$ in Step 2. */
11 **for** $j = 1$ **to** B_n **do**
12 $\boldsymbol{N_j} \leftarrow \mathtt{SAT}_{\mathtt{clust}}((\boldsymbol{D_{in}}, \boldsymbol{d^k_{out,j}}), W_\alpha)$
13 /* k corresponds to the index of element in $\boldsymbol{d_{i-W_{min}}}$ as can be seen in Step 2. */
14 $\mathtt{CALCU}_{\mathtt{Prob}}(P, \boldsymbol{N})$
15 **return** (P)

Lines 11–13: Count the number of differential characteristics with weight α to $\alpha + W_\alpha - 1$ in Branch 1 to B_n.

Line 14: Calculate the probability of a differential characteristic by combining the differential characteristics in each branch obtained in lines 11–13, and then add the sum of their probabilities to P.

Line 15: Return the probability of a differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$.

Here, we give brief explanations of functions $\mathtt{SAT}_{\mathtt{all.out}}$, $\mathtt{SAT}_{\mathtt{clust}}$, $\mathtt{CHECK}_{\mathtt{overlap}}$, and $\mathtt{CALCU}_{\mathtt{prob}}$ in Algorithm 1.

Function $\mathtt{SAT}_{\mathtt{all.out}}$(): This function searches all combinations of the output differences of each branch followed by a given difference $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$, i.e., $(\boldsymbol{d^0_{out}}, \boldsymbol{d^1_{out}}, \ldots, \boldsymbol{d^{m-1}_{out}})$ in Step 1. Such a function can be realized by a SAT-based automatic search tool proposed by Sun et al. [20] with a small modification.

Function $\mathtt{SAT}_{\mathtt{clust}}$(): This function evaluates the clustering effect of each branch with a difference $(\boldsymbol{D_{in}}, \boldsymbol{d^k_{out,j}})$. The weight range taken into account in this evaluation is arbitrary. Note that this range has a great impact on both the final probability of $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ and the computational cost. Such a function can also be realized by a SAT-based automatic search tool proposed by Sun et al. [19].

Function $\texttt{CHECK}_{\texttt{overlap}}()$**:** This function checks the overlap of $(\boldsymbol{d^0_{out}}, \boldsymbol{d^1_{out}}, \ldots, \boldsymbol{d^{m-1}_{out}})$ for all weight in Step 5. If a certain $\boldsymbol{d^j_{out}}$ has already appeared, it will be removed to avoid the overlap in the evaluation.

Function $\texttt{CALCU}_{\texttt{prob}}()$**:** This function calculates the probability of a differential characteristics with $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ by combining differential characteristics in each branch in Step 2. Suppose that a differential characteristic with $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ constructed the differential characteristics in each branch whose weights are w_b where b is the branch number, and its probability is calculated by $\prod_{i=1}^{m} 2^{-w_i}$. The total number of differential characteristics with $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ is equal to $\sum_{i=0}^{m-1} \prod_{j=1}^{n} \sum_{k \in \boldsymbol{k}} N^i_{j,k}$, where $\boldsymbol{k}$ is a set of all weight taken into account in the evaluation of the clustering effect on each branch. Then, this function sums their probabilities to the probability P.

We emphasize that how to construct these functions affects the efficiency of Algorithm 1. In particular, for $\texttt{SAT}_{\texttt{clust}}()$, we can decide α arbitrary, and the choice of α significantly affects the efficiency of Algorithm 1. Intuitively, the most efficient choice of α is the minimum weight of each branch since there are no differential characteristics under the minimum weight. For a fair comparison, we always set α to 0 in our evaluation because the general strategy does not require any information without a differential $(\boldsymbol{D_{in}}, \boldsymbol{D_{out}})$ corresponding the optimal differential characteristic.

