# 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 for 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 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 the longest distinguisher for all the variants, which improves 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. We believe that our findings shed light on new structural properties of these important primitives. In the context of key recovery attacks, our framework allows us to derive the key-recovery-friendly truncated differentials for all variants of QARMA. Consequently, we report key recovery attacks based on (truncated) differential cryptanalysis on QARMA for the first time and show these key recovery attacks are competitive with existing other attacks.

**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 the relationship between a differential characteristic and a differential, revealing that a gap between their probabilities can be significant [2,8,22]. 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 [23] demonstrated an interesting case in Simeck32 [39] 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 [25] invested significant efforts into guaranteeing the resistance against differential cryptanalysis in their works. Nevertheless, they were broken by differential cryptanalysis [11, 17]. Furthermore, the best attack to the first low-latency design PRINCE [10] is also (multiple) differential cryptanalysis on 10 (out of 12) rounds proposed by Canteaut et al. [14]. Hence, it is evidently essential 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. [34,35], focused on accelerating the search for an optimal differential characteristic by incorporating the Matsui's bounding conditions [29]. 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 execution on multiple threads. Moreover, it is also of great importance to investigate the impact of the 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* [18], 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 threads 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 [10] 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 | – | [16] |
| | | 6 | SK | D | $2^{62}$ | [2] |
| | | 6 | SK | I | $2^{62}$ | [12] |
| | | 6 | SK | D | $2^{56.42}$ | [14] |
| | | **7** | SK | D | $2^{55.771}$ | Sect. 4.1 |
| QARMA64 | 16 | 6 | SK | ID | – | [38] |
| | | **7** | SK | D | $2^{58.921}$ | Sect. 4.2 |
| | | 4.5 | RT | ID | – | [27] |
| | | 7 | RT | ID | – | [41] |
| | | 8 | RT | SS | $2^{57}$ | [26] |
| | | 9 | RT | ZC/I | $2^{44}$ | [1] |
| | | **10** | RT | D | $2^{60.831}$ | Sect. 4.2 |
| QARMA128 | 24 | 6 | SK | ID | – | [38] |
| | | **10** | SK | D | $2^{121.549}$ | Sect. 4.2 |
| | | 6.5 | RT | ID | – | [27] |
| | | 8 | RT | TDIB | $2^{124.1}$ | [26] |
| | | **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 the longest ones among existing ones. It is important to note that while the previous attacks may have been adjusted for key recovery, identifying the 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 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 [23] 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 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.

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 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.

*Extension for Finding Key-Recovery-Friendly Differentials.* We provide key recovery attacks on QARMA under the RT setting, which are inspired by Guo et al.'s truncated differential attacks [20]. Unlike the existing SAT-based automatic search tools [34, 35], the proposed framework enables us to efficiently construct the key-recovery-friendly truncated differentials with a high probability. To this end, we take a closer look at the round function of QARMA and consider strategies for finding the key-recovery-friendly differential. Then, we specify the conditions for finding the key-recovery-friendly differentials and conduct experiments using the proposed framework. Table 2 lists our key recovery attacks and the previous ones. We note here that the designer [3] 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 $\epsilon$ (e.g., $\epsilon = 2$), respectively. Hence, the attack can be considered to be valid when it is feasible within the designer's security claim.

Our results can be summarized as follows. For QARMA64 under the RT setting, our attack is valid up to 10 rounds. In addition, our 11-round attack cannot outperform the designer's security claim, but we provide the attack that outperforms the standard security notion (i.e., security against the exhaustive search for the secret key). However, the previous best attack on QARMA64 under the RT setting [1] is valid for up to 12 rounds; thus, our attacks cannot outperform the previous best attack. For QARMA128 under the RT setting, our attack is valid up to 11 rounds. This can achieve the best key recovery attack on the 11-round QARMA128 under the RT setting. In addition, our 12- and 13-round attacks cannot outperform the designer's security claim, but we provide the attack that outperforms the standard security notion. However, the previous best attack on QARMA128 under the RT setting [27] is valid for up to 12 rounds; thus, our attacks cannot outperform the previous best attack.

Table 2: 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 | $2^{70.1}$ | $2^{53}$ | $2^{116}$ | ✓ | [40] |
| | 10 (3+2+5) | ID | Yes | $2^{119.3}$ | $2^{61}$ | $2^{72}$ | × | [38] |
| | 11 (3+2+6) | ID | Yes | $2^{120.4}$ | $2^{61}$ | $2^{116}$ | × | [38] |
| QARMA64 (RT) | 10 (2+2+6) | ID | Yes | $2^{125.8}$ | $2^{62}$ | $2^{37}$ | × | [41] |
| | 10 (4+2+4) | TD | Yes | $2^{83.53}$ | $2^{47.06}$ | $2^{80}$ | × | Sect. 5.2 |
| | 10 (3+2+5) | TD | Yes | $2^{75.13}$ | $2^{47.12}$ | $2^{72}$ | ✓ | Sect. 5.2 |
| | 10 (3+2+5) | SS | Yes | $2^{59.0}$ | $2^{59.0}$ | $2^{29.6}$ | ✓ | [26] |
| | 11 (4+2+5) | TD | Yes | $2^{111.16}$ | $2^{34.26}$ | $2^{108}$ | × | Sect. 5.2 |
| | 11 (4+2+5) | ID | No | $2^{64.92}$ | $2^{58.38}$ | $2^{63.38}$ | ✓ | [27] |
| | 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 | $2^{141.7}$ | $2^{105}$ | $2^{232}$ | ✓ | [40] |
| | 10 (3+2+5) | ID | Yes | $2^{237.3}$ | $2^{122}$ | $2^{144}$ | × | [38] |
| | 11 (3+2+6) | ID | Yes | $2^{241.8}$ | $2^{122}$ | $2^{232}$ | × | [38] |
| QARMA128 (RT) | 11 (4+2+5) | TDIB | Yes | $2^{126.1}$ | $2^{126.1}$ | $2^{71}$ | ✓ | [26] |
| | 11 (4+2+5) | ID | No | $2^{137.0}$ | $2^{111.38}$ | $2^{120.38}$ | ✓ | [27] |
| | 11 (7+2+2) | TD | Yes | $2^{104.60}$ | $2^{124.05}$ | $2^{48}$ | ✓ | Sect. 5.3 |
| | 12 (7+2+3) | TD | Yes | $2^{154.53}$ | $2^{108.52}$ | $2^{144}$ | × | Sect. 5.3 |
| | 12 (3+2+7) | MITM | Yes | $2^{156.06}$ | $2^{88}$ | $2^{154}$ | ✓ | [27] |
| | 13 (8+2+3) | TD | Yes | $2^{238.02}$ | $2^{106.63}$ | $2^{240}$ | × | Sect. 5.3 |

[†] 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 $\epsilon$ (e.g., $\epsilon = 2$), respectively. The symbol '✓' indicates that the attack is feasible within the designer's security claim and the symbol '×' indicates otherwise.

