Parallel SAT Framework to Find Clustering of Differential Characteristics and Its Applications

Kosei Sakamoto[1], Ryoma Ito[2], and Takanori Isobe[3]

[1] Mitsubishi Electric Corporation, Kamakura, Japan.Sakamoto.Kosei@dc.MitsubishiElectric.co.jp
[2] National Institute of Information and Communications Technology, Koganei, Japan. itorym@nict.go.jp
[3] University of Hyogo, Kobe, Japan.
takanori.isobe@ai.u-hyogo.ac.jp

Abstract. The most crucial but time-consuming task for differential cryptanalysis is to find a differential with a high probability. To tackle this task, we propose a new SAT-based automatic search framework to efficiently figure out a differential with the highest probability under a specified condition. As the previous SAT methods (e.g., the Sun et al's method proposed at ToSC 2021(1)) focused on accelerating the search of an optimal single differential characteristic, these are not optimized for evaluating a clustering effect to obtain a tighter differential probability of differentials. In contrast, our framework takes advantage of a method to solve incremental SAT problems in parallel using a multi-threading technique, and consequently, it offers the following advantages compared with the previous methods: (1) speedy identification of a differential with the highest probability under the specified conditions; (2) efficient construction of the truncated differential with the highest probability from the obtained multiple differentials; and (3) applicability to a wide class of the symmetric-key primitives. To demonstrate the effectiveness of our framework, we apply it to the block cipher PRINCE and the tweakable block cipher QARMA. We successfully figure out the tight differential bounds for all variants of PRINCE and QARMA within the practical time, thereby identifying longest distinguisher for all the variants, which improve existing ones by one to four more rounds. Besides, we uncover notable differences between PRINCE and QARMA in the behavior of differential, especially for the clustering effect. In the context of key recovery attacks, our framework allows us to derive the key-recovery-friendly truncated differentials for all variants of QARMA and give the first key recovery attack based on differential cryptanalysis. We believe that our findings shed light on new structural properties of these important primitives.

Keywords: Differential · SAT-based automatic search · Incremental SAT problem · Low-latency primitives.

1 Introduction

Background. The most crucial but time-consuming part of differential cryptanlysis [7] is to determine a pair of plaintext differences and the corresponding ciphertext differences and construct a differential distinguisher with high probability. To this end, cryptographers frequently use a *differential characteristic*, which is a sequence of the internal differences in each round. However, from the attackers' viewpoint, they are interested in not the internal differences but only a pair of the input and output differences, which is called a *differential* in literature. A differential is more useful than a differential characteristic for the attackers, as a differential has a higher probability than that of a differential characteristic.

Several studies investigated relationship between a differential characteristic and a differential, revealing that a gap between their probabilities can be significant [2, 8, 19]. Most of these studies focused on a differential constructed by only a differential characteristic with the highest probability, which is called the *optimal* differential characteristic. This seems reasonable, as the probability of the optimal differential characteristic dominates the probability of a differential in numerous designs. However, Kölbl and Roy [20] demonstrated an interesting case in Simeck32 [31] where a differential with a higher probability can be constructed by the non-optimal differential characteristic. Although this appears to be a special case, it can be valid for any design. From these aspects, finding a differential with a higher probability still remains a challenging task.

Finding such a differential is not only useful from the attackers' aspect, but also crucial from the designers' aspect. In particular, the ultra-low-latency designs must be carefully designed against differential cryptanalysis, because they are usually based on a substitution–permutation network with a small number of rounds, and the growth of the differential probability is not sufficient at the beginning of the rounds. In fact, the designers of MANTIS [6] and SPEEDY [22] invested significant efforts into guaranteeing the resistance against differential cryptanalysis in their works. Nevertheless, they were broken by differential cryptanalysis [10, 15]. Furthermore, the best attack to the first low-latency design PRINCE [9] is also (multiple) differential cryptanalysis on 10 (out of 12) rounds proposed by Canteaut et al. [12]. Hence, it is evidently important to investigate a differential in detail, especially for low-latency designs.

Limitations of SAT-based Automatic Search Tools. The existing SAT-based automatic search tools, proposed by Sun et al. [27, 28], focused on accelerating the search of an optimal differential characteristic by incorporating the Matsui's bounding conditions [25]. These tools are valid for evaluating a single differential characteristic, but the Matsui's bounding conditions are not suitable for the purpose of evaluating the clustering effect of multiple differential characteristics; thus, the existing tools are not suitable for efficiently finding a differential with a higher probability. Certainly, it can be applied to evaluate the clustering effect of multiple differential characteristics by removing the Matsui's bounding conditions and adding some new conditions. However, such a straightforward

adjustment can be inefficient because Sun et al. assumed only an environment with a single thread execution even though their SAT solver accepts an execution on multiple threads. Considering that the evaluation for the clustering effect of multiple differential characteristics having different input and output differences requires a much more computational cost than that for finding the optimal differential characteristic, the tool for finding a differential with the highest possible probability should be optimized for an execution on multiple threads. Moreover, it is also of great importance to investigate the impact of relation on the efficiency in which between the number of threads to be assigned to solve a single SAT problem and the degree of the parallelization for the evaluation of the clustering effect, as we have to evaluate the clustering effect for each found differential characteristic having different input and output differences with a high probability. Therefore, without these considerations, it is hard to efficiently investigate the clustering effect of numerous differential characteristics having different input and output differences in detail. This investigation leads to understanding the behavior of the probability about differentials more deeply; thus, optimizing these SAT-based tools to evaluate for the clustering effect of differential characteristics is crucial.

Our Contributions. In this study, we propose a new generic SAT-based automatic search framework that aims to figure out a differential with a higher probability under the specified condition, in contrast to existing approaches. The main concept of the framework involves investigating the clustering effect of all differential characteristics having different input and output differences with a specified range of weight and identifying the good differential. Our framework fully leverages a method to solve *incremental SAT problems*, which can efficiently solve a SAT problem with small modifications multiple times, in parallel using a multi-threading technique. As an incremental SAT problem can be efficiently solved by the *bounded variable elimination method* [16], it is known that we can efficiently evaluate the clustering effect by converting the evaluation of the clustering effect into an incremental SAT problem. In our method, we also take advantage of an incremental SAT problem to efficiently find all differential characteristics having different input and output differences that are seeds to construct differentials, as well as the evaluation of the clustering effect. By carefully investigating the most suitable parameters, such as the number of thread to be assigned to solve a single incremental SAT problem and the degree of the parallelization for the evaluation of the clustering effect, to solve multiple incremental SAT problems efficiently, our framework enables us to thoroughly evaluate the clustering effect of such all differential characteristics not only with the highest probability but also with any probability. Hence, we evaluate the probability of differentials more comprehensively than any other previous methods.

Identifying Good Differentials on PRINCE and QARMA. To demonstrate the effectiveness of our framework, we apply it to PRINCE [9] and QARMA [3], which

Table 1: Comparison of our results with existing ones regarding distinguishers.

Cipher	Total # Rounds	Attacked # Rounds	Setting†	Type‡	Time/Data	Reference
PRINCE PRINCEv2	12	4	SK	ID	–	[14]
		6	SK	D	2^{62}	[2]
		6	SK	I	2^{62}	[11]
		6	SK	D	$2^{56.42}$	[12]
		7	SK	D	$2^{55.771}$	Sect. 4.1
QARMA64	16	6	SK	ID	–	[30]
		7	SK	D	$2^{58.921}$	Sect. 4.2
		4.5	RT	ID	–	[24]
		7	RT	ID	–	[33]
		8	RT	SS	2^{57}	[23]
		9	RT	ZC/I	2^{44}	[1]
		10	RT	D	$2^{60.831}$	Sect. 4.2
QARMA128	24	6	SK	ID	–	[30]
		10	SK	D	$2^{121.549}$	Sect. 4.2
		6.5	RT	ID	–	[24]
		8	RT	TDIB	$2^{124.1}$	[23]
		12	RT	D	$2^{120.024}$	Sect. 4.2

† SK: Single-Key, RT: Related-Tweak
‡ D: Differential, I: Integral, ID: Impossible Differential, SS: Statistical Saturation, ZC: Zero-Correlation, TDIB: Tweak Difference Invariant Bias

are the reflection ciphers for low-latency applications. As a result, we significantly improve previous differential bounds for all variants of these ciphers as shown in Table 1, and our differential distinguishers are longest ones among existing one. It is important to note that while the previous attacks may have been adjusted for key recovery, identifying longest distinguisher is very important to deeply comprehend the structural properties of these primitives as pseudo random permutations. These results demonstrate that the proposed framework is effective for evaluating the tight differential bounds.

Difference in Behavior of Clustering Effect between PRINCE and QARMA. We look into the difference between PRINCE and QARMA in the behavior of a differential. Our experiments observe that the gaps in the probability between a differential characteristic and a differential can be large in QARMA under the SK setting compared to that in PRINCE. Specifically, QARMA under the single-key (SK) setting has a large impact on the clustering effect, and the case reported by Kölbl and Roy [20] can occur in QARMA under the SK setting. A detailed

investigation of such gaps reveals that they are influenced by different design strategies for the linear layers (i.e., matrices). After conducting the additional experiments using four types of matrices with different properties, we find that the target cipher has a good resistance to a clustering effect when each output bit of the round function depends on as many input bits of the round function as possible. We conclude that a cipher using a matrix with the same property as that used in QARMA has a large impact on a clustering effect, and a clustering effect in non-optimal weights can strongly affect the probability of a differential.

In the context of the key recovery attack, we only show the summary of our key recovery attacks to QARMA in Appendix D due to the page limitation (the detailed attack procedure will be given in the full version of this paper). To date, no study has been reported on a key recovery attack based on straightforward differential cryptanalysis against QARMA. One possible issue is that it was difficult to find the key-recovery-friendly differentials for QARMA, as QARMA has a large impact on a clustering effect. Our proposed SAT-based automatic search framework can solve this issue, and consequently, a key recovery attack based on the straightforward differential cryptanalysis can be performed with the time and data complexities comparable to the best attacks. Our framework can be applied to any symmetric-key primitive. Also, it is very important to analyze the tight differential bound in the field of the symmetric-key cryptanalysis. Therefore, we believe that our work is a significant contribution in terms of the tight security analysis for a wide class of symmetric-key primitives.

2 Preliminaries

2.1 Definitions of Differential Characteristic and Differential

We frequently use terms *differential characteristic* and *differential* throughout this paper. To avoid mixing these terms, we specify their definitions and how to calculate their probabilities. Further, we provide the definition of *weight* that is also frequently used in this paper. Notably, we explain a differential characteristic and differential over an r-round iterated block cipher $E(\cdot) = f_r(\cdot) \circ \cdots \circ f_1(\cdot)$.

Definition 1 (Differential characteristic) *A differential characteristic is a sequence of differences over E defined 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 the differences in the output of each round, i.e., $\boldsymbol{c_0}$ and $\boldsymbol{c_r}$ denote the differences in a plaintext and ciphertext, respectively.