4 Application to Orthros

4.1 Specification of Orthros

Orthros is a 128-bit low-latency PRF with a 128-bit plaintext M, ciphertext C, and kay K proposed by Banik et al. [5]. Orthros consists of two 128-bit keyed permutations Branch1 $E_1 : \mathbb{F}_2^{128} \times \mathbb{F}_2^{128} \to \mathbb{F}_2^{128}$ and Branch2 $E_2 : \mathbb{F}_2^{128} \times \mathbb{F}_2^{128} \to \mathbb{F}_2^{128}$. The encryption algorithm of Orthros is expressed as $C = E_1(K, M) \oplus E_2(K, M)$. The specifications of Branch1 and Branch2 are detailed below.

Specifications of Branch1 **and** Branch2 Branch1 and Branch2 are 128-bit keyed permutations based on an SPN structure with 12 rounds. The round function R_{f_N}, which denotes the round function in Branch N, consists of S-box (SB), bit-permutation (P_{brN}), nibble-permutation (P_{nN}), MixColumn (MC), AddRoundKey (AK) and AddConstant (AC), where $N \in \{1, 2\}$ as follows:

$$R_{f_N} = AC \circ AK \circ MC \circ P_{brN}(P_{nN}) \circ SB.$$

In the round functions of Branch1 and Branch2, bit permutation P_{brN} is applied in the first four rounds, and nibble permutation P_{nN} is applied in rounds 5 and later. The detailed explanation of each branch will be provided in Appendix A.

4.2 Existing Security Evaluation by Designers

The designers of Orthros evaluated the security against several attacks, including differential, linear, impossible and integral attacks [5], In their work, they showed the 7-round integral distinguisher as the most effective attack to Orthros.

For the differential cryptanalysis, they provided only the lower bounds for # AS and concluded that the 9-round Orthros is secure against this type attack. However, this security bound is very rough since it is provided by the sum of the lower bounds for # AS in each branch. Moreover, their lower bounds of each branch are also rough because they are independently evaluated in the first 4 rounds and the remaining rounds, i.e., they are just a sum of the lower bound in the first 4 rounds and the remaining rounds due to the high computational cost. Furthermore, the lower bound in 5–12 rounds is evaluated by a nibble-wise evaluation, and it brings a rougher bound than that in a bit-wise evaluation.

Note that designers of Orthros considers that Orthros can be secure against differential attacks when a sum of the lower bounds for # AS in Branch1 and Branch2 exceeds 64 ($2^{-2\times 64} \leq 2^{-128}$). Therefore, we follow this metric in our evaluation, namely, considering the probability of a differential characteristic in Orthros as a product of the probabilities in Branch1 and Branch2.

4.3 Updating Bounds for Differential Attacks

We apply the SAT-based automatic search method [19]. to Orthros to obtain tighter security bounds for differential attacks. Specifically, we first give the strict lower bounds of # AS based on a bit-wise difference and further obtain the optimal differential characteristics by taking differential transitions with each probability via an S-box into consideration.

AS-Based Evaluation. We provide the "exact" lower bounds for # AS up to 7 rounds of Orthros and the full rounds of each branch using a SAT-based automatic search tool proposed by Sun et al. [19]. As our evaluation is based on a bit-wise difference and takes into account bit-level differential transitions of S-box, we can find the exact lower bounds of # AS. In other words, the differential propagation found in this evaluation is always valid.

Table 3 shows our lower bounds of Orthros and each branch in comparison to the designer's results. Our result shows that 8/11/11 rounds of Orthros/Branch1/Branch2 are sufficient to guarantee security against differential attacks, while the designer's result requires at least 9/12/12 rounds, respectively. Thus, our bit-level evaluation enables updating these bounds by one round.

Our evaluation is conducted on Threadripper™3990X @2.9GHz (128 cores) with 256GB RAMs by a SAT solver P-MCOMSPS [23](40 threads used).