– For QARMA64 under the RT setting, our attack is valid up to 10 rounds. In addition, our 11-round attack cannot outperform the designer's security claim, but we provide the attack that outperforms the standard security notion (i.e., security against the exhaustive search for the secret key). However, the previous best attack on QARMA64 under the RT setting [1] is valid for up to 12 rounds; thus, our attacks cannot outperform the previous best attack.
– For QARMA128 under the RT setting, our attack is valid up to 11 rounds. This can achieve the best key recovery attack on the 11-round QARMA128 under the RT setting. In addition, our 12- and 13-round attacks cannot outperform the designer's security claim, but we provide the attack that

outperforms the standard security notion. However, the previous best attack on QARMA128 under the RT setting [27] is valid for up to 12 rounds; thus, our attacks cannot outperform the previous best attack.

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 straightforward differential cryptanalysis can be performed with the time and data complexities comparable to the previous best attack. 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 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.

Finally, we attempt to explore key-recovery-friendly differentials on PRINCE and PRINCEv2 by our tool. Specifically, we take two approaches: (1) using the same approach as in the case of QARMA discussed earlier, and (2) enhancing the probability of differentials utilized in the known best attack proposed by Canteaut et al. [14]. As a result, our tool confirms that the existing differentials [14] are the best ones for the purpose of key recovery attacks.

## 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 a 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 [24] 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 [15]. 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 are more powerful than MILP-based ones as shown in [35]. 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 by 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. [34, 35]. We give the detailed modeling method for each operation in Appendix B. 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 [4,33,37]. Among these, we employ *Sequential Encoding Method* [33] 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 [34]. $\sum_{j=0}^{r \cdot i-1} p_j = B$ 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 [34]. 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:

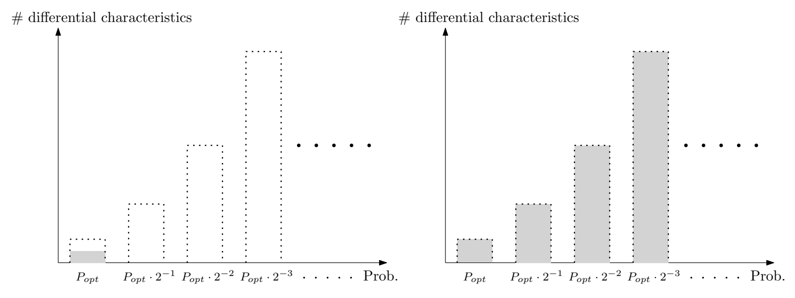
**Speedy identification of a good differential.** Most of the existing studies on the solver-aided search methods have focused on searching for the optimal differential characteristics as efficiently as possible. In contrast, our framework aims to identify a good differential among numerous differential 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 the 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. [35] 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 symmetric-key primitives.

### 3.1 Our Approach

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



(a) Conventional approach (b) Our approach

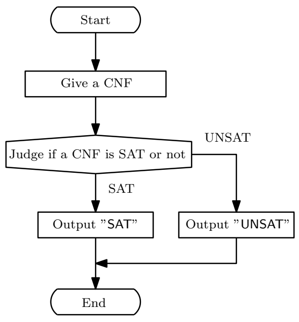
Fig. 1: Approaches to identifying a good differential. "# differential characteristics" denotes the total number of differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$ with the corresponding probability in horizontal axis. $P_{opt}$ denotes the probability of an optimal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

egy seems reasonable in many cases; therefore, most of the 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 [23].

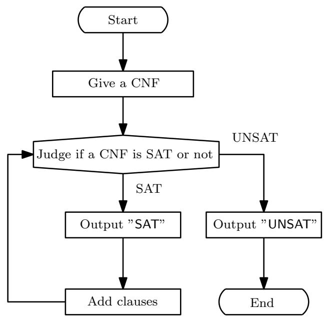
To investigate differentials in more detail, we need to evaluate a clustering effect of numerous differential characteristics having different $(\boldsymbol{c_0}, \boldsymbol{c_r})$. Since this requires a huge computational cost, it is a time-consuming task even with the state-of-the-art approach, such as a pure SAT-based automatic search method proposed by Sun et al. [35]. 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* [18] 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 incremental SAT problems.



(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 [19], 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 gain as the degree of the parallelization. From this fact, our work starts by 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 the ones presented in [35]. We leave a detailed explanation of them in Appendix C.