The probability of a differential characteristic is estimated by the product of the corresponding differential probabilities for each round on the Markov cipher assumption [21] as follows:

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

Definition 2 (Differential) *A differential is a pair of the input and output differences* $(\boldsymbol{c_0}, \boldsymbol{c_r})$.

The probability of a differential is estimated by a sum of probabilities for all differential characteristics sharing the same input and output differences $(\boldsymbol{c_0}, \boldsymbol{c_r})$ as follows:

$$\Pr(\boldsymbol{c_0} \xrightarrow{E} \boldsymbol{c_r}) = \sum_{\boldsymbol{c_1}, \boldsymbol{c_2}, \cdots \boldsymbol{c_{r-1}}} \Pr(\boldsymbol{c_0} \xrightarrow{f_1} \boldsymbol{c_1} \xrightarrow{f_2} \cdots \xrightarrow{f_r} \boldsymbol{c_r}).$$

We finally provide the definition of *weight* which corresponds to the probability of a differential characteristic.

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

$$w = -\log_2 P_r$$

2.2 SAT-Based Automatic Search for Differential Characteristics

SAT. When a formula consists of only AND ($\wedge$), OR ($\vee$), and NOT ($\bar{\cdot}$) operations based on Boolean variables, we refer to it as a *Boolean formula*. In a SAT problem, a SAT solver checks whether there is an assignment of Boolean variables that can validate a Boolean formula or not. If such an assignment exists, a SAT solver returns *satisfiable* or "SAT". Generally, a SAT problem is an NP-complete [13]. However, owing to numerous efforts for SAT problems, nowadays, there are numerous excellent SAT solvers that can solve a SAT problem very efficiently, such as `CaDiCaL`, `Kissat`, and `CryptoMiniSat5`.

A Boolean formula can be converted into a *Conjunctive Normal Form* (CNF), which is expressed by the conjunction ($\wedge$) of the disjunction ($\vee$) on (possibly negated) Boolean variables, such as $\bigwedge_{a=0}^{i}(\bigvee_{b=0}^{j_a} c_{i,j})$, where $c_{i,j}$ is a Boolean variable. We call each disjunction $\bigvee_{b=0}^{j_a} c_{i,j}$ in a Boolean formula a *clause*.

SAT-Based Automatic Tools. SAT-based automatic tools are known as a valid approach to find optimal differential/linear characteristics and more powerful than MILP-based ones as shown in [28]. To implement its approach with the SAT method, the differential/linear propagation over all operations in a primitive must be converted into a CNF, and then we check if there exists a differential/linear characteristic along with a specified weight as a SAT problem. We can know the optimal differential/linear characteristics by solving some SAT problems with changing the number of specified weights.

SAT Models for Basic Operations. Our framework is based on a pure-SAT model proposed by Sun et al. [27, 28]. Due to the page limitation, we do not give the detailed modeling method (for more information, please refer to Sun et al.'s work). Herein, we specify some basic notations that are used in this study to construct a whole SAT model as follows:

$\mathcal{M}_{SAT}$: A whole SAT model that we solve.
$\mathcal{M}_{cla.operations}$: Clauses to express the propagation of differences in a certain operation. These clauses also contain variables to express a weight corresponding to the propagation of differences in a probabilistic operation.
$\mathcal{M}_{var}$: Variables to construct clauses.

In this study, we use $\mathcal{M}_{cla.xor}$, $\mathcal{M}_{cla.matrix}$, and $\mathcal{M}_{cla.sbox}$ as clauses to express the propagation of differences in PRINCE and QARMA. In addition, we also use $\mathcal{M}_{cla.input}$ and $\mathcal{M}_{cla.sec(B)}$ to evaluate a minimum weight. These clauses play a role as follows:

$\mathcal{M}_{cla.input}$: Clauses to avoid a trivial differential propagation, such as all input differences being zero at the same time.
$\mathcal{M}_{cla.sec(B)}$: Clauses to count the total weight of a primitive. More specifically, the constraint of $\sum_{i=0}^{j} p_i \leq B$ can be added, where p_i is a Boolean variable to express a weight and j is the total number of p_i. There are several methods to realize such a constraint in a Boolean formula [4,26,29]. Among these, we employ *Sequential Encoding Method* [26] that was used in numerous works.

Finding Differential Characteristics with Minimum Weight. With the clauses and variables introduced in this section, we construct a whole SAT model as follows:

$$\mathcal{M}_{SAT} \leftarrow (\mathcal{M}_{cla.matrix},\ \mathcal{M}_{cla.sbox},\ \mathcal{M}_{cla.sec},\ \mathcal{M}_{cla.input}).$$

Now, we are ready to find a differential characteristic with the minimum weight by feeding $\mathcal{M}_{SAT}$ and $\mathcal{M}_{var}$ to a SAT solver. If a SAT solver returns "UNSAT", there is no differential characteristic with a weight of $\leq B$. In that case, we increment B and repeat it until a SAT solver returns "SAT". This means that we obtain a differential characteristic with the minimum weight of B.

Modeling for a Clustering Effect. To take a clustering effect into account, we must solve a SAT problem multiple times with the same input and output differences, while the identical internal differential propagation is deleted from the solution space of the initial SAT problem. To realize this procedure, we introduce the following clauses:

$\mathcal{M}_{cla.clust}$: Clauses to fix the input and output differences to find multiple differential characteristics with the same input and output differences.
$\mathcal{M}_{cla.\overline{clust}}$: Clauses to remove the internal differential propagation from a SAT model. These will be repeatably added to a SAT model whenever another internal differential propagation is found.

When evaluating a clustering effect, we attempt to find a differential characteristic with the weight of B, not the weight of $\leq B$ so as to calculate the exact probability of a differential. due to the same reason mentioned in [27]. $\sum_{j=0}^{r\cdot i-1} p_j = B$ can be obtained by applying both $\sum_{j=0}^{r\cdot i-1} p_j \leq B$ and $\sum_{j=0}^{r\cdot i-1} p_j \geq B$. The first

constraint is already given above, and the second one can be easily obtained from $\sum_{j=0}^{r\cdot i-1} p_j \leq B$ with a small change. More information is provided in the previous study [27]. Hereafter, $\mathcal{M}_{cla.\overline{sec}(B)}$ denotes the clauses to express $\sum_{j=0}^{r\cdot i-1} p_j \geq B$. The detailed comprehensive algorithm for finding differential characteristics and evaluating the clustering effect will be given in the following section.

3 A New SAT Framework to Find the Best Differential

In this section, we propose a new generic SAT-based automatic search framework to find a differential with a higher probability under a specified condition (we refer to it as a *good* differential in this paper). Specifically, our framework can efficiently investigate the clustering effect of all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with a specified range of probability and identify a good differential. Our framework leverages a method to solve *incremental SAT problems* in parallel using a multi-threading technique, leading to an efficient search for all differentials under the specified condition. Specifically, the unique features of our framework are listed as follows:

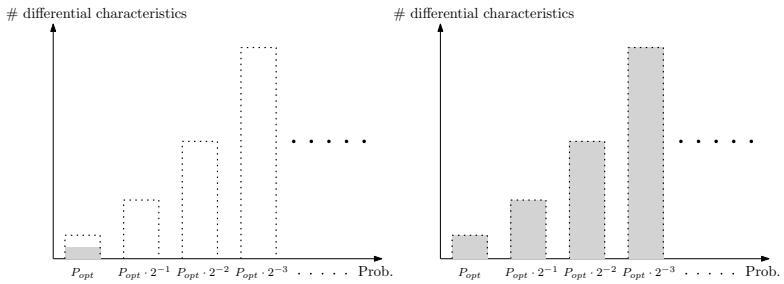
Speedy identification of a good differential. Most of existing studies on the solver-aided search methods have focused on searching for the optimal differential characteristics as efficiently as possible. In contrast, our framework aims to identify a good differential among numerous differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ by evaluating the clustering effect of them within the practical time. This can be realized by taking a method to solve incremental SAT problems in parallel using a multi-threading technique into consideration. Thereby, our framework enables us to find good differentials under the specified range of the weight that the corresponding differential characteristic has.

Efficient construction of a good truncated differential. Our framework also enables us to find a good truncated differential. This can be realized by combining all the obtained differentials under the specified truncated differential. The truncated differential attack is more powerful than ordinary differential attack; thus, our framework leads to a better differential attack on many symmetric-key primitives.

Applicability to a wide class of the symmetric-key primitives. Our framework leverages the existing SAT-based automatic search method proposed by Sun et al. [28] and maintains its availability of applications; thus, our framework can be applied to a wide class of the symmetric-key primitives. Therefore, compared with existing solver-aided tools, our framework can be the best tool to construct the (truncated) differential distinguisher for a wide class of the symmetric-key primitives.

3.1 Our Approach

Conventionally, when we attempt to obtain a good differential, we adopt a strategy of searching it based on the optimal differential characteristic. This strategy



(a) Conventional approach (b) Our approach

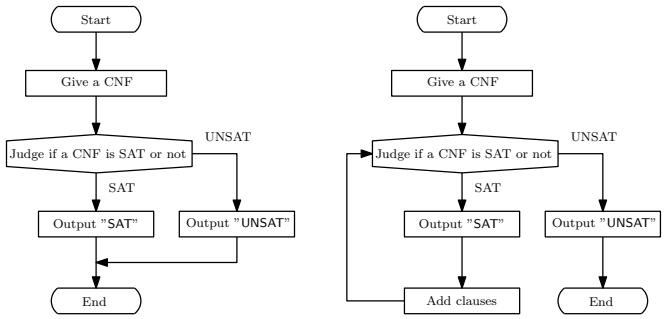
Fig. 1: Approaches to identifying a good differential. "# differential characteristics" denotes the total number of differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with the corresponding probability in horizontal axis. P_{opt} denotes the probability of an optimal differential characteristic. The gray area depicts evaluated differentials.

seems reasonable in many cases; therefore, most of existing studies followed this strategy and improved the differential attacks based on the differentials obtained by this strategy. However, this strategy might overlook the better one because the non-optimal differential characteristic sometimes constructs the better differentials than that by the optimal differential characteristic, as the case on Simeck32 reported by Kölbl and Roy [20].

To investigate differentials in more detail, we need to evaluate a clustering effect of numerous differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$. Since this requires a huge computational cost, it is a time-consuming task even with the state-of-the-art approach, such as a pure SAT-based automatic search method proposed by Sun et al. [28]. To tackle this task, we focus on a method to efficiently solve an incremental SAT problem and consider a new strategy to speedily obtain all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with a specified range of weight to evaluate the clustering effect of them. The essential idea of our search strategy is very simple; we first enumerate all single differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with a relatively high probability and then investigate the clustering effect of every obtained differential characteristic. Fig. 1 illustrates the overview of our approach in comparison with the conventional one.