Finding Optimal Differential Characteristics. In the AS-based evaluation, we only consider whether an S-box is active or not. To obtain tighter bounds for differential attacks, we take the probability of differential transitions over an

Table 3: The lower bound for # AS in Orthros, Branch 1, and Branch 2.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	Ref.
B1	1	4	6	8	9	12	16	24	33	44	58	**68**	[6]
B1	1	4	6	8	11	18	28	37	48	58	**67**	80	**O**ur
B2	1	4	5	8	9	12	16	24	33	44	59	**68**	[6]
B2	1	4	5	8	10	16	26	36	49	58	**70**	80	**O**ur
Orthros	2	8	12	16	18	24	36	56	**84**	88	117	136	[6]
Orthros	2	8	12	16	22	36	58	**79**	98	129	188	196	**O**ur

S-box into account, namely, we aim at finding the optimal differential characteristics for Orthros and each branch.

Table 4 shows the optimal differential characteristic up to 7 rounds of Orthros and the full rounds of Branch1 and Branch2, where the evaluation environment is the same as that of Sect. 4.3. In comparison to the result of the AS-based evaluation in Table 3, we can reduce the number of rounds of Orthros/Branch1/Branch2 by one round to ensure secure against differential attacks, i.e. from 8/11/11 to 7/10/10, respectively.

In summary, our bit-level evaluation can improve the designer's security bounds by 2 rounds for Orthros/Branch1/Branch2, respectively. We emphasize that the optimal differential characteristics in 10 rounds of Branch1 and Branch2 can be the best distinguishing attacks for them, where known best attacks are 7-round integral distinguishers [5].

Table 4: Weight of the optimal differential characteristics in Orthros, Branch 1, and Branch 2.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	Ref.
B1	2	8	14	19	29	41	61	91	113	**142**	160	181	Our
B2	2	8	13	19	26	38	58	82	117	**136**	163	180	Our
Orthros	4	16	29	42	59	90	**136**	-	-	-	-	-	Our

4.4 How to Efficiently Capture the Clustering Effect

We leverage our SAT-based automatic search method for evaluating the clustering effect on multiple-branch-based designs to increase the differential probability. Specifically, we evaluate the clustering effect of the 7-round optimal differential characteristic of Orthros by the general and our method. For a fair comparison, we apply the identical differential characteristic to both methods and compare their efficiency in terms of how much we can enhance the probability of a given differential, $\#SAT$, and the practical run-time.

Table 5 shows the result of the general and our method.

Table 5: Comparison of our method and the general method. The parameters of our method are $W_{min} = 136, W_c = 5$ and $W_\alpha = 15$. The general method takes the clustering effect from weight 136 to 149 into consideration..

	Prob.[-log2]	$\#SAT$	Time
Our method	**121.297**	145245	**36m12.644s**
General method	127.395	2288883	114h28m28.438s
Our / general	6.098	0.0634	0.005

In the general method, we can evaluate a weight up to 151 and cannot evaluate a weight over 152 because it is computationally infeasible in our environment. As can be seen in Table 5, both methods can improve the probability to more than 2^{-128}, that is, we can improve the distinguishing attack from 7 rounds to 6 rounds due to the clustering effect. However, our method demonstrates superior efficiency compared to the general method. Specifically, our method increases the probability from 2^{-136} to $2^{-121.297}$, while the general method increases it to $2^{-127.395}$.

Furthermore, our method exhibits a significant improvement in efficiency, achieving a 93.6% and 99.5% reduction in $\#SAT$ and runtime, respectively, compared to the general method. The gap in an improvement between $\#SAT$ and a run-time comes from the difference in a size of the SAT model solved in each method. The general method solves a SAT model expressing a differential propagation in a whole Orthros while our method primarily solves a SAT model expressing a differential propagation in one branch, i.e, a size of a SAT model solved in our method is roughly half that of the general method. Since a computational cost becomes larger with increasing a size of a SAT model in general, this gap becomes larger with growing the number of branches. From this observation, our method will be getting more and more advantageous with the number of branches increasing.

4.5 Better Choice of W_α

The choice of W_α has a large impact on the probability, $\#SAT$, and a practical run-time. In this section, we present experimental results for several choices of W_α and discuss which choices of W_α are most favorable.