[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: $\boldsymbol{D}, \boldsymbol{N}$
1  begin
2      $\boldsymbol{D} \leftarrow (\boldsymbol{D_0}, \boldsymbol{D_1}, \ldots, \boldsymbol{D_{T_w-1}})$
3      $\boldsymbol{N} \leftarrow (\boldsymbol{N_0}, \boldsymbol{N_1}, \ldots, \boldsymbol{N_{T_w-1}})$
4      for $i = W_{min}$ to $W_{min} + T_w - 1$ do
5          $\boldsymbol{D_{i-W_{min}}} \leftarrow \texttt{SAT}_{\texttt{diff.all}}(i, r, \mathbf{1}, \mathbf{1})$
6          $\boldsymbol{N_{i-W_{min}}} \leftarrow \emptyset$
7          $j \leftarrow 0$
8          for all pairs in $\boldsymbol{D_{i-W_{min}}}$ do
9              add $\texttt{SAT}_{\texttt{diff.clust}}(i, i + T_c - 1, r, \boldsymbol{D}^{(j)}_{\boldsymbol{i-W_{in}}})$ to $\boldsymbol{N_{i-W_{in}}}$
10             $j \leftarrow j + 1$
11             /* $j$ denotes the index of $\boldsymbol{D_{i-W_{in}}}$, i.e., $MAX(j) = |\boldsymbol{D_{i-W_{in}}}|$ */
12     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 $\texttt{SAT}_{\texttt{diff.min}}()$.
**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 $\texttt{SAT}_{\texttt{diff.all}}()$.
**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 $\texttt{SAT}_{\texttt{diff.min}}()$ 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 $\texttt{SAT}_{\texttt{diff.min}}()$ 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.

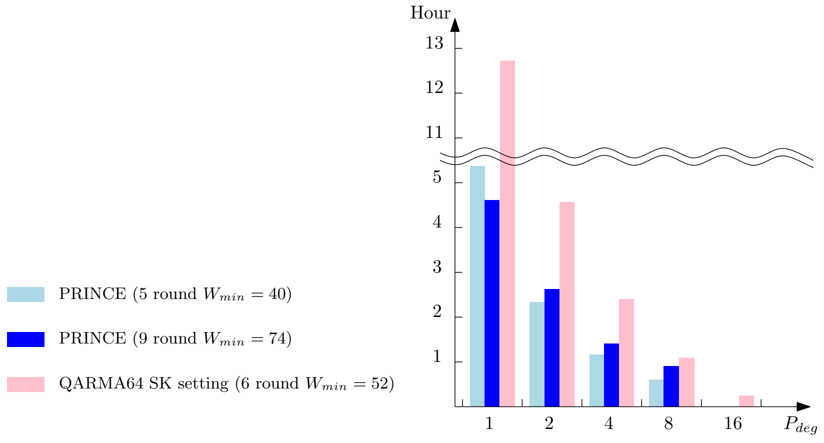


Fig. 3: The runtime for each environment according to Eq. (1). The evaluations of PRINCE and QARMA are conducted on computers with 8 and 16 threads, respectively. $W_{min}$ denotes the weight of the differential characteristics evaluated for the clustering effect.

$(P_{deg} = 8, T_m = 8, T_s = 1)$ 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 [21]. 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 multiple 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 emphasize 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: $(\boldsymbol{c_{opt.in}}, \boldsymbol{c_{opt.out}}), P_{opt}$
 1 begin
 2     $P_{opt} \leftarrow 0$, $\boldsymbol{P_{thr}} \leftarrow (P^0_{thr}, P^1_{thr}, \ldots, P^{N_{thr}-1}_{thr})$
 3     $\boldsymbol{D} \leftarrow (\boldsymbol{D_0}, \boldsymbol{D_1}, \ldots, \boldsymbol{D_{N_{thr}-1}})$
 4     for $i = W_{min}$ to $W_{min} + T_w - 1$ do
 5         $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \mathtt{SET_{model}}(i, r)$
 6         add auxiliary Boolean variables of $\mathcal{M}_{cla.\overline{sec}(i)}$ to $\mathcal{M}_{var}$
 7         add $\mathcal{M}_{cla.\overline{sec}(i)}$ to $\mathcal{M}_{SAT}$
 8         count $\leftarrow 0$
 9         /* incremental SAT problem                                        */
10         while $\mathtt{SAT_{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var}) =$ (“SAT”, $\boldsymbol{C_r}$) do
11             $\boldsymbol{D_{count \bmod N_{thr}}} \leftarrow (\boldsymbol{c_0}, \boldsymbol{c_r})$
12             count $\leftarrow$ count $+ 1$
13             if count mod $N_{thr} = 0$ then
14                 for each thread do
15                     $P^{thread}_{thr} \leftarrow \mathtt{Thread}(i, r, T_c, T_s, T_t, P_{opt}, \boldsymbol{D_{thread}})$
16                 if $MAX(\boldsymbol{P_{thr}}) > P_{opt}$ then
17                     $(\boldsymbol{D_{opt}}, P_{opt}) \leftarrow MAX(\boldsymbol{D}, \boldsymbol{P_{thr}})$
18             add $\bigvee_{k=0}^{n-1} (v_{0,k} \oplus c_{0,k}) \vee (v_{r,k} \oplus c_{r,k})$ to $\mathcal{M}_{SAT}$
19         if count mod $N_{thr} \neq 0$ then
20             for each thread do
21                 $P^{thread}_{thr} \leftarrow \mathtt{Thread}(i, r, T_c, T_s, T_t, P_{opt}, \boldsymbol{D_{thread}})$
22             if $MAX(\boldsymbol{P_{thr}}) > P_{opt}$ then
23                 $(\boldsymbol{D_{opt}}, P_{opt}) \leftarrow MAX(\boldsymbol{D}, \boldsymbol{P_{thr}})$
24     return $(\boldsymbol{D_{opt}}, P_{opt})$

25 Function Thread($W, r, T_c, T_s, T_t, P_{opt}, \boldsymbol{D}$) // A multi-threading technique
26 begin
27     $\boldsymbol{N} \leftarrow (N_0, N_1, \ldots, N_{T_c-1})$
28     $\boldsymbol{N} \leftarrow \mathtt{SAT_{diff.clust}}(W, W + T_t - 1, r, \boldsymbol{D})$
29     $P_{tmp} \leftarrow \sum_{i=W}^{W+T_t-1} (N_{i-W} \cdot 2^{-i})$
30     if $T_s \cdot P_{tmp} > P_{opt}$ then
31         $\boldsymbol{N} \leftarrow \mathtt{SAT_{diff.clust}}(W + T_t, W + T_c - 1, r, \boldsymbol{D})$
32         $P_{tmp} \leftarrow P_{tmp} + \sum_{i=W+T_t}^{W+T_c-1} (N_{i-W} \cdot 2^{-i})$
33     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 3: 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 D, 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 3 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 combinations of $T_t$ and $T_s$.