3.2 Incremental SAT Problem

An *incremental SAT problem* is a kind of SAT problem, which solves a general SAT problem multiple times with a small modification, which the *bounded variable elimination method* [16] can efficiently realize. Several SAT solvers support the function to efficiently solve the incremental SAT problem, such as *CryptoMiniSAT* which is the most popular SAT solver in the field of symmetric-key cryptography. Fig. 2 illustrates flowcharts of solving the general and an incremental SAT problem.



(a) General SAT problem (b) Incremental SAT problem

Fig. 2: Flowcharts of solving the general and an incremental SAT problem.

Some Insights about Solving an Incremental SAT Problem. According to the Erlacher et al's work [17], assigning multiple threads to solve a single general SAT problem has a positive impact on reducing the runtime, but does not obtain the same degree of the gain as the degree of the parallelization. From this fact, our work starts at investigating whether the same phenomenon happens in the case of an incremental SAT problem. As a result, we find that it happens in the case of an incremental SAT problem as well. Moreover, we also find that assigning multiple threads to solve a single incremental SAT problem does not improve the efficiency of the evaluation at all (see Sect. 3.4). This means that solving multiple incremental SAT problems in parallel on each single thread is more efficient than solving a single incremental SAT problem on multiple threads. We leverage this insight into our framework.

Good Solver for an Incremental SAT Problem. There are numerous excellent SAT solvers tending to solve a general SAT problem, while not so many of them support solving an incremental SAT problem. Since our framework requires to efficiently solve not a general SAT problem but an incremental SAT problem, we must employ a SAT solver suitable for solving an incremental SAT problem. To the best of our knowledge, `CryptoMiniSat5`[4] is the most efficient SAT solver to solve an incremental SAT problem[5]. Hence, we use `CryptoMiniSat5` throughout all of our evaluations.

3.3 Finding a Good Differential

We present a new method to find a good differential under a specified condition. Our method requires several basic algorithms to find differential characteristics, such as ones presented in [28]. We leave the detailed explanation of them in Appendix B due to a page limitation.

[4] `https://www.msoos.org/cryptominisat5/`

[5] `CryptoMiniSat5` is the winner of the incremental library track at SAT competition 2020.

Algorithm 1: Finding the best differential.

input : W_{min}, r, T_w, T_c
output: $\boldsymbol{D}, \boldsymbol{N}$

begin
  $\boldsymbol{D} \leftarrow (\boldsymbol{D_0}, \boldsymbol{D_1}, \ldots, \boldsymbol{D_{T_w-1}})$
  $\boldsymbol{N} \leftarrow (\boldsymbol{N_0}, \boldsymbol{N_1}, \ldots, \boldsymbol{N_{T_w-1}})$
  for $i = W_{min}$ **to** $W_{min} + T_w - 1$ **do**
    $\boldsymbol{D_{i-W_{min}}} \leftarrow \mathtt{SAT_{diff.all}}(i, r, \mathbf{1}, \mathbf{1})$
    $\boldsymbol{N_{i-W_{min}}} \leftarrow \emptyset$
    $j \leftarrow 0$
    for *all pairs in* $\boldsymbol{D_{i-W_{min}}}$ **do**
      add $\mathtt{SAT_{diff.clust}}(i, i + T_c - 1, r, \boldsymbol{D^{(j)}_{i-W_{in}}})$ **to** $\boldsymbol{N_{i-W_{in}}}$
      $j \leftarrow j + 1$
      /* j denotes the index of $\boldsymbol{D_{i-W_{in}}}$, i.e., $MAX(j) = |\boldsymbol{D_{i-W_{in}}}|$ */
  return $(\boldsymbol{D}, \boldsymbol{N})$

The idea of our method is to investigate a clustering effect about all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with not only the minimum weight, but also a specified range of weight, and then identify a good differential. Before giving a detailed algorithm of our method, we explain the procedure of this method step by step as follows:

Step 1: Identify the weight W_{min} of the r-round optimal differential characteristic by $\mathtt{SAT_{diff.min}}()$.
Step 2: Obtain all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with the weight from W_{min} to $W_{min} + \alpha$ by $\mathtt{SAT_{diff.all}}()$.
Step 3: Evaluate the clustering effect of all differential characteristics obtained in Step 2, and then find a good differential.

As can be seen in the above steps, this method can investigate the probability of differentials in more detail than any other existing tools. We give the detailed algorithm of this method in Algorithm 1.

As inputs to Algorithm 1, we provide the minimum weight W_{min}, the number of target rounds r, and two thresholds T_w and T_c. We can obtain W_{min} by $\mathtt{SAT_{diff.min}}()$ and decide T_w as the range of weights taken into account in the whole evaluation. For example, suppose that we obtain $W_{min} = 60$ by $\mathtt{SAT_{diff.min}}()$ and set $T_w = 3$, Algorithm 1 searches a good differential in all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with the weight of $60, 61$, and 62. We can also decide T_c as the range of weight taken into account in a clustering effect for each differential characteristic. After executing Algorithm 1, we obtain lists of $\boldsymbol{D}$ and $\boldsymbol{N}$ which store all differentials $(\boldsymbol{c_0}, \boldsymbol{c_r})$ and the number of the differential characteristics for each weight in each differential, respectively. Then, we can calculate the probability for each differential with $\boldsymbol{D}$ and $\boldsymbol{N}$.

The computational cost of Algorithm 1 highly depends on T_w and T_c, because these two thresholds highly influence the number of times to solve an incremental

SAT problem in the whole procedure of Algorithm 1. Therefore, T_w and T_c must be set depending on the computational environment. It should be noted that the clustering effect for each differential will be evaluated in parallel because of some observations discussed in Sect. 3.4.

3.4 Optimizing the Efficiency by a Multi-Threading Technique

To optimize the efficiency of our algorithms, we investigate the feature of an incremental SAT problem, e.g., the most efficient way to solve multiple incremental SAT problems. More specifically, we examine the difference in the runtimes depending on the relationship between the number of threads assigned to solve each incremental SAT problem and the degree of parallelization to solve multiple incremental SAT problems. To this end, we define a rule for assigning the number of threads and the degree of parallelization to satisfy the following equation:

$$P_{deg} = \frac{T_m}{T_s}, \tag{1}$$

where P_{deg}, T_m, and T_s denote the degree of parallelization to solve multiple incremental SAT problems, the total number of threads assigned for our evaluations, and the number of threads assigned to solve a single incremental SAT problem, respectively. Based on the above assignment rule, to clarify the relationship between the number of threads and the degree of parallelization, we conduct experimental evaluations for the 5-round PRINCE, the 9-round PRINCE, and the 6-round QARMA64 under the SK setting based on Algorithm 1. Due to the limitations of our experimental environments, the total number of threads T_m assigned for our evaluations of PRINCE and QARMA is 8 and 16, respectively.

Fig. 3 shows the runtime of each evaluation. In this figure, the vertical axis represents the runtime of each evaluation and the horizontal axis represents the degree of parallelization P_{deg}. Besides, to further investigate the effect of the number of threads assigned to a single incremental SAT problem, we conduct additional experiments for PRINCE and QARMA on the environment of $(P_{deg} = 4, T_m = 4, T_s = 1)$ and $(P_{deg} = 8, T_m = 8, T_s = 1)$, respectively[6]. These results show the runtime of 1h8m8s, 1h26m31s, and 35m15s for the 5-round PRINCE, the 9-round PRINCE, and the 6-round QARMA64, respectively. From all our evaluations, we can see the following interesting observations:

- Increasing the degree of parallelization is greatly useful to improve the runtime of our algorithms. This can be intuitively seen from Fig. 3.
- Assigning many threads to solve a single incremental SAT problem does not improve the runtime of our algorithms even though we can improve in the case of a general SAT problem by the same approach to some extent. Unfortunately, it worsens the efficiency of our algorithms in the case of the 6-round QARMA64. This is because our experimental results for the 6-round QARMA64 show the runtime of 35m15s on the environment of

[6] Both evaluations are conducted by the same computers as the evaluation in Fig. 3.

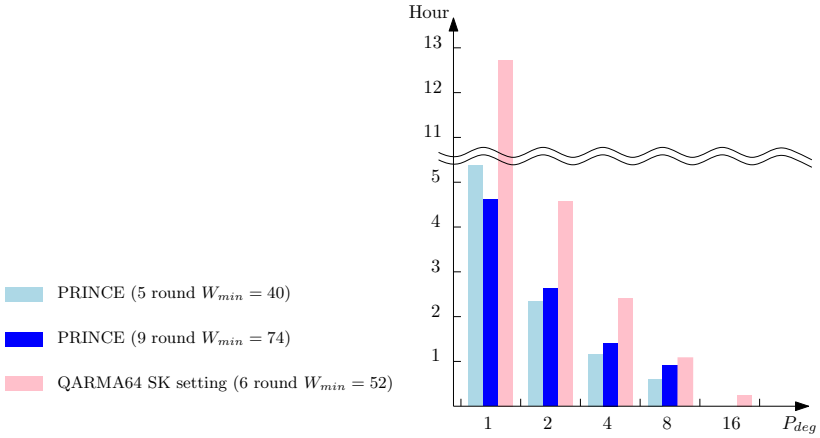


Fig. 3: The runtime for each environment according to Eq. (1). The evaluations of PRINCE and QARMA are conducted on computers with 8 and 16 threads, respectively. W_{min} denotes the weight of the differential characteristics evaluated for the clustering effect.

$(P_{deg} = 8, T_m = 8, T_s = 1)$ but the runtime of 1h6m4s on the environment of $(P_{deg} = 8, T_m = 16, T_s = 2)$.

These features probably could come from how to solve a given SAT problem on multiple threads in a SAT solver. `CryptoMiniSat5` employs the *portfolio* approach[7] that provides the interface to efficiently share the learned clauses for a set of CDCL solver instances [18]. The essence of this approach is to assign the same SAT problems to each thread, each of which attempts to solve them individually by sharing learned clauses with other threads. Hence, it seems natural that this approach is more effective for a difficult SAT problem than for an easy SAT problem, because the overhead for sharing learned clauses cannot be negligible in a small SAT problem. In the evaluation of the clustering effect, we solve an incremental SAT problem that aims to very efficiently solve a general SAT problem with a modification multiple times. Therefore, we expect that evaluating the clustering effect of a single differential by multiple threads does not have a positive effect on the efficiency of our algorithms. We would like to mention that this phenomenon could be also observed in not only SAT solvers with portfolio approach but also ones with other approaches because assigning multiples threads to a single small SAT problem is excess even in other approaches.

From the above observations, we conclude that assigning a single incremental SAT problem to each thread is more advantageous than assigning many threads to a single incremental SAT problem. It should be mentioned that this observation may be consistent between incremental SAT problems whose the number

[7] The portfolio approach is a popular approach to solve a given SAT problem on multiple threads. Note that the portfolio approach is not for an incremental SAT problem but for a general SAT problem.

of clauses and Boolean variables vary because we can see the same feature in the results of the 5-round PRINCE and the 9-round PRINCE. Thus, we decide to assign an independent incremental SAT problem to each thread when evaluating the clustering effect.