Table 6 shows the detailed results for $W_\alpha = 5, 10, 15, 20, 25, 30$ with $W_{min} = 136$ and $W_c = 6$. According to Table 6, the gap in the probability is not large across the range of $W_\alpha = 4$ to 30 even though each $\#SAT$ is different. In other words, the differential characteristics constructed with larger values of W_α have a limited contribution to the probability, and it is a natural observation as a higher number of differential characteristics is required to enhance the probability of a differential when the probability of these differential characteristics is low.

For a practical run-time, it seems to increase significantly with W_α becoming large. This comes from the fact that the clustering effect occurs easily in weight

Table 6: The probability, $\#SAT$, and a run-time on $W_\alpha = 5$ to 30.

$W_{\min}$ $+W_c$-1	W_α	Prob. [-log2]	$\#SAT$	Run-time	$W_{\min}$ $+W_c$-1	W_α	Prob. [-log2]	$\#SAT$	Run-time	$W_{\min}$ $+W_c$-1	W_α	Prob. [-log2]	$\#SAT$	Run-time
136	5	131.585	201	5m29s	138	5	127.532	697	8m48s	140	5	124.329	2742	10m56s
	10	130.098	1463	5m31s		10	126.231	7382	9m56s		10	123.091	29229	15m09s
	15	129.915	5607	6m27s		15	126.098	22733	11m09s		15	122.981	75619	20m20s
	20	129.911	7319	6m26s		20	126.096	26802	14m38s		20	122.980	83875	38m32s
	25	129.911	7356	6m07s		25	126.096	26913	22m04s		25	122.980	84279	1h21m02s
	30	129.911	7366	8m33s		30	126.096	26993	39m34s		30	122.980	84649	2h46m56s
137	5	131.585	201	7m01s	139	5	126.074	1174	10m10s	141	5	122.588	6325	17m11s
	10	130.098	1463	7m25s		10	124.767	12905	12m10s		10	121.396	61742	25m39s
	15	129.915	5607	7m33s		15	124.640	36727	13m39s		15	121.298	145245	36m12s
	20	129.911	7319	8m28s		20	124.638	42147	20m07s		20	121.297	157340	1h24m20s
	25	129.911	7356	8m35s		25	124.638	42318	34m08s		25	121.297	158320	3h23m00s
	30	129.911	7366	9m40s		30	124.638	42458	1h04m48s		30	121.297	159230	11h45m27s

far from W_{min} up to a point. Notably, $\#SAT$ for $W_\alpha = 30$ with $W_{min} = 141$ is almost the same as that for $W_\alpha = 25$ with $W_{min} = 141$ while the run-times of them are quite different. It is because that the distribution of the differential characteristic is biased depending on weight. Figure 6 illustrates the distribution of the differential characteristic in Branch 1 and Branch 2 for $W_\alpha = 30$ with $W_{min} = 141$. As can be seen in Fig. 6, # differential characteristics reaches the peak when the weight is around +15 to +20 from weight that a solver first returns "SAT". After reaching the peak, the differential characteristics become sparse with increasing weight, that is, there are few differential characteristics in a large W_α. Therefore, the gap in $\#SAT$ on $W_\alpha = 25$ and $W_\alpha = 30$ becomes small. However, this small gap affects a practical run-time so much because P-MCOMSPS takes a much longer run-time to solve "UNSAT" than that of "SAT" and an SAT problem that will be "UNSAT" dominates this small gap of $\#SAT$. This is the reason why the case of $W_\alpha = 30$ takes longer run-time than the case of $W_\alpha = 25$ even though their $\#SAT$ and weight are almost the same.

Therefore, $W_\alpha = 10, 15$ appear to be favorable choices for balancing both probability and practical run-time in the evaluation of Orthros. Of course, the better choice may be different depending on the designs, but we expect that $W_\alpha = 10, 15$ will be a suitable choice for most designs, as a similar distribution in Fig. 6 may appear in other designs.