Table 3 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, 14]. 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 4 shows the results of QARMA64 and QARMA128, both of which are evaluated on three Linux machines 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 4, 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 [38] 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, 26] 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 [23] 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 4: 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 5: 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 $64 \times 64$ matrix, the matrix of PRINCE consists of two $16 \times 16$ matrices $\widehat{M}^{(0)}$ and $\widehat{M}^{(1)}$ while that of QARMA64 consists of only one $16 \times 16$ 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 5 and 6 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 6: 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 good resistance to the clustering effect.

Finally, the case reported by Kölbl and Roy [23] 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 Key Recovery Attacks on QARMA

In this section, we provide key recovery attacks on QARMA. First, we discuss two strategies to search for key-recovery-friendly differentials under both single-key (SK) and related-tweak (RT) settings. When we use the best differentials presented in Sect. 4.2 for key recovery attacks, we cannot append more than two key-recovery rounds to the differential distinguisher. In contrast, key-recovery-friendly differentials allow us to append at most four key-recovery rounds to the differential distinguisher. Next, we present key-recovery-friendly differentials derived based on the proposed search strategies; and then, we obtain the best truncated differential from key-recovery-friendly differentials for our key recovery attacks. Finally, we launch key recovery attacks using the best truncated differentials. Our attacks are inspired by Guo et al.'s truncated differential attacks [20].

Hereafter, we denote the $r$-round internal state matrix and the $r$-round tweak matrix by $\mathsf{IS}^{(r)} = s_0^{(r)}||s_1^{(r)}|| \cdots ||s_{15}^{(r)} = s_{\{0,1,\ldots,15\}}^{(r)}$ and $T^{(r)} = t_0^{(r)}||t_1^{(r)}|| \cdots ||t_{15}^{(r)} = t_{\{0,1,\ldots,15\}}^{(r)}$, respectively.

## 5.1 Key-Recovery-Friendly Differentials for QARMA

**Search Strategy for QARMA under SK Setting.** We discuss here a strategy to search for key-recovery-friendly differentials for QARMA under the SK setting. Before proceeding with our discussion, we provide the following properties of the `MixColumns` and `ShuffleCellsInv` operations in the round function.

*Property 1.* Regarding the propagation of active cells through the `MixColumns` operation, when there exists only one active cell in the input column to the operation, the corresponding cell position is always inactive after the operation, but the others in the corresponding output column are always active. In contrast, there exist two or more active cells in the input column to the operation, at most four cells in the corresponding output column are active after the operation.

This is directly derived from Fig. 5 in the literature [3]. We should assume the worst case when considering the propagation of active cells through key-recovery rounds. For this reason, when there exist two or more active cells in the input column to the `MixColumns` operation, we assume that four cells in the corresponding output column are always active after the operation.

*Property 2.* Regarding the propagation of active cells through the `ShuffleCellsInv` operation, when there exist four active cells in a certain column to the operation, these active cells are evenly propagated across the four columns after the operation. For example, assuming that four cells in the first column are active (i.e., the active cells of $\mathtt{IS}^{(r)}$ are 0th, 4th, 8th, and 12th cell positions), these active cells are evenly propagated across the first, third, second, and fourth column after the `ShuffleCellsInv` operation, respectively.

This is also directly derived from the `ShuffleCells` operation, as described in Appendix A.

Based on these properties, we first examine to append some key-recovery rounds to the differential distinguisher. As an example, we use the best 6-round differential under the SK setting. In this case, the active cells of both input and output differentials are 0th, 1st, 10th, and 11th cell positions; thus, there exists one active cell in each column at the input and output differentials. We note here that the input differential is the input matrix to the `SubCells` operation in the forward round function (`FR`) and the output differential is the output matrix from the `SubCellsInv` operation in the backward round function (`BR`). In other words, to append some key-recovery rounds to the differential distinguisher, we examine the propagation of active cells through the `MixColumns` operation as the first step in both `FR` and `BR` directions. From Property 1, 12 cells except for the 0th, 1st, 10th, and 11th cell positions are always active after the first `MixColumns` operation. The next step is the `ShuffleCellsInv` operation. From

Property 2, 12 cells except for the 0th, 3rd, 9th, and 11th cell positions are always active after the first `ShuffleCellsInv` operation. This means that there exist more than two active cells in each column after the first `ShuffleCellsInv` operation. Subsequently, the propagation of active cells is not changed during the `AddRoundTweakey` and `SubCellsInv` operations. This completes the process of appending one key-recovery round in both `FR` and `BR` directions. Here, we further examine to append one more key-recovery round in both `FR` and `BR` directions. Similarly in the above steps, from Property 1, all cells are always active after the second `MixColumns` operation. This is because there exist two or more active cells in each column before the second `MixColumns` operation. It is obvious that a key recovery attack is infeasible when all cells become active; therefore, when we use best differentials presented in Sect. 4.2 for key recovery attacks, we cannot append more than two key-recovery rounds in total to the differential distinguisher.

Next, we consider how to append as many key-recovery rounds as possible while avoiding all cells becoming active. To resolve this issue, we provide the following property.

*Property 3.* One active cell is propagated across at least three columns of the state matrix during the `MixColumns` and `ShuffleCellsInv` operations.

This is directly derived from Properties 1 and 2. Property 3 implies that to minimize the propagation of active cells, we need to examine a strategy to search for input/output differentials in such a way that active cells are propagated across only three columns of the state matrix (i.e., all cells in a certain column of the state matrix are inactive) after the first `ShuffleCellsInv` operation in the key-recovery round. To tackle this task, we provide the following property.