Therefore, based on the above observations, we incorporate a method to solve incremental SAT problems in parallel using a multi-threading technique into Algorithm 1.

3.5 A More Efficient Algorithm to Find a Good Differential

Algorithm 1 can find a good differential under the specified condition, while a computational cost becomes vast along with increasing T_w and T_c. The downside of Algorithm 1 is that it never returns any result when all differentials cannot be found out, and this situation happens often along with a weight far from W_{min}.

To address this problem, we propose Algorithm 2, which can evaluate a clustering effect whenever a differential characteristic having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ is found. In Algorithm 2, it is not always possible to identify a good differential under a specified condition, as we discard some differentials $(\boldsymbol{c_0}, \boldsymbol{c_r})$ in the middle of the procedure. However, we place emphasis on evaluating a clustering effect as efficiently as possible. To reduce the entire computational cost, we screen the differential $(\boldsymbol{c_0}, \boldsymbol{c_r})$ depending on its differential probability by a certain threshold whenever evaluating a clustering effect. If it does not satisfy a certain threshold, the evaluation of a clustering effect for this differential $(\boldsymbol{c_0}, \boldsymbol{c_r})$ halts, and this differential is discarded. In Algorithm 2, we assume to execute it in parallel on an environment with multiple threads based on the fact in Sect. 3.4. We explain the overview of the procedure step by step as follows:

Step 1: Find the same number of differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with the weight W_{min} as the degree of parallelization.

Step 2: Evaluate the clustering effect for each obtained differential characteristic in parallel. During this evaluation, we store or update the information of a differential $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with the highest probability (specifically, the differential and its probability), and this information is used to specify the threshold. If the probability of a differential $(\boldsymbol{c_0}, \boldsymbol{c_r})$ in the middle of evaluating the clustering effect does not surpass a certain threshold, this evaluation halts, and such a differential is discarded. Otherwise, the evaluation proceeds, and the highest probability is updated if the probability of the resulting differential exceeds the previous highest one.

Step 3: Repeat Step 1–2 until all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with the weight W_{min} are found. If it is infeasible to find all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$, we stop the evaluation and obtain the highest probability of a differential in this evaluation so far.

Step 4: Increase W_{min} and repeat Step 1–3 until W_{min} reaches a specified weight.

As inputs to Algorithm 2, we provide the same parameters in Algorithm 1 and the additional two thresholds T_s and T_t which are the bounding condition

Algorithm 2: Finding the (almost) good differential for a multi thread programming technique

input : $W_{min}, r, T_w, T_c, T_s, T_t, N_{thr}$
output: $(\boldsymbol{c_{opt.in}}, \boldsymbol{c_{opt.out}}), P_{opt}$

begin
  $P_{opt} \leftarrow 0$, $\boldsymbol{P_{thr}} \leftarrow (P_{thr}^0, P_{thr}^1, \ldots, P_{thr}^{N_{thr}-1})$
  $\boldsymbol{D} \leftarrow (\boldsymbol{D_0}, \boldsymbol{D_1}, \ldots, \boldsymbol{D_{N_{thr}-1}})$
  for $i = W_{min}$ **to** $W_{min} + T_w - 1$ **do**
    $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \mathtt{SET}_{\mathtt{model}}(i, r)$
    add *auxiliary Boolean variables of* $\mathcal{M}_{cla.\overline{sec}(i)}$ **to** $\mathcal{M}_{var}$
    add $\mathcal{M}_{cla.\overline{sec}(i)}$ **to** $\mathcal{M}_{SAT}$
    $count \leftarrow 0$
    `/* incremental SAT problem */`
    while $\mathtt{SAT}_{\mathtt{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var}) =$ ("SAT", $\boldsymbol{C_r}$) **do**
      $\boldsymbol{D_{count \bmod N_{thr}}} \leftarrow (\boldsymbol{c_0}, \boldsymbol{c_r})$
      $count \leftarrow count + 1$
      if $count \bmod N_{thr} = 0$ **then**
        for *each thread* **do**
          $P_{thr}^{thread} \leftarrow \mathtt{Thread}(i, r, T_c, T_s, T_t, P_{opt}, \boldsymbol{D_{thread}})$
        if $MAX(\boldsymbol{P_{thr}}) > P_{opt}$ **then**
          $(\boldsymbol{D_{opt}}, P_{opt}) \leftarrow MAX(\boldsymbol{D}, \boldsymbol{P_{thr}})$
      add $\bigvee_{k=0}^{n-1}(v_{0,k} \oplus c_{0,k}) \vee (v_{r,k} \oplus c_{r,k})$ **to** $\mathcal{M}_{SAT}$
    if $count \bmod N_{thr} \neq 0$ **then**
      for *each thread* **do**
        $P_{thr}^{thread} \leftarrow \mathtt{Thread}(i, r, T_c, T_s, T_t, P_{opt}, \boldsymbol{D_{thread}})$
      if $MAX(\boldsymbol{P_{thr}}) > P_{opt}$ **then**
        $(\boldsymbol{D_{opt}}, P_{opt}) \leftarrow MAX(\boldsymbol{D}, \boldsymbol{P_{thr}})$
  return $(\boldsymbol{D_{opt}}, P_{opt})$

Function $\mathtt{Thread}(W, r, T_c, T_s, T_t, P_{opt}, \boldsymbol{D})$ `// A multi-threading technique`
begin
  $\boldsymbol{N} \leftarrow (N_0, N_1, \ldots, N_{T_c-1})$
  $\boldsymbol{N} \leftarrow \mathtt{SAT}_{\mathtt{diff.clust}}(W, W + T_t - 1, r, \boldsymbol{D})$
  $P_{tmp} \leftarrow \sum_{i=W}^{W+T_t-1}(N_{i-W} \cdot 2^{-i})$
  if $T_s \cdot P_{tmp} > P_{opt}$ **then**
    $\boldsymbol{N} \leftarrow \mathtt{SAT}_{\mathtt{diff.clust}}(W + T_t, W + T_c - 1, r, \boldsymbol{D})$
    $P_{tmp} \leftarrow P_{tmp} + \sum_{i=W+T_t}^{W+T_c-1}(N_{i-W} \cdot 2^{-i})$
  return P_{tmp}

used to narrow down the search space. We specify T_t and T_s as a range of the evaluated weight in the clustering effect before screening and a specific threshold of screening, respectively. Besides, we specify the degree of parallelization in

Table 2: Differential probabilities of (almost) good differentials of PRINCE. W_{min} denotes the same parameter as in Algorithms 1 and 2. #differentials denotes the number of different differentials with a particular weight. The minimum weight of a differential characteristic for each round is written in bold. The highest differential probability for each round is written in red. The probabilities in a white and gray cell are obtained by Algorithms 1 and 2, respectively. For all results, we set $T_w = 1$ and $T_c = 10$.

PRINCE

Rounds	4 (1+2+1)					5 (1+2+2/2+2+1)				
W_{min}	**32**	33	34	35	36	**39**	40	41	42	43
Prob.	$2^{-30.868}$	$2^{-31.861}$	$2^{-32.587}$	$2^{-33.333}$	$2^{-32.979}$	$2^{-38.810}$	$2^{-39.385}$	$2^{-40.017}$	$2^{-40.607}$	$2^{-40.837}$
# differentials	477452	3792944	4929816	5537848	5547896	576	12512	113840	598592	2231756
Time	6h06m57s	48h48m43s	47h34m17s	47h35m06s	48h01m15s	1m21s	26m09s	4h08m26s	23h14m24s	48h03m32s

Rounds	6 (2+2+2)					7 (2+2+3/3+2+2)				
W_{min}	**44**	45	46	47	48	**56**	57	58	59	60
Prob.	$2^{-43.907}$	$2^{-44.907}$	$2^{-45.195}$	$2^{-46.111}$	$2^{-46.374}$	$2^{-55.771}$	$2^{-55.887}$	$2^{-56.810}$	$2^{-57.37}$	$2^{-57.990}$
# differentials	64	512	1984	6592	25968	5632	100976	835456	205272	212280
Time	51s	4m21s	17m57s	1h07m16s	4h46m53s	5h07m16s	90h40m16s	48h00m00s	73h03m01s	71h43m12s

Rounds	8 (3+2+3)					9 (3+2+4/4+2+3)				
W_{min}	**66**	67	68	69	70	**74**	75	76	77	78
Prob.	$2^{-64.389}$	$2^{-65.384}$	$2^{-66.303}$	$2^{-66.970}$	$2^{-67.075}$	$2^{-73.888}$	$2^{-74.881}$	$2^{-74.970}$	$2^{-75.970}$	$2^{-76.166}$
# differentials	256	3584	46736	18352	24056	64	544	3400	26592	13968
Time	1h55m50s	24h34m09s	290h41m48s	47h32m37s	48h4m28s	34m49s	5h11m49s	32h10m51s	235h42m42s	48h04m53s

Step 2 by N_{thr}. After executing Algorithm 2, we obtain a good differential $\boldsymbol{D_{opt}}$ with its probability P_{opt}.

In Appendix C, we show experimental results for some parameters of T_s and T_t and discuss how parameters are acceptable to be set.

4 Applications to PRINCE and QARMA

We apply our framework to PRINCE and QARMA in some rounds. To make our results clear, we show the results on each W_{min} with $T_w = 1$, i.e., we consistently set $T_w = 1$ for each W_{min}. Furthermore, we set $T_c = 10$ unless noted otherwise.

4.1 Good Differentials for PRINCE

Table 2 shows the results of PRINCE, which are evaluated on Apple M1 MAX with 64 GB of main memory. In the case where the number of all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ is not so many, and the number of rounds is small, we can apply Algorithm 1, i.e., we can find a good differential with $T_c = 10$. In other cases, the cost of the evaluation of a clustering effect becomes so high that we apply Algorithm 2. For the results by Algorithm 1, the evaluation of a clustering effect is parallelized on multiple threads to make the most of our computational environment, as described in Sect. 3.3 and 3.4. For the results by Algorithm 2, we pick up the best one among results on several combination of T_t and T_s.

Table 2 shows that the distinguishing attack can be applied up to seven rounds of PRINCE/PRINCEv2 that improves the previous best attack by one round [2,12]. It must be mentioned that the previous best distinguishing attack by differential cryptanalysis is adjusted for the key recovery that restricts the space of the input and output differences.

4.2 Good Differentials for QARMA

Table 3 shows the results of QARMA64 and QARMA128, both of which are evaluated on Linux machine with Intel Xeon Gold 6258R CPU (2.70 GHz) and 256 GB of main memory. As with the case of PRINCE, we apply Algorithm 1 when the number of all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ is not so many, and the number of rounds is small. Otherwise, we apply Algorithm 2. Particularly, the computational cost becomes excessive in the evaluation of QARMA128, because the state length is 128 bits. Hence, we apply only Algorithm 2 in most cases of the evaluation of QARMA128. For the results by Algorithm 1, the evaluation of a clustering effect is parallelized on multiple threads to make the most of our computational environment, as well as the evaluation of PRINCE. For the results by Algorithm 2, we pick up the best one among results on several combinations of T_t and T_s.