4.6 Maximizing the Clustering Effect with Optimal Choice of W_α

In Sect. 4.4 and 4.5, our method consistently investigates the clustering effect in each branch starting from weight 0 for a fair comparison with the general method. However, given that we have knowledge of the minimum weight of each branch, we can further enhance the efficiency of our approach by initiating the evaluation of the clustering effect at the minimum weight of each branch rather than at 0. Here, we aim to maximize the probability of a given differential by utilizing the information of the minimum weight of each branch and the optimal selection of W_α discussed previously.

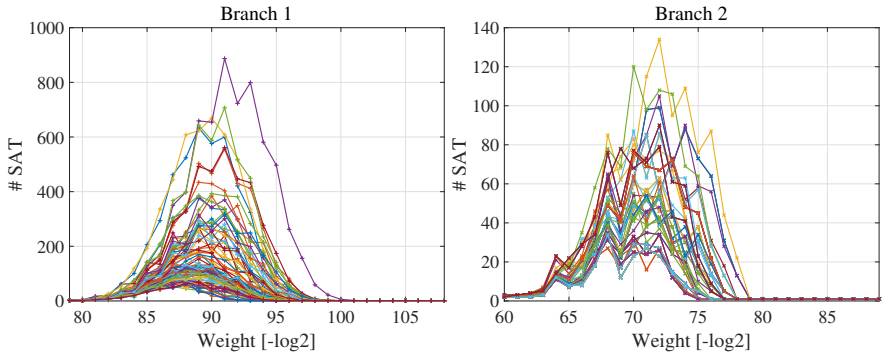

Fig. 6: $\#SAT$ on $W_\alpha = 30$ with $W_{min} = 141$. Colored lines show the distribution of the differential characteristic of each $(\boldsymbol{D_{in}}, \boldsymbol{d_1^i})$ and $(\boldsymbol{D_{in}}, \boldsymbol{d_2^i})$

Table 7 shows the result of setting the starting weight of the evaluation of the clustering effect to the minimum weight of each branch with $W_\alpha = 15$. With the optimization of our method, we can further improve the probability from $2^{-121.297}$ to $2^{116.806}$.

Table 7: The highest probability of a differential that we found.

Method	$W_{\min}$	W_c	W_α	Prob.[-log2]	$\#SAT$	Time
Our (optimized) Sect. 4.6	136	14	15	116.806	1431466	25h38m39s
Our in Sect. 4.5	136	5	15	121.298	145245	36m12s
General	-		-	127.395	2288883	114h28m28s

5 Conclusion

In this paper, we proposed a new SAT-Based automatic search method for efficiently evaluating the clustering effect. We applied our method to Orthros and showed that our method is much more efficient than the general method. As a results, we presented the distinguishing attack up to 7 rounds of Orthros with $2^{116.806}$ time/data complexity, which is the best distinguishing attack to Orthros. Besides, we updated the designer's security bound against the differential cryptanalysis from 9/12/12 to 7/10/10 rounds for Orthros/Branch 1/Branch 2, respectively.

We expect that our method would be useful to investigate the behavior of a differential in the future multiple-branch-based designs.

Acknowledgments

Takanori Isobe is supported by JST, PRESTO Grant Number JPMJPR2031. These research results were also obtained from the commissioned research (No.05801) by National Institute of Information and Communications Technology (NICT), Japan. Kosei Sakamoto is supported by Grant-in-Aid for JSPS Fellows (KAKENHI 20J23526) for Japan Society for the Promotion of Science.

A Detailed Explanation of Branch 1 and Branch 2

Orthros is a two-branch-based design in which the underlying components are SPN-based PRPs as shown in Fig. 7. The underlying two keyed permutations consist of S-box (SB), bit-permutation (P_{brN}), nibble-permutation (P_{nN}), MixColumn (MC), AddRoundKey (AK) and AddConstant (AC). We provide the detailed explanation of those function. Note that we do not give the explanation of a key scheduling because our evaluation does not consider the impact of the round keys.