*Property 4.* When the active cell positions of the input/output difference correspond to any combination in one of the following four patterns, all cells in a certain column of the state matrix are always inactive after the first `ShuffleCellsInv` operation in the key-recovery round:

$$(0,\ 13,\ 11,\ 6),\ (5,\ 8,\ 14,\ 3),\ (15,\ 2,\ 4,\ 9),\ \text{or}\ (10,\ 7,\ 1,\ 12).$$

This is also directly derived from Properties 1 and 2. As an example, we assume that the active cells in an input differential are 0th, 13th, 11th, and 6th cell positions. We note that any combination of these four active cell positions (e.g., 0th and 11th cell positions, etc) also satisfies Property 4. In this case, from Property 1, 12 cells except for the 0th, 13th, 11th, and 6th cell positions are always active after the first `MixColumns` operation. Subsequently, from Property 2, 12 cells except for the 0th, 4th, 8th, and 12th cell positions (i.e., all cells in the first column of the state matrix) are always active after the first `ShuffleCellsInv` operation. Finally, considering the propagation of active cells when appending two key-recovery rounds to the `FR` or `BR` direction, the 0th, 5th, 10th, and 15th cell positions are always inactive.

We conclude that a strategy to search for key-recovery-friendly differentials for QARMA under the SK setting is equivalent to finding input/output differentials such that Property 4 is satisfied. To implement this strategy, all we have to do is to add a constraint into Algorithm 1 that controls input/output differentials based on the strategy.

**Search Strategy for QARMA under RT Setting.** We discuss here a strategy to search for key-recovery-friendly differentials for QARMA under the RT setting. The only difference between the SK and RT settings is whether tweak differences are used or not. This fact implies that based on the search strategy for the SK setting, we need to examine a new strategy to search for input/output and tweak differentials in such a way that tweak differences do not accelerate the increase in the number of active cells. To tackle this task, we provide the following property.

*Property 5.* Assuming that two key-recovery rounds are appended to the `FR` direction from the differential distinguisher. Regarding four active cell patterns for the input differential described in Property 4, when the active cell positions of the tweak differential correspond to any combination of the pattern shown in the following table, such a tweak differential does not accelerate the increase in the number of active cells:

Table 7: Constraints for active cell positions of the input/output and tweak differentials.

| Input/Output differential | Tweak differential |
|---|---|
| 0, 13, 11, 6 | 0, 3, 10, 13, 14 |
| 5, 8, 14, 3 | 3, 4, 8, 9, 11, 12, 14 |
| 15, 2, 4, 9 | 1, 3, 5, 7, 9, 15 |
| 10, 7, 1, 12 | 1, 5, 6, 10, 11 |

As an example, assuming that the active cell positions in an input differential are $s^{(2)}_{\{0,13,11,6\}}$. In this case, retracing the propagation of active cells to the `FR` direction, at least $t^{(2)}_{\{0,4,8,12\}}$ must be inactive. This is because $s^{(2)}_{\{0,4,8,12\}}$ are always inactive when executing the `AddRoundTweakey` operation of $T^{(2)}$ (i.e., after the first `ShuffleCellsInv` operation). Similarly, further retracing to the `FR` direction, $t^{(1)}_{\{0,5,10,15\}}$ and $t^{(0)}_{\{0,5,10,15\}}$ must be inactive. From these conditions, considering the propagation of inactive cells based on the tweak update function, all cells except for $t^{(3)}_{\{0,3,10,13,14\}}$ must be inactive. Therefore, activating $t^{(3)}_{\{0,3,10,13,14\}}$ does not accelerate the increase in number of active cells.

We conclude that a strategy to search for key-recovery-friendly differentials for QARMA under the RT setting is equivalent to finding input/output and tweak differentials such that Properties 4 and 5 are satisfied. To implement this strategy, all we have to do is to add a constraint into Algorithm 1 that controls input/output and tweak differentials based on the strategy. We note here that once a tweak differential is determined, all tweak differences are uniquely

Table 8: Best truncated differentials of QARMA.

| Cipher (Setting) | Rounds | Constraints for active cell positions | | | Probability |
|---|---|---|---|---|---|
| | | Input differential | Tweak differential | Output differential | |
| QARMA64 (SK) | 6 (2+2+2) | 0 | – | 0 | $2^{-46.171}$ |
| | 7 (2+2+3) | 0 | – | 5, 10, 15 | $2^{-48.582}$ |
| QARMA64 (RT) | 6 (1+2+3) | 13 | 13 | 2, 3, 6, 7, 11, 13, 14 | $2^{-11.947}$ |
| | 7 (2+2+3) | 4, 9 | 1, 9 | 0, 11, 12, 14, 15 | $2^{-18.296}$ |
| | 8 (2+2+4) | 4, 9 | 1, 9 | 1, 2, 5, 11 | $2^{-31.565}$ |
| | 9 (2+2+5) | 4, 9 | 1, 9 | 0, 7, 12, 15 | $2^{-48.230}$ |
| QARMA128 (SK) | 7 (3+2+2) | 5, 14 | – | 0, 1, 4, 5, 10, 11 | $2^{-63.660}$ |
| | 8 (4+2+2) | 0, 11 | – | 1, 4, 14 | $2^{-86.459}$ |
| | 9 (4+2+3) | 0, 11 | – | 0, 1, 3, 4, 5, 6, 12, 14, 15 | $2^{-102.261}$ |
| QARMA128 (RT) | 7 (2+2+3) | 4, 9 | 1, 9 | 0, 1, 4, 5, 11, 12, 15 | $2^{-17.986}$ |
| | 8 (2+2+4) | 4, 9 | 1, 9 | 2, 4, 5, 7, 11, 13, 14 | $2^{-33.625}$ |
| | 9 (2+2+5) | 4, 9 | 1, 9 | 0, 2, 5, 12, 13, 14 | $2^{-60.789}$ |
| | 10 (6+2+2) | 4, 9 | 1, 9 | 1, 4, 11, 14 | $2^{-80.431}$ |

determined by the tweak update function. This means that when considering a tweak differential such that Property 5 is satisfied, tweak differences cannot be controlled in the BR direction.

**Best Truncated Differentials for QARMA.** To obtain key-recovery-friendly differentials, we implement the proposed two search strategies into our SAT-based method (specifically, Algorithm 1) and then apply it to QARMA under the SK and RT settings. Our experimental procedure and environment are the same as described in Sect. 4.