As shown in Table 3, the distinguishing attack in the SK setting can be applied up to 7 and 10 rounds of QARMA64 and QARMA128, both of which improve the previous best attack [30] by 1 and 4 rounds, respectively. Further, the distinguishing attack in the RT setting can be applied up to 10 and 12 rounds of QARMA64 and QARMA128, both of which improve the previous best attacks [1,23] by 1 and 4 rounds, respectively. As with the case of PRINCE, we note that the previous best distinguishing attack may be adjusted for the key recovery. Besides, it must be mentioned that the same case reported by Kölbl and Roy [20] often happens in both QARMA64 and QARMA128, i.e., there are some better differentials corresponding to a differential characteristic with not the highest probability than that by the optimal differential characteristic.

4.3 Discussion: Comparison with PRINCE and QARMA

We observe that the gaps in the probability between a differential characteristic and a differential can be large in QARMA64 and QARMA128 under the SK setting compared to that in PRINCE. When looking at each construction in detail, for the non-linear layer, the 4-bit S-boxes used in PRINCE and QARMA have the same property in terms of security, such as a full diffusion property and guaranteeing the maximum differential probability and the absolute linear bias of 2^{-2}. In contrast, their linear layers are designed with a different strategy. The linear layer of PRINCE is designed to ensure 16 active S-boxes in consecutive four rounds, while that of QARMA is designed based on an almost MDS matrix suitable for hardware implementation. We summarize the difference in their matrices from the macro and micro perspectives as follows. Hereafter, we mainly

Table 3: Differential probabilities of (almost) good differentials of QARMA. W_{min} denotes the same parameter as in Algorithms 1 and 2. #differentials denotes the number of different differentials with a particular weight. The minimum weight of a differential characteristic for each round is written in bold. The highest differential probability for each round is written in red. The probabilities in a white and gray cell are obtained by Algorithms 1 and 2, respectively. For all results, we set $T_w = 1$ and $T_c = 10$.

QARMA64 under the SK setting

Rounds	6 (2+2+2)			7 (2+2+3/3+2+2)			8 (3+2+3)		
W_{min}	**52**	53	54	**64**	65	66	**72**	73	74
Prob.	$2^{-45.741}$	$2^{-46.019}$	$2^{-46.112}$	$2^{-60.278}$	$2^{-60.111}$	$2^{-58.921}$	$2^{-64.845}$	$2^{-64.503}$	$2^{-64.693}$
# differentials	1024	18048	315360	512	16896	313280	400	21904	333776
Time	35m15s	19h47m31s	109h51m44s	48m19s	39h48m41s	186h21m10s	15h47m58s	53h01m41s	508h11m56s

QARMA64 under the RT setting

Rounds	6 (2+2+2)			7 (2+2+3/3+2+2)			8 (3+2+3)		
W_{min}	**14**	15	16	**28**	29	30	**36**	37	38
Prob.	$2^{-14.000}$	$2^{-14.913}$	$2^{-15.193}$	$2^{-27.541}$	$2^{-28.000}$	$2^{-28.286}$	$2^{-36.000}$	$2^{-36.679}$	$2^{-36.679}$
# differentials	17	202	2571	84	3030	48840	20	840	18509
Time	36s	1m44s	13m33s	5m35s	1h15m24s	15h28m20s	11m16s	30m22s	10h18m25s

Rounds	9 (3+2+4/4+2+3)			10 (4+2+4)			11 (4+2+5/5+2+4)		
W_{min}	**52**	53	54	**62**	63	64	**77**	78	79
Prob.	$2^{-51.415}$	$2^{-51.415}$	$2^{-52.246}$	$2^{-60.831}$	$2^{-60.831}$	$2^{-60.831}$	$2^{-77.000}$	$2^{-77.415}$	$2^{-77.509}$
# differentials	8	688	11290	273	4822	49585	64	7616	18424
Time	6h32m25s	10h27m32s	49h31m02s	96h12m59s	114h45m17s	303h33m25s	596h07m26s†	1317h17m08s†	1317h16m57s†

QARMA128 under the SK setting

Rounds	6 (2+2+2)			7 (2+2+3/3+2+2)			8 (2+2+4/4+2+2)		
W_{min}	**60**	61	62	**76**	77	78	**87**	88	89
Prob.	$2^{-54.494}$	$2^{-54.521}$	$2^{-54.581}$	$2^{-71.930}$	$2^{-72.321}$	$2^{-72.614}$	$2^{-84.850}$	$2^{-85.093}$	$2^{-85.539}$
# differentials	1312	98984	391352	516	32880	31960	16	708	14300
Time	15h27m17s	499h19m12s	1316h25m40s†	40h57m50s	530h05m58s	430h44m47s	57h59m37s	92h7m23s	693h25m04s

Rounds	9 (3+2+4/4+2+3)			10 (3+2+5/5+2+3)		
W_{min}	**106**	107	108	**125**	126	127
Prob.	$2^{-104.285}$	$2^{-103.616}$	$2^{-103.255}$	$2^{-121.549}$	$2^{-121.667}$	$2^{-122.304}$
# differentials	240	561	1172	12	54	31
Time	249h25m14s†	1004h00m44s†	1004h00m32s†	794h25m35s†	794h25m23s†	794h25m13s†

QARMA128 under the RT setting

Rounds	7 (2+2+3/3+2+2)			8 (3+2+3)			9 (3+2+4/4+2+3)		
W_{min}	**28**	29	30	**42**	43	44	**64**	65	66
Prob.	$2^{-28.000}$	$2^{-27.415}$	$2^{-28.000}$	$2^{-42.000}$	$2^{-42.415}$	$2^{-42.187}$	$2^{-63.679}$	$2^{-64.415}$	$2^{-64.679}$
# differentials	32	2144	64368	64	5248	203200	1815	6870	26105
Time	38m43s	4h51m52s	48h32m23s	21h17m20s	52h32m19s	470h54m17s	1154h39m26s†	1154h39m16s†	1154h39m05s†

Rounds	10 (4+2+4)			11 (4+2+5/5+2+4)			12 (5+2+5)		
W_{min}	**80**	81	82	**100**	101	102	**125**	126	127
Prob.	$2^{-78.005}$	$2^{-79.005}$	$2^{-78.408}$	$2^{-96.466}$	$2^{-97.929}$	$2^{-96.521}$	$2^{-120.024}$	$2^{-123.499}$	$2^{-124.084}$
# differentials	2	72	51	9	6	2	3	3	2
Time	978h51m03s†	1316h34m33s†	1316h33m53s†	794h24m09s†	794h23m59s†	1036h39m39s†	794h16m56s†	1036h44m17s†	1036h44m02s†

† These experiments were stopped before all differentials were obtained because the program took too long to run.

take a comparison between PRINCE and QARMA64 as an example for a better understanding.

Table 4: Probability of differential characteristic and differential.

PRINCE (6 (2+2+2) rounds) $T_w = 1$, $T_c = 10$				
Matrix	Original	M_{e1}	M_{e2}	M_{e3}
W_{min}	**44**	**40**	**44**	**42**
Prob.	$2^{-43.907}$	$2^{-38.526}$	$2^{-38.616}$	$2^{-37.458}$
Gap (Prob./$2^{-W_{min}}$)	$2^{0.093}$	$2^{1.474}$	$2^{5.384}$	$2^{4.542}$
# differentials	64	256	8	272

Macro perspective. When looking at the matrices of PRINCE and QARMA64 as a single 64×64 matrix, the matrix of PRINCE consists of two 16×16 matrices $\widehat{M}^{(0)}$ and $\widehat{M}^{(1)}$ while that of QARMA64 consists of only one 16×16 matrix M. Hence, the (forward and backward) round function of PRINCE can be seen as constructed on two super S-boxes, while that of QARMA64 can be seen as constructed on the one super S-box.

Micro perspective. When focusing on output nibbles, each output nibble in the matrix of PRINCE comes from four input nibbles, while that of QARMA64 comes from three input nibbles. Thus, each output bit of the round function of PRINCE depends on 16 input bits of the round function, while that of QARMA64 depends on 12 input bits of the round function.

To further investigate an impact of a matrix on a gap in the probability, we conduct three experiments with a change of the matrix in PRINCE focusing on the above perspectives. Hence, we change the matrix in PRINCE to:

$$M_{e1} = diag(\widehat{M}^{(0)}, \widehat{M}^{(0)}, \widehat{M}^{(0)}, \widehat{M}^{(0)});$$
$$M_{e2} = diag(circ(0, \rho^1, \rho^2, \rho^1), circ(0, 1, \rho^2, 1), circ(0, 1, \rho^2, 1), circ(0, \rho^1, \rho^2, \rho^1));$$
$$M_{e3} = diag(circ(0, \rho^1, \rho^2, \rho^1), circ(0, \rho^1, \rho^2, \rho^1), circ(0, \rho^1, \rho^2, \rho^1), circ(0, \rho^1, \rho^2, \rho^1)).$$

Notably, $circ(0, 1, \rho^2, 1)$ in M_{e2} has the same diffusion property as $circ(0, \rho^1, \rho^2, \rho^1)$ given in [3]. With M_{e1}, the round function can be viewed as constructed on the one super S-box, but each output bit of the round function still depends on 16 input bits of the round function. With M_{e2}, the round function can be viewed as constructed on two super S-boxes like the original PRINCE, but each output bit of the round function depends on 12 input bits of the round function. With M_{e3}, the matrix in PRINCE changes to the same matrix as QARMA64 into PRINCE, that is, the round function can be viewed as constructed on the one super S-box and each output bit of the round function depends on 12 input bits of the round function.

Tables 4 and 5 show the gap in the probability of the differential characteristic and differential on the six rounds of each variant of PRINCE and their distribution of the differential characteristics, respectively. From a macro perspective, the number of super S-boxes based on a primitive does not seem to have an impact on the gap as far as comparing the cases of the original matrix with M_{e1} and M_{e2} with M_{e3}. Meanwhile, the number of the input bits influencing each output bit seems to have a large impact on the gap as far as comparing

Table 5: Distribution of differential characteristics.

PRINCE (6 (2+2+2) rounds) $T_w = 1$, $T_c = 10$											
Matrix \ Weight		W_{min}	$W_{min}+1$	$W_{min}+2$	$W_{min}+3$	$W_{min}+4$	$W_{min}+5$	$W_{min}+6$	$W_{min}+7$	$W_{min}+8$	$W_{min}+9$
# DC†	Original	1	0	0	0	1	0	0	0	1	0
	M_{e1}	2	0	0	0	11	0	0	0	23	0
	M_{e2}	1	2	7	16	55	116	452	848	2152	3498
	M_{e3}	1	0	5	2	56	38	358	210	1719	1102

† DC: Differential Characteristic

the cases of the original matrix with M_{e2} and M_{e1} with M_{e3}. These observations can fit into MIDORI64 [5] and SKINNY64 [6], both of which have the matrix with each output nibble depending on less than four input nibbles. Ankele and Kölbl showed that the probability of the optimal differential characteristic in MIDORI64 and SKINNY is dramatically increased by considering a clustering effect [2]. When each output bit depends on 16 input bits, the number of the differential characteristics for each weight is curbed very few. Therefore, we predict that a cipher can have good resistance to a clustering effect when each output bit of the round function depends on more input bits of the round function.