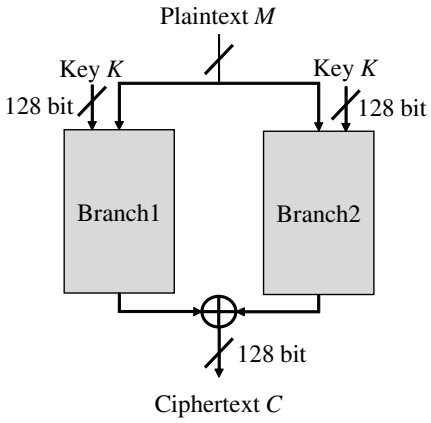

Fig. 7: Overview of Orthros

SB A 4-bit S-box will be applied to each nibbles in parallel for Branch1 and Branch2. The specification of the 4-bit S-box is given in Table 8.

P_{brN},P_{nN} For the first 4 rounds of Branch1 and Branch2, P_{br1} and P_{br2} will be applied, respectively. From the 5th round to the 11th round, the nibble permutations P_{n1} and P_{n2} will be adopted in each branch respectively. The details of the permutation P_{brN} and P_{nN}, where $N \in \{1, 2\}$, are shown in Table. 9 and Table. 10, respectively.

Table 8: 4-bit S-box of Branch1 and Branch2

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S(x)	1	0	2	4	3	8	6	d	9	a	b	e	f	c	7	5

Table 9: BP of Branch1 and Branch2

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pbr1(x)	6	46	62	126	70	52	28	14	36	125	72	83	106	95	4	35
Pbr2(x)	20	122	74	62	119	35	15	66	9	85	32	117	21	83	127	106
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Pbr1(x)	25	41	10	76	87	74	120	42	88	21	11	67	64	38	112	50
Pbr2(x)	11	98	115	59	71	90	56	26	2	44	103	121	114	107	68	16
x	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Pbr1(x)	85	109	24	65	99	0	49	37	8	66	114	47	127	100	56	40
Pbr2(x)	84	1	102	33	80	52	76	36	27	94	37	55	82	12	112	64
x	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Pbr1(x)	13	117	78	86	92	58	124	101	55	89	97	9	18	116	59	15
Pbr2(x)	105	14	91	17	108	124	6	93	29	86	123	79	72	53	19	99
x	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Pbr1(x)	20	45	75	2	77	27	1	60	115	107	26	69	119	3	84	51
Pbr2(x)	50	18	81	73	67	88	4	61	111	49	24	45	57	78	100	22
x	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
Pbr1(x)	123	110	31	82	113	53	81	102	63	118	93	12	30	94	108	32
Pbr2(x)	110	47	116	54	60	70	97	39	3	41	48	96	23	42	113	87
x	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
Pbr1(x)	5	111	29	43	91	19	79	33	73	44	98	48	22	61	68	105
Pbr2(x)	126	13	31	40	51	25	65	125	8	101	118	28	38	89	5	104
x	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
Pbr1(x)	34	71	54	104	17	57	80	103	96	121	23	39	122	90	7	16
Pbr2(x)	109	120	69	43	7	77	58	34	10	63	30	95	75	46	0	92

Table 10: NP of Branch1 and Branch2

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pn1(x)	10	27	5	1	30	23	16	13	21	31	6	14	0	25	11	18
Pn2(x)	26	13	7	11	29	0	17	21	23	5	18	25	12	10	28	2
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Pn1(x)	15	28	19	24	7	8	22	3	4	29	9	2	26	20	12	17
Pn2(x)	14	19	24	22	1	8	4	31	15	6	27	9	16	30	20	3

MC Let M_b be 4×4 binary matrix over nibbles defined as

$$M_b = \begin{pmatrix} 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix}.$$

Four nibbles (a_0, a_1, a_2, a_3) will be updated as follows:

$$(a_0, a_1, a_2, a_3)^T \leftarrow M_b \cdot (a_0, a_1, a_2, a_3)^T.$$

$\boldsymbol{R_{f_N}}$ Figure 8 and 9 show the first four and remaining rounds of each branch. Note that MC and NP are not applied in the final round.