As mentioned in the beginning of this section, our key recovery attacks are inspired by Guo et al.'s truncated differential attacks [20]. For this reason, we provide here the best truncated differentials based on the obtained key-recovery-friendly differentials for use in the following subsections. Table 8 shows the best truncated differentials of QARMA under the SK and RT settings. Due to page limitations, the table lists only the experimental results with the lowest number of active cells and the highest truncated differential probability in each target round. It can be derived from the table that only in the case of the 6-round QARMA64 under the SK setting, the obtained truncated differentials enable us to append two key-recovery rounds before and after the differential distinguisher, respectively. In contrast, the other cases enable us to append two and one key-recovery rounds before and after the differential distinguisher, respectively.

We use these truncated differentials to launch our key recovery attacks on QARMA in the following subsections.

### 5.2 Truncated Differential Attacks on Reduced-Round QARMA64

In this subsection, we provide truncated differential attacks on reduced-round QARMA64, especially under the RT setting. Incidentally, under the SK setting, our attack was invalid more than eight rounds, whereas existing studies [38, 40] reported that key recovery attacks were valid up to 10 and 11 rounds[8].

Under the RT setting, the noticeable studies [1, 26, 27] reported that key recovery attacks were valid up to 10, 11, and 12 rounds. Ankele et al. [1] proposed a related-tweak zero-correlation linear attack on 12-round QARMA64 with the time, data, and memory complexities of $2^{66.2}$, $2^{48.4}$, and $2^{53.7}$, respectively. Li et al. [26] proposed a related-tweak statistical saturation attack on 10-round QARMA64 with the time, data, and memory complexities of $2^{59.0}$, $2^{59.0}$, and $2^{29.6}$, respectively. Liu et al. [27] presented a related-tweak impossible differential attack on 11-round QARMA64 with the time, data, and memory complexities of $2^{64.92}$, $2^{58.38}$, and $2^{63.38}$, respectively. We note that the Liu et al.'s attack did not consider the outer whitening key. From these reports, our targets are 10 or more rounds of QARMA64 under the RT setting, and our attacks do consider the outer whitening key.

We first focused on the use of the 9-round truncated differential listed in Table 8, but we were faced with the fact that it would be difficult to launch an attack with this truncated differential. For this reason, we provide here key recovery attacks on the 10- and 11-round QARMA64 under the RT setting with the 8-round truncated differential listed in Table 8. We can only append at most three key-recovery rounds to the 8-round truncated differential distinguisher; thus, it must be mentioned that the feasibility of our attacks is up to 11 rounds.

**10-round Attack on QARMA64 under RT Setting.** To launch our 10-round attack, we append one key-recovery round before and after the 8-round truncated differential distinguisher. Fig. 4 shows the propagation of active cells for the 10-round attack on QARMA64 under the RT setting, where the gray cubes represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$). We note that $a^{(1)}$, $a^{(9)}$, and $tk^{(2)}$ represent the target input, output, and tweak differences, respectively. Our attack is based on the following two phases: data collection and key recovery phases.

*Data Collection Phase.* From Fig. 4, the plaintext $P_{\{0,1,4,5,8,11,12,15\}}$ and the tweak $T^{(0)}_{\{1,8\}}$ are always active. In addition, $\Delta a^{(0)}_8 = \Delta P_8 \oplus \Delta T^{(0)}_8 = 0$ (i.e., $\Delta P_8 = \Delta T^{(0)}_8$) must hold. Under these conditions, the details of the data collection phase are described as follows.

**Step 1:** We prepare $\mathcal{S}$ structures. In each structure, there are $2^{36}$ plaintext-tweak pairs such that the plaintext $P$ and the tweak $T^{(0)}$ satisfying the above

[8] The attacks in [40] did not consider the outer whitening key. Besides, the complexity of the attacks in [38] beyond the designer's security claims.

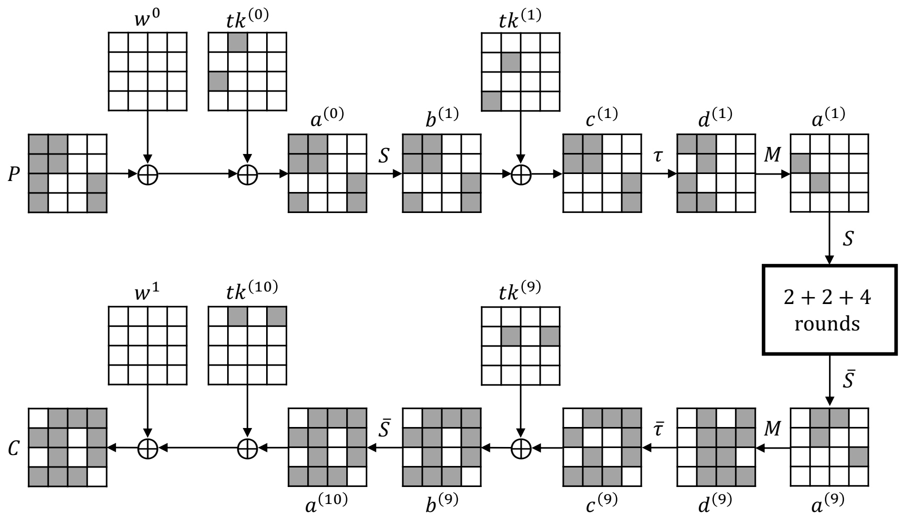


Fig. 4: Propagation of active cells for the 10-round attack on QARMA64 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$).

conditions traverse all possible values while the remaining cells are fixed to some random constants. Each structure leads to approximately $(2^{36})^2/2 = 2^{71}$ differential pairs.

**Step 2:** For each structure, we simulate the encryption oracle to get the ciphertexts $C$ from $(P, T^{(0)})$ pairs. Then, we insert every ciphertext with its corresponding plaintext-tweak pair $(P, T^{(0)}, C)$ into a hash table $\mathbb{H}$ at index $C_{\{0,6,8,10,15\}}$. Thus, each pair from the same index of $\mathbb{H}$ satisfies the differential pattern of $\Delta C_{\{0,6,8,10,15\}} = 0$. All pairs violating its differential pattern are discarded without further processing because these pairs have no hope to comply with the target truncated differential. This step reduces the differential pair used in the next phase from $2^{71} \times \mathcal{S}$ to $2^{51} \times \mathcal{S}$.