In the RT setting, this gap of QARMA becomes small compared to that in the SK setting, i.e., the permutation-based tweak update function like that used in QARMA brings resistance to a clustering effect. That is mainly because the transition of the differential propagation is uniquely fixed in the tweak update function, and it contributes to making clustering difficult in the whole cipher. Therefore, we expect that a tweakable block cipher with a linear tweak (tweakey) update function can have a good resistance to the clustering effect.

Finally, the case reported by Kölbl and Roy [20] can occur in any cipher, as a clustering effect in non-optimal weights can strongly affect the probability of a differential, especially for a cipher like QARMA.

5 Conclusion

We provide a new generic SAT-based automatic search framework to find a good differential under the specified conditions. Our framework introduces a method to solve incremental SAT problems in parallel using a multi-threading technique, and consequently, it allows us to evaluate differentials more comprehensively than any other previous methods.

Our framework can be applied to a wide class of symmetric-key primitives. In this study, to demonstrate the effectiveness of our framework, we apply it to PRINCE and QARMA from aspects of distinguishing and key recovery attacks. Our results are summarized as follows:

– We specify the conditions of finding a good differential to build a distinguisher and conduct experiments using our framework. As a result, we improve previous differential bounds for all variants of the target ciphers.

– We investigate the gap in the probability between a differential characteristic and a differential for PRINCE and QARMA and find that different design strategies for the linear layers has a significant impact on this gap.

For future direction, it would be interesting to expand the incremental SAT problem to more efficiently find the optimal differential/linear characteristics and other kinds of distinguishers. Further, it would be useful for future designs to more comprehensively investigate the impact of the design construction on the gap in the probability between a differential characteristic and a differential.

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.

References

1. Ankele, R., Dobraunig, C., Guo, J., Lambooij, E., Leander, G., Todo, Y.: Zero-correlation attacks on tweakable block ciphers with linear tweakey expansion. IACR Trans. Symmetric Cryptol. **2019**(1), 192–235 (2019)
2. 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)
3. 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)
4. Bailleux, O., Boufkhad, Y.: Efficient CNF encoding of boolean cardinality constraints. In: CP. Lecture Notes in Computer Science, vol. 2833, pp. 108–122. Springer (2003)
5. 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)
6. 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)
7. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: CRYPTO. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990)
8. Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: FSE. Lecture Notes in Computer Science, vol. 8540, pp. 546–570. Springer (2014)
9. 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)

10. 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)
11. Bozilov, D., Eichlseder, M., Knezevic, M., Lambin, B., Leander, G., Moos, T., Nikov, V., Rasoolzadeh, S., Todo, Y., Wiemer, F.: Princev2 - more security for (almost) no overhead. In: SAC. Lecture Notes in Computer Science, vol. 12804, pp. 483–511. Springer (2020)
12. Canteaut, A., Fuhr, T., Gilbert, H., Naya-Plasencia, M., Reinhard, J.: Multiple differential cryptanalysis of round-reduced PRINCE. In: FSE. Lecture Notes in Computer Science, vol. 8540, pp. 591–610. Springer (2014)
13. Cook, S.A.: The complexity of theorem-proving procedures. In: STOC. pp. 151–158. ACM (1971)
14. Ding, Y., Zhao, J., Li, L., Yu, H.: Impossible differential analysis on round-reduced PRINCE. J. Inf. Sci. Eng. **33**(4), 1041–1053 (2017)
15. Dobraunig, C., Eichlseder, M., Kales, D., Mendel, F.: Practical key-recovery attack on MANTIS5. IACR Trans. Symmetric Cryptol. **2016**(2), 248–260 (2016)
16. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: SAT. Lecture Notes in Computer Science, vol. 3569, pp. 61–75. Springer (2005)
17. Erlacher, J., Mendel, F., Eichlseder, M.: Bounds for the security of ascon against differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2022**(1), 64–87 (2022)
18. Hamadi, Y., Jabbour, S., Sais, L.: Manysat: a parallel SAT solver. J. Satisf. Boolean Model. Comput. **6**(4), 245–262 (2009)
19. 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)
20. Kölbl, S., Roy, A.: A brief comparison of simon and simeck. In: LightSec. Lecture Notes in Computer Science, vol. 10098, pp. 69–88. Springer (2016)
21. 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)
22. 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)
23. Li, M., Hu, K., Wang, M.: Related-tweak statistical saturation cryptanalysis and its application on QARMA. IACR Trans. Symmetric Cryptol. **2019**(1), 236–263 (2019)
24. Liu, Y., Zang, T., Gu, D., Zhao, F., Li, W., Liu, Z.: Improved cryptanalysis of reduced-version QARMA-64/128. IEEE Access **8**, 8361–8370 (2020)
25. Matsui, M.: On correlation between the order of s-boxes and the strength of DES. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 950, pp. 366–375. Springer (1994)
26. Sinz, C.: Towards an optimal CNF encoding of boolean cardinality constraints. In: CP. Lecture Notes in Computer Science, vol. 3709, pp. 827–831. Springer (2005)
27. Sun, L., Wang, W., Wang, M.: More accurate differential properties of LED64 and midori64. IACR Trans. Symmetric Cryptol. **2018**(3), 93–123 (2018)
28. 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)

29. Warners, J.P.: A linear-time transformation of linear inequalities into conjunctive normal form. Inf. Process. Lett. **68**(2), 63–69 (1998)
30. Yang, D., Qi, W., Chen, H.: Impossible differential attack on QARMA family of block ciphers. IACR Cryptol. ePrint Arch. p. 334 (2018)
31. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The Simeck Family of Lightweight Block Ciphers. In: CHES. Lecture Notes in Computer Science, vol. 9293, pp. 307–329. Springer (2015)
32. Zong, R., Dong, X.: Meet-in-the-middle attack on QARMA block cipher. IACR Cryptol. ePrint Arch. p. 1160 (2016)
33. Zong, R., Dong, X.: Milp-aided related-tweak/key impossible differential attack and its applications to qarma, joltik-bc. IEEE Access **7**, 153683–153693 (2019)

A Specifications of PRINCE, PRINCEv2, and QARMA

PRINCE and PRINCEv2. PRINCE [9] and PRINCEv2[8] [11] are two family of a block cipher with a 64-bit block and 128-bit key. Both ciphers have the same structure except for the round constant, key scheduling, and how to insert the round keys. Both PRINCE and PRINCEv2 are constructed in the same three functions the *forward round function* FR, *middle round function* MR, and *backward round function* BR as follows:

$$\begin{aligned}\texttt{FR}(\cdot) &= SR \circ MC \circ SB(\cdot),\\ \texttt{MR}(\cdot) &= SB^{-1} \circ MC \circ SB(\cdot),\\ \texttt{BR}(\cdot) &= SB^{-1} \circ MC^{-1} \circ SR^{-1}(\cdot).\end{aligned}$$

SB is the parallel use of the 4-bit S-box defined by Table 6. SR is the shift row operation that applies the same permutation used in AES as shown in Table 7. MC is the MixColumns operation composed from the following four basic 4×4 binary matrices:

$$M_1 = diag(0,1,1,1),\ M_2 = diag(1,0,1,1),\ M_3 = diag(1,1,0,1),\ M_4 = diag(1,1,1,0),$$

where $diag()$ denotes a diagonal matrix. These four matrices build two 16×16 binary matrices as follows:

$$\widehat{M}^{(0)} = \begin{pmatrix} M_1 & M_2 & M_3 & M_4 \\ M_2 & M_3 & M_4 & M_1 \\ M_3 & M_4 & M_1 & M_2 \\ M_4 & M_1 & M_2 & M_3 \end{pmatrix},\ \widehat{M}^{(1)} = \begin{pmatrix} M_2 & M_3 & M_4 & M_1 \\ M_3 & M_4 & M_1 & M_2 \\ M_4 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_4 \end{pmatrix}.$$

Finally, $\widehat{M}^{(0)}$ and $\widehat{M}^{(1)}$ build the 64×64 matrix M' applied in MC as follows:

$$M' = diag(\widehat{M}^{(0)}, \widehat{M}^{(1)}, \widehat{M}^{(1)}, \widehat{M}^{(0)}).$$

[8] The differential characteristics and differentials are identical since PRINCE and PRINCEv2 have the same round function except for the round constants and how to insert the round keys and we do not care about the influence of the round keys on the internal differences in this work.

Table 6: 4-bit S-box of SB.

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

Table 7: Permutation of SR.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11

The total number of rounds can be expressed as $(r_1{+}2{+}r_2)$ when the number of rounds for `FR`, `MR`, and `BR` are r_1, 2, and r_2, respectively. Notably, `MR` has two rounds while `FR` and `BR` have one, i.e., the number of rounds is counted by the number of SB and SB^{-1}. More information is provided in [9, 11].

QARMA. QARMA [3] is a family of lightweight tweakable block ciphers. It has two variants, QARMA64 and QARMA128, that support the block size n of 64 bits and 128 bits, respectively. The corresponding tweak size is equal to n bits, while the master key K has $2n$ bits. All n-bit values can be viewed as an array of 16 m-bit cells or 4×4 matrices, i.e.,

$$\mathsf{IS} = s_0||s_1||\cdots||s_{14}||s_{15} = \begin{pmatrix} s_0 & s_1 & s_2 & s_3 \\ s_4 & s_5 & s_6 & s_7 \\ s_8 & s_9 & s_{10} & s_{11} \\ s_{12} & s_{13} & s_{14} & s_{15} \end{pmatrix}$$

so that 4×4 matrices operate column-wise on these values by left multiplication.

Next, we briefly introduce the round and tweak update functions.

Round Function. The round function is composed of the following operations:

`AddRoundTweakey.` The i-th round tweakey, which consists of the round key and round constant, is XORed to IS.

`ShuffleCells.` $(\tau(\mathsf{IS}))_i = s_{\tau(i)}$ for $0 \leq i \leq 15$, where τ is the cell permutation of Midori [5], i.e., $\tau = [0, 11, 6, 13, 10, 1, 12, 7, 5, 14, 3, 8, 15, 4, 9, 2]$.

`MixColumns.` Each column of IS is multiplied by the matrix M, i.e., $\mathsf{IS} = M \cdot \mathsf{IS}$. The matrix M is defined as follows:

$$M = circ(0, \rho^a, \rho^b, \rho^c) = \begin{pmatrix} 0 & \rho^a & \rho^b & \rho^c \\ \rho^c & 0 & \rho^a & \rho^b \\ \rho^b & \rho^c & 0 & \rho^a \\ \rho^a & \rho^b & \rho^c & 0 \end{pmatrix},$$

where ρ^i is a simple left circular rotation of the element by i bits. The matrix of QARMA64 is selected as $M = \overline{M} = Q = circ(0, \rho^1, \rho^2, \rho^1)$, while the matrix of QARMA128 is selected as $M = \overline{M} = Q = circ(0, \rho^1, \rho^4, \rho^5)$.