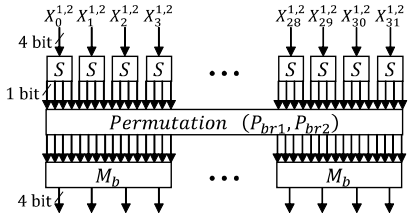

Fig. 8: The first four rounds.

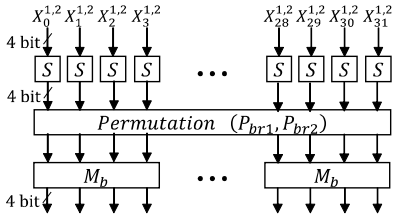

Fig. 9: The 5th to 11th round.

References

1. Ankele, R., Kölbl, S.: Mind the gap - A closer look at the security of block ciphers against differential cryptanalysis. In: SAC. Lecture Notes in Computer Science, vol. 11349, pp. 163–190. Springer (2018)
2. Avanzi, R.: The QARMA block cipher family. almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017)
3. Bailleux, O., Boufkhad, Y.: Efficient cnf encoding of boolean cardinality constraints. In: International conference on principles and practice of constraint programming. pp. 108–122. Springer (2003)
4. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 9453, pp. 411–436. Springer (2015)
5. Banik, S., Isobe, T., Liu, F., Minematsu, K., Sakamoto, K.: Orthros: A low-latency PRF. IACR Trans. Symmetric Cryptol. **2021**(1), 37–77 (2021)
6. Banik, S., Isobe, T., Liu, F., Minematsu, K., Sakamoto, K.: Orthros: A low-latency prf. IACR Transactions on Symmetric Cryptology pp. 37–77 (2021)

7. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016)
8. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. Journal of CRYPTOLOGY **4**(1), 3–72 (1991)
9. 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: CHES. Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer (2007)
10. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer (2012)
11. Boura, C., David, N., Boissier, R.H., Naya-Plasencia, M.: Better steady than speedy: Full break of SPEEDY-7-192. IACR Cryptol. ePrint Arch. p. 1351 (2022)
12. Cook, S.A.: The complexity of theorem-proving procedures. In: STOC. pp. 151–158. ACM (1971)
13. Dobraunig, C., Eichlseder, M., Kales, D., Mendel, F.: Practical key-recovery attack on MANTIS5. IACR Trans. Symmetric Cryptol. **2016**(2), 248–260 (2016)
14. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 9215, pp. 161–185. Springer (2015)
15. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 547, pp. 17–38. Springer (1991)
16. Leander, G., Moos, T., Moradi, A., Rasoolzadeh, S.: The SPEEDY family of block ciphers engineering an ultra low-latency cipher from gate level for secure processor architectures. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(4), 510–545 (2021)
17. Mouha, N., Preneel, B.: Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Paper 2013/328 (2013)
18. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Inscrypt. Lecture Notes in Computer Science, vol. 7537, pp. 57–76. Springer (2011)
19. Sun, L., Wang, W., Wang, M.: More accurate differential properties of LED64 and midori64. IACR Trans. Symmetric Cryptol. **2018**(3), 93–123 (2018)
20. Sun, L., Wang, W., Wang, M.: Accelerating the search of differential and linear characteristics with the SAT method. IACR Trans. Symmetric Cryptol. **2021**(1), 269–315 (2021)
21. 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. Cryptology ePrint Archive, Paper 2014/747 (2014)
22. Teh, J.S., Biryukov, A.: Differential cryptanalysis of WARP. IACR Cryptol. ePrint Arch. p. 1641 (2021)
23. Vallade, V., Le Frioux, L., Oanea, R., Baarir, S., Sopena, J., Kordon, F., Nejati, S., Ganesh, V.: New concurrent and distributed painless solvers: P-mcomsps, p-mcomsps-com, p-mcomsps-mpi, and p-mcomsps-com-mpi. SAT COMPETITION 2021 p. 40