*Key Recovery Phase.* Hereafter, we denote $wk^0 = w^0 \oplus k^0$ and $wk^1 = w^1 \oplus k^0$. In the key recovery phase, we try to recover all bits of $wk^0$ and $wk^1$. Once these bits are recovered, all of the outer whitening and core keys (i.e., $w^0$, $w^1$, $k^0$, and $k^1$) can be naturally obtained based on the key specialization of QARMA (see Appendix A). The details of the key recovery phase are described as follows.

**Step 1:** For each possible 12-bit value of $wk^0_{\{5,10,15\}}$, we compute $\Delta a^{(1)}_{\{0,4,8,12\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{0,8,12\}} = 0$. This step leads to a 12-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{39} \times \mathcal{S}$.

**Step 2:** For each possible 12-bit value of $wk^0_{\{1,4,11\}}$, we compute $\Delta a^{(1)}_{\{1,5,9,13\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{1,5,13\}} = 0$. This step leads to a 12-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{27} \times \mathcal{S}$.

**Step 3:** For each possible 4-bit value of $wk^0_{12}$, we compute $\Delta c^{(1)}_{12}$ from the plaintexts $P$ and check whether $\Delta c^{(1)}_{12} = 0$. This step leads to a 4-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{23} \times \mathcal{S}$. Here, we obtain $N = 2^{23} \times \mathcal{S}$ pairs that match the input truncated differential of the 8-round truncated distinguisher.

**Step 4:** For each possible 4-bit value of $wk^1_5$, we compute $\Delta c^{(9)}_5$ from the ciphertexts $C$ and check whether $\Delta c^{(9)}_5 = 0$. This step leads to a 4-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{19} \times \mathcal{S}$.

**Step 5:** For each possible 12-bit value of $wk^1_{\{3,9,12\}}$, we compute $\Delta a^{(9)}_{\{2,6,10,14\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(9)}_{\{6,10,14\}} = 0$. This step leads to a 12-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^7 \times \mathcal{S}$.

**Step 6:** For each possible 12-bit value of $wk^1_{\{2,7,13\}}$, we compute $\Delta a^{(9)}_{\{3,7,11,15\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(9)}_{\{3,7,15\}} = 0$. This step leads to a 12-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{-5} \times \mathcal{S}$.

**Step 7:** For each possible 16-bit value of $wk^1_{\{1,4,11,14\}}$, we compute $\Delta a^{(9)}_{\{1,5,9,13\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(9)}_{\{9,13\}} = 0$. This step leads to a 8-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{-13} \times \mathcal{S}$.

**Step 8:** For the 72-bit key guesses in total, we prepare counters to confirm the number of right pairs that validate the differential pairs of the 8-round truncated differential distinguisher. The number of right pairs follows a binomial distribution with parameters $(N, p_0 = 2^{-31.565})$ in the case of the good key and $(N, p_1 = 2^{-48})$ otherwise.

**Step 9:** We fix the threshold as $\Upsilon$, and the key guess will be accepted as a candidate if the counter of right pairs is no less than $\Upsilon$. For all surviving candidates for the 72-bit key, we exhaustively search for the remaining key bits with at most two plaintext-ciphertext pairs.

*Complexity Analysis.* We apply the method in [9] to estimate the complexity. We denote here $\alpha$ and $\beta$ as the non-detection error probability and the false alarm error probability, respectively. Then, we have

$$\beta \overset{N\to\infty}{\sim} \frac{(1-p_1)\sqrt{\Upsilon/N}}{(\Upsilon/N - p_1)\sqrt{2\pi N(1-\Upsilon/N)}} \exp\left[-N \cdot D\left(\frac{\Upsilon}{N} \middle\| p_1\right)\right], \tag{2}$$

$$\alpha \overset{N\to\infty}{\sim} \frac{p_0\sqrt{1-(\Upsilon-1)/N}}{(p_0 - (\Upsilon-1)/N)\sqrt{2\pi(\Upsilon-1)}} \exp\left[-N \cdot D\left(\frac{\Upsilon-1}{N} \middle\| p_0\right)\right], \tag{3}$$

Table 9: Detailed computation of the time complexity for the 10-round attack.

| Step | Guessed key bits | Time complexity | # of remaining pairs | Note |
|---|---|---|---|---|
| Step 1 | $wk^0_{\{5,10,15\}}$ | $2^{51} \times 2^{12} \times \frac{1}{10} \times \mathcal{S} \approx 2^{59.67} \times \mathcal{S}$ | $2^{51} \times 2^{-12} \times \mathcal{S} = 2^{39} \times \mathcal{S}$ | 1-round enc |
| Step 2 | $wk^0_{\{1,4,11\}}$ | $2^{39} \times 2^{24} \times \frac{1}{10} \times \mathcal{S} \approx 2^{59.67} \times \mathcal{S}$ | $2^{39} \times 2^{-12} \times \mathcal{S} = 2^{27} \times \mathcal{S}$ | 1-round enc |
| Step 3 | $wk^0_{12}$ | $2^{27} \times 2^{28} \times \frac{1}{10} \times \mathcal{S} \approx 2^{51.67} \times \mathcal{S}$ | $2^{27} \times 2^{-4} \times \mathcal{S} = 2^{23} \times \mathcal{S}$ | 1-round enc |
| Step 4 | $wk^1_5$ | $2^{23} \times 2^{32} \times \frac{1}{10} \times \mathcal{S} \approx 2^{51.67} \times \mathcal{S}$ | $2^{23} \times 2^{-4} \times \mathcal{S} = 2^{19} \times \mathcal{S}$ | 1-round dec |
| Step 5 | $wk^1_{\{3,9,12\}}$ | $2^{19} \times 2^{44} \times \frac{1}{10} \times \mathcal{S} \approx 2^{59.67} \times \mathcal{S}$ | $2^{19} \times 2^{-12} \times \mathcal{S} = 2^{7} \times \mathcal{S}$ | 1-round dec |
| Step 6 | $wk^1_{\{2,7,13\}}$ | $2^{7} \times 2^{56} \times \frac{1}{10} \times \mathcal{S} \approx 2^{59.67} \times \mathcal{S}$ | $2^{7} \times 2^{-12} \times \mathcal{S} = 2^{-5} \times \mathcal{S}$ | 1-round dec |
| Step 7 | $wk^1_{\{1,4,11,14\}}$ | $2^{-5} \times 2^{72} \times \frac{1}{10} \times \mathcal{S} \approx 2^{63.67} \times \mathcal{S}$ | $2^{-5} \times 2^{-8} \times \mathcal{S} = 2^{-13} \times \mathcal{S}$ | 1-round dec |
| Step 9 | remaining bits | $2^{128} \times \beta \times (1 - 2^{-64})$ | - | 10-round enc |