`SubCells.` $s_i \leftarrow \sigma(s_i)$ for $0 \leq i \leq 15$, where σ is the chosen S-box. We choose $\sigma_0 = [0, 14, 2, 10, 9, 15, 8, 11, 6, 4, 3, 7, 13, 12, 1, 5]$. Details are provided in [3].

In this study, we separate the round function of QARMA into five parts: the *initial tweakey masking* `IT`, *forward round function* `FR`, *middle round function*

MR, *backward round function* BR, and *final tweakey masking* FT. Notably, this separation differs from that of the original design [3]. IT and FT execute only AddRoundTweakey. FR, MR, and BR are redefined as follows:

$$\begin{aligned}
\texttt{FR}(\mathsf{IS}) &= M \circ \tau \circ \texttt{AddRoundTweakey} \circ S(\mathsf{IS}),\\
\texttt{MR}(\mathsf{IS}) &= \overline{S} \circ \overline{\tau} \circ \texttt{AddRoundTweakey} \circ Q \circ \tau \circ S(\mathsf{IS}),\\
\texttt{BR}(\mathsf{IS}) &= \overline{S} \circ \texttt{AddRoundTweakey} \circ \overline{\tau} \circ \overline{M}(\mathsf{IS}).
\end{aligned}$$

This structural separation allows QARMA to be considered to have the same structure as PRINCE and PRINCEv2, i.e., MR has two rounds in these ciphers. In the following, the total number of rounds is expressed as $(r_1{+}2{+}r_2)$ when the number of rounds for FR, MR, and BR are r_1, 2, and r_2, respectively.

Tweak Update. The cells of the tweak are permuted as $h(T) = t_{h(0)}||\cdots||t_{h(15)}$, where h is the same permutation $h = [6, 5, 14, 15, 0, 1, 2, 3, 7, 12, 13, 4, 8, 9, 10, 11]$ used in MANTIS [6]. Then, an LFSR ω updates the tweak cells with indexes 0, 1, 3, 4, 8, 11, and 13. For QARMA64, ω is a maximal period LFSR that maps cell (b_3, b_2, b_1, b_0) to $(b_0 \oplus b_1, b_3, b_2, b_1)$. For QARMA128, it maps cell $(b_7, b_6, \ldots, b_0)$ to $(b_0 \oplus b_2, b_7, b_6, \ldots, b_1)$.

B Basic Algorithms

We describe the basic algorithms employed to construct our framework with the notations in Sect. 2.2. Sun et al. [27, 28] already briefly presented the algorithm to evaluate the clustering effect of a certain differential characteristics. Herein, we describe it more formally, such that we must adjust some parameters appropriately in our new search framework. In the following sections, all our SAT models assume that the target cipher is based on an S-box, permutation, and matrix (XOR), such as PRINCE and QARMA.

Algorithm to Construct a Certain Differential. As a first step to investigate the clustering effect of the differential characteristics, we must find such differential characteristics with a high probability, which are the seeds of differentials in advance. Let $\boldsymbol{C_r} = (\boldsymbol{c_0}, \boldsymbol{c_1}, \ldots, \boldsymbol{c_r})$ be an r-round differential characteristic with the minimum weight W_{min} on an n-bit cipher, where $\boldsymbol{c_i} = (c_{i,0}, c_{i,1}, \ldots, c_{i,n-1})$ for $0 \leq i \leq r$. Note that $\boldsymbol{c_0}$ and $\boldsymbol{c_r}$ imply the input and output differences, respectively. Algorithm 3 shows the procedure to obtain the differential characteristics $\boldsymbol{C_r}$ with the minimum weight W_{min} for an r-round cipher. $\boldsymbol{v_r} = (v_{r,0}, v_{r,1}, \ldots, v_{r,n-1})$ denotes Boolean variables to express the input differences of the r-th round, where $\boldsymbol{v_0}$ and $\boldsymbol{v_r}$ express differences in plaintext and ciphertext, respectively. We describe each function as follows:

Function $\texttt{SAT}_{\texttt{diff.min}}()$**:** This function takes the lower threshold T and the number of target rounds r as inputs. T is basically set to the minimum weight of $(r-1)$ rounds. As outputs, it returns an r-round differential characteristic $\boldsymbol{C_r}$ with the minimum weight W_{min}.

Algorithm 3: The underlying functions to obtain the differential characteristics $\boldsymbol{C_r}$ with the minimum weight W_{min} for an r-round cipher.

Function $\texttt{SAT}_{\texttt{diff.min}}(T, r)$
begin
  $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \texttt{SET}_{\texttt{model}}(T, r)$
  while $\texttt{SAT}_{\texttt{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var})$ = ("UNSAT", $\emptyset$) **do**
    $T \leftarrow T + 1$
    $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \texttt{SET}_{\texttt{model}}(T, r)$
  $W_{min} \leftarrow T$
  return $(\boldsymbol{C_r}, W_{min})$

Function $\texttt{SET}_{\texttt{model}}(T, r)$
begin
  $\mathcal{M}_{var} \leftarrow$ *All Boolean variables to produce all clauses, such as* $\boldsymbol{v_i}$ *for* $0 \leq i \leq r$
  $\mathcal{M}_{SAT} \leftarrow \begin{cases} \mathcal{M}_{cla.matrix} & \text{for all matrices in the } r\text{-round cipher} \\ \mathcal{M}_{cla.sbox} & \text{for all S-boxes in the } r\text{-round cipher} \\ \mathcal{M}_{cla.sec(T)} & \\ \mathcal{M}_{cla.input} & \end{cases}$
  return $(\mathcal{M}_{SAT}, \mathcal{M}_{var})$

Function $\texttt{SAT}_{\texttt{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var})$
begin
  if $\texttt{Solver}(\mathcal{M}_{SAT}, \mathcal{M}_{var})$ = "SAT" **then**
    for $i = 0$ **to** r **do**
      $\boldsymbol{c_i} \leftarrow \boldsymbol{v_i}$
    return ("SAT", $\boldsymbol{C_r}$)
  else
    return ("UNSAT", $\emptyset$)

Function $\texttt{SET}_{\texttt{model}}()$**:** This function takes the weight T and the number of target rounds r as inputs. It is used to set a SAT model to verify whether there is a differential characteristic with the weight of $\leq T$ on the r-round cipher. As outputs, it returns a SAT model $\mathcal{M}_{SAT}$ and its Boolean variables $\mathcal{M}_{var}$.

Function $\texttt{SAT}_{\texttt{diff.char}}()$**:** This function takes a SAT model $\mathcal{M}_{SAT}$ and its Boolean variables $\mathcal{M}_{var}$ as inputs. It is used to check whether the given SAT model is "SAT" or "UNSAT". If a SAT solver $\texttt{Solver}()$ returns "SAT", this function returns "SAT" and Boolean variables to express the input differences of each round which are equal to the differential characteristics $\boldsymbol{C_r}$. Otherwise, it returns "UNSAT" and $\emptyset$. In Algorithm 3, $\texttt{SAT}_{\texttt{diff.min}}()$ does not use $\boldsymbol{C_r}$, whereas $\texttt{SAT}_{\texttt{diff.char}}()$ returns $\boldsymbol{C_r}$ because the algorithm described later utilizes it to take a clustering effect into account.

After obtaining $(\boldsymbol{C_r}, W_{min})$ from $\texttt{SAT}_{\texttt{diff.min}}()$, the clustering effect of $\boldsymbol{C_r}$ is evaluated by Algorithm 4, namely, this algorithm find the differential character-

Algorithm 4: The basic algorithm to evaluate the clustering effect.

Function $\texttt{SAT}_{\texttt{diff.clust}}(T_{low}, T_{upp}, r, (\boldsymbol{c_0}, \boldsymbol{c_r}))$
begin
  $\boldsymbol{N_{T_{upp}-T_{low}+1}} \leftarrow (N_0, N_1, \ldots, N_{T_{upp}-T_{low}})$
  for $i = T_{low}$ *to* T_{upp} **do**
    $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \texttt{SET}_{\texttt{model}}(i, r)$
    add $\mathcal{M}_{cla.clust}$ **to** $\mathcal{M}_{SAT}$
    add *auxiliary Boolean variables of* $\mathcal{M}_{cla.\overline{sec}(i)}$ **to** $\mathcal{M}_{var}$
    add $\mathcal{M}_{cla.\overline{sec}(i)}$ **to** $\mathcal{M}_{SAT}$
    $N_{i-T_{low}} \leftarrow 0$
    while $\texttt{SAT}_{\texttt{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var}) =$ ("SAT", $\boldsymbol{C_r}$) **do**
      $N_{i-T_{low}} \leftarrow N_{i-T_{low}} + 1$
      add $\mathcal{M}_{cla.\overline{clust}}$ **to** $\mathcal{M}_{SAT}$
  return $\boldsymbol{N_{T_{upp}-T_{low}+1}}$

istics with the same input and output differences $(\boldsymbol{c_0}, \boldsymbol{c_r})$ for a specified range of weight. Notably, we do not solve a general SAT problem but an incremental SAT problem to find differential characteristics in Algorithm 4.

As inputs to Algorithm 4, we provide two thresholds T_{low} and T_{upp}, the number of target rounds r, as well as a pair of input and output differences, namely $(\boldsymbol{c_0}, \boldsymbol{c_r})$. We specify the two thresholds T_{low} and T_{upp} as the lower and upper weights taken into account in a clustering effect, respectively. Upon executing Algorithm 4, we obtain a list indicating the number of differential characteristics with $(\boldsymbol{c_0}, \boldsymbol{c_r})$ for each weight. Subsequently, we can compute the probability of the differential $(\boldsymbol{c_0}, \boldsymbol{c_r})$ by applying the formula $\sum_{i=T_{low}}^{T_{upp}} N_{i-T_{low}} \cdot 2^{-i}$.

Note that our new method described later employs a variety of values of T_{low} while T_{low} is generally set to the minimum weight of the differential characteristics W_{min}.

Algorithm to Enumerate All Differential Characteristics in a Certain Weight. Our new method requires all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with a specified range of weight to evaluate their clustering effects and identify the best differential. This can be easily realized by Algorithm 3 with a small modification, as shown in Algorithm 5. However, a SAT problem changes from a general SAT problem to an incremental SAT problem to efficiently find all differential characteristics. As inputs to Algorithm 5, we provide a weight W and the number of target rounds r. Additionally, we can optionally provide the constraint in the input and output differences to search the specific (truncated) differentials, denoted by $(\boldsymbol{d_{in}}, \boldsymbol{d_{out}})$ where $\boldsymbol{d_{in/out}} = (d_{in/out,0}, d_{in/out,1}, \cdots, d_{in/out,n-1}), d_{in/out,i} \in \mathbb{F}_2$. We specify $\boldsymbol{d_{in/out}}$ as the position of inactive bits, that is, the i-th bit in the input/output differences is fixed to 0 if $d_{in/out,i} = 0$. Otherwise, the i-th bit in the input/output differences can take either 1 or 0. After executing Algorithm 5, we obtain a list of all differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with the weight W.

Algorithm 5: Finding all the input and output differences.

Function $\mathtt{SAT}_{\mathtt{diff.all}}(W, r, \boldsymbol{d_{in}}, \boldsymbol{d_{out}})$
begin
  $\boldsymbol{D} = \emptyset$
  $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \mathtt{SET}_{\mathtt{model}}(W, r)$
  for $i = 0$ **to** $n - 1$ **do**
    /* n denotes the index of bits in the input and output differences */
    if $d_{in,i} = 0$ **then**
      add $v_{0,i} \oplus \overline{d_{in,i}}$ **to** $\mathcal{M}_{SAT}$
    if $d_{out,i} = 0$ **then**
      add $v_{r,i} \oplus \overline{d_{out,i}}$ **to** $\mathcal{M}_{SAT}$
  add *auxiliary Boolean variables of* $\mathcal{M}_{cla.\overline{sec}(W)}$ **to** $\mathcal{M}_{var}$
  add $\mathcal{M}_{cla.\overline{sec}(W)}$ **to** $\mathcal{M}_{SAT}$
  while $\mathtt{SAT}_{\mathtt{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var}) =$ ("SAT", $\boldsymbol{C_r}$) **do**
    add $(\boldsymbol{c_0}, \boldsymbol{c_r})$ **to** $\boldsymbol{D}$
    add $\bigvee_{k=0}^{n-1}(v_{0,k} \oplus c_{0,k}) \vee (v_{r,k} \oplus c_{r,k})$ **to** $\mathcal{M}_{SAT}$
  return $\boldsymbol{D}$

C Good Parameters for Algorithm 2

As mentioned in Sect. 3.5, the result by Algorithm 2 depends on the parameters T_t and T_s, i.e., we must appropriately set T_t and T_s to obtain the best differential. However, it is impossible to show the specific parameters of T_t and T_s for any primitive, as they depend on several factors, including the construction of a primitive and the number of rounds. Hence, we provide some experimental results about T_t and T_s on PRINCE and QARMA128 in Table 8. To investigate an effect of a combination of T_t and T_s on a range of results that is as wide as possible, we show the results of all combinations $3 \leq T_t \leq 7$ and $1.1 \leq T_s \leq 2.0$ in increments of 0.1, i.e., 50 combinations.

Table 8 shows that we can obtain the best differential when $5 \leq T_t$ in both cases of PRINCE and QARMA128. In particular, we can obtain the best differential of the six rounds of PRINCE, regardless of a choice of T_t and T_s. This is because the distribution of the differential characteristics for a weight in PRINCE is sparse, and the most contributing differential characteristic to the probability of a differential is the one with a weight of W_{min}. Hence, the probability of the best differential is dominated by that of the differential characteristics with a weight of W_{min}.

In contrast, for QARMA128, we often fail to determine the best differential when T_t is low. This is because the distribution of the differential characteristics is dense, like QARMA64, in contrast to PRINCE, i.e., the differential characteristics with not only a weight of W_{min} but also a weight of $> W_{min}$ contribute to enhancing the probability of a differential.

Table 8: Comparison of several combinations of T_t and T_s.

PRINCE (4 (1+2+1) rounds) $W_{min} = 32,\ T_w = 1,\ T_c = 10,\ N_{thr} = 8$

T_t	3									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$
Time	2h36m23s	2h33m48s	2h33m08s	2h36m26s	2h35m35s	2h35m08s	2h37m57s	2h36m29s	2h35m21s	2h37m09s
T_t	4									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$
Time	3h08m50s	3h11m28s	3h10m50s	3h11m35s	3h06m52s	3h09m32s	3h11m20s	3h11m11s	3h09m16s	3h12m30s
T_t	5									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$
Time	3h42m41s	3h43m46s	3h43m39s	3h41m44s	3h46m42s	3h47m24s	3h46m00s	3h45m41s	3h47m00s	3h49m03s
T_t	6									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$
Time	4h19m52s	4h21m12s	4h20m16s	4h22m59s	4h22m59s	4h19m27s	4h20m31s	4h18m33s	4h19m39s	4h23m02s
T_t	7									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$	$2^{-30.868}$
Time	4h56m43s	4h56m51s	4h53m21s	4h57m08s	4h55m13s	4h54m22s	4h53m10s	4h55m17s	4h53m59s	4h56m03s

QARMA128 under the SK setting (6 (2+2+2) rounds) $W_{min} = 60,\ T_w = 1,\ T_c = 10,\ N_{thr} = 8$

T_t	3									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-55.177}$	$2^{-55.805}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-55.177}$	$2^{-55.177}$	$2^{-55.177}$	$2^{-55.177}$	$2^{-54.494}$	$2^{-54.494}$
Time	7h02m02s	6h47m35s	7h24m59s	6h40m27s	6h46m13s	6h48m24s	6h18m13s	6h16m06s	6h50m11s	7h24m04s
T_t	4									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-54.494}$	$2^{-54.494}$	$2^{-55.177}$	$2^{-55.177}$	$2^{-55.177}$	$2^{-55.177}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$
Time	8h04m30s	8h19m41s	8h56m52s	8h50m04s	8h58m59s	8h59m46s	9h20m47s	8h35m39s	8h58m39s	8h49m44s
T_t	5									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$
Time	9h34m53s	9h31m51s	9h40m25s	9h40m37s	10h02m59s	9h53m17s	10h35m23s	10h24m21s	10h03m32s	10h13m55s
T_t	6									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$
Time	11h24m58s	11h23m24s	10h55m20s	10h44m30s	11h44m56s	12h19m07s	11h22m33s	11h33m24s	11h23m19s	12h43m52s
T_t	7									
T_s	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Prob.	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$	$2^{-54.494}$
Time	12h31m34s	11h25m49s	11h11m23s	11h55m37s	11h47m58s	12h14m15s	12h41m06s	12h39m31s	12h42m19s	13h08m30s

For T_s, it is trivially better to set it to high, but it seems like that the choice of T_s does not have as significant influence on the obtained differential compared with that of T_t. Besides, it also does not so affect a runtime compared with that of T_t. From these observations, we summarize our recommendation for the choice of T_t and T_s as follows:

For a cipher with a big clustering effect like QARMA. For T_t, it is recommended to set it to about half of T_c. For T_s, it is recommended to set it

around 2.0. It must be mentioned that a differential with not the highest, but a high probability can be obtained, even though T_t is set to a low value in our experiments. Therefore, it is another choice to set T_t to a low value if a computational environment is not expensive and finding the best differential is not required.

For a cipher with a small clustering effect like PRINCE. For T_t, it is recommended to set it around one-third of T_c. For T_s, it is recommended to set it around 2.0 as well as for cipher with a high clustering effect like QARMA. Note that our experimental results imply that we can set T_t to a lower value than our recommendation. However, we highly recommend to investigate a feature of a target cipher before setting it to a low value, because that is expected to depend on each cipher.

D Summary of Key Recovery Attacks

To show the key recovery attacks, we search the key-recovery-friendly truncated differentials on QARMA64 and QARMA128 by Algorithm 1. Due to a page limitation, we only give the summary of our key recovery attacks in Table 9. The detailed attack procedure will be given in the full version of this paper.

Table 9 shows that our key recovery attacks do not outperform the best know attacks. However, we would like to emphasize that our attacks are the first key recovery attack based on straightforward differential cryptanalysis with the time and data complexities comparable to the best attacks.

Table 9: Comparison of our results with existing ones regarding key recovery.

Cipher (Setting[†])	Attacked # Rounds	Type[‡]	Outer whitening	Time	Data	Memory	Validity[$]	Reference
QARMA64 (SK)	10 (3+2+5)	MITM	No	$2^{70.1}$	2^{53}	2^{116}	✓	[32]
	10 (3+2+5)	ID	Yes	$2^{119.3}$	2^{61}	2^{72}	×	[30]
	11 (3+2+6)	ID	Yes	$2^{120.4}$	2^{61}	2^{116}	×	[30]
QARMA64 (RT)	10 (2+2+6)	ID	Yes	$2^{125.8}$	2^{62}	2^{37}	×	[33]
	10 (4+2+4)	TD	Yes	$2^{83.53}$	$2^{47.06}$	2^{80}	×	Our
	10 (3+2+5)	TD	Yes	$2^{75.13}$	$2^{47.12}$	2^{72}	✓	Our
	10 (3+2+5)	SS	Yes	$2^{59.0}$	$2^{59.0}$	$2^{29.6}$	✓	[23]
	11 (4+2+5)	TD	Yes	$2^{111.16}$	$2^{34.26}$	2^{108}	×	Our
	11 (4+2+5)	ID	No	$2^{64.92}$	$2^{58.38}$	$2^{63.38}$	✓	[24]
	12 (3+2+7)	ZC/I	Yes	$2^{66.2}$	$2^{48.4}$	$2^{53.7}$	✓	[1]
QARMA128 (SK)	10 (3+2+5)	MITM	No	$2^{141.7}$	2^{105}	2^{232}	✓	[32]
	10 (3+2+5)	ID	Yes	$2^{237.3}$	2^{122}	2^{144}	×	[30]
	11 (3+2+6)	ID	Yes	$2^{241.8}$	2^{122}	2^{232}	×	[30]
QARMA128 (RT)	11 (4+2+5)	TDIB	Yes	$2^{126.1}$	$2^{126.1}$	2^{71}	✓	[23]
	11 (4+2+5)	ID	No	$2^{137.0}$	$2^{111.38}$	$2^{120.38}$	✓	[24]
	11 (7+2+2)	TD	Yes	$2^{104.60}$	$2^{124.05}$	2^{48}	✓	Our
	12 (7+2+3)	TD	Yes	$2^{154.53}$	$2^{108.52}$	2^{144}	×	Our
	12 (3+2+7)	MITM	Yes	$2^{156.06}$	2^{88}	2^{154}	✓	[24]
	13 (8+2+3)	TD	Yes	$2^{238.02}$	$2^{106.63}$	2^{240}	×	Our

[†] SK: Single-Key, RT: Related-Tweak

[‡] MITM: Meet-in-the-Middle, ID: Impossible Differential, TD: Truncated Differential, SS: Statistical Saturation, ZC: Zero-Correlation, I: Integral, TDIB: Tweak Difference Invariant Bias

[$] The designer claims that the multiplication of time and data complexities for QARMA64 and QARMA128 should be less than $2^{128-\epsilon}$ and $2^{256-\epsilon}$ for a small ϵ (e.g., $\epsilon = 2$), respectively. The symbol '✓' indicates that the attack is feasible within the designer's security claim and the symbol '×' indicates otherwise.