where $D(p\|q) \overset{\Delta}{=} p \cdot \ln(\frac{p}{q}) + (1-p) \cdot \ln(\frac{1-p}{1-q})$ is the Kullback-Leibler divergence between two Bernoulli probability distributions with parameters being $p$ and $q$, respectively.

Table 9 lists the detailed computation of the time complexity for the 10-round attack. As with the Guo et al.'s work [20], we try to select the values of $\Upsilon$ and $\mathcal{S}$ such that the following two conditions are validated simultaneously:

- the success probability $P_S = 1 - \alpha$ is not lower than 80%;
- the overall time complexity of the attack is minimized.

When we set the threshold as $\Upsilon = 4$ and the number of structures as $\mathcal{S} = 2^{11.12}$, we derive the success probability of $P_S = 80.0\%$ and the false alarm error probability of $\beta = 2^{-60.07}$. To summarize, the 10-round attack on QARMA64 under the RT setting is feasible with the time, data, and memory complexities of $2^{75.13}$, $2^{47.12}$, and $2^{72.00}$, respectively.

*Discussion.* We also apply the best differentials presented in Sect. 4.2 for the 10-round attack under the RT setting. Specifically, we append one key-recovery round before and after the 8-round truncated differential distinguisher with the probability of $2^{-28.236}$ and use the active cell positions of input, tweak, and output differentials as (3, 6, 12), (2, 6), and (3, 6, 12), respectively. Due to page limitations, we omit the details of the attack, but the attack can be applied in a similar way as the above procedure. To summarize, the attack with the best differential is performed with the time, data, and memory complexities of $2^{83.53}$, $2^{47.06}$, and $2^{80.00}$, respectively, and the success probability of $P_S = 80.1\%$. It can be seen from these results that the 10-round attack with the key-recovery-friendly differential is about $2^{8.4}$ times more efficient in terms of the time complexity than that with the best differential. Therefore, our results suggest that finding key-recovery-friendly differentials rather than the best differentials is of critical importance for key recovery attacks based on differential cryptanalysis.

Regarding the designer's security claim, the multiplication of time and data complexities (TD) for QARMA64 should be less than $2^{128-\epsilon}$ for a small $\epsilon$ (e.g.,

$\epsilon = 2$). From this point of view, our 10-round attack is valid because $\mathsf{TD} = 2^{75.13} \times 2^{47.12} = 2^{122.25} < 2^{126}$. However, our 10-round attack cannot outperform the existing best one [26] since its performance is $\mathsf{TD} = 2^{59.0} \times 2^{59.0} = 2^{118.0} < 2^{122.25}$. Li et al's work [26] is based on the technique using the bias of a linear hull called the key difference invariant bias; thus, it can be regarded as one of the fields of linear cryptanalysis. In contrast, our work provides the first key recovery attack based on straightforward differential cryptanalysis. As described in Sect. 3.3, the proposed SAT-based method enables us to obtain the best differentials under the specified conditions. Thanks to this, our key recovery attack based on the straightforward differential cryptanalysis can be performed with the time and data complexities comparable to the existing best attack on the 10-round QARMA64 under the RT setting. Therefore, we believe that this is a significant contribution in terms of the tight security analysis of QARMA64.

**11-round Attack on QARMA64 under RT Setting.** Our 11-round attack can be launched in a similar way as described in Sect. 5.2. We provide the details of the attack in Appendix E. As a result, the 11-round attack on QARMA64 under the RT setting is feasible with the time, data, and memory complexities of $2^{111.16}$, $2^{34.26}$, and $2^{111.00}$, respectively. Our attack appears to be valid when compared to the exhaustive search for the secret key because $2^{111.16} < 2^{128}$, but in fact, the attack is invalid in terms of the designer's security claim because $\mathsf{TD} = 2^{111.16} \times 2^{34.26} = 2^{145.42} > 2^{126}$. Also, our attack cannot outperform the existing best one [27] since its performance is $\mathsf{TD} = 2^{64.92} \times 2^{58.38} = 2^{123.30} < 2^{145.42}$. We note that our attack considers the outer whitening key, whereas the Liu et al.'s attack [27] did not consider it.

To summarize, our 11-round attack cannot outperform the designer's security claim, but we provide the attack that outperforms the standard security notion (i.e., security against the exhaustive search for the secret key). In addition, our work provides the first key recovery attack on the 11-round QARMA64 under the RT setting with the outer whitening key.

## 5.3 Truncated Differential Attacks on Reduced-Round QARMA128

In this subsection, we provide truncated differential attacks on reduced-round QARMA128, especially under the RT setting. As with the case of QARMA64, under the SK setting, our attack was invalid more than nine rounds, whereas existing studies [38, 40] reported that key recovery attacks were valid up to 10 and 11 rounds[9].

Under the RT setting, the noticeable studies [26, 27] reported that key recovery attacks were valid up to 11 and 12 rounds. Li et al. [26] proposed a tweak difference invariant bias attack on 11-round QARMA128 with the time, data, and memory complexities of $2^{126.1}$, $2^{126.1}$, and $2^{71.0}$, respectively. Liu et al. [27] presented a related-tweak impossible differential attack on 11-round QARMA128

[9] The attacks in [40] did not consider the outer whitening key. Besides, the complexity of the attacks in [38] beyond the designer's security claims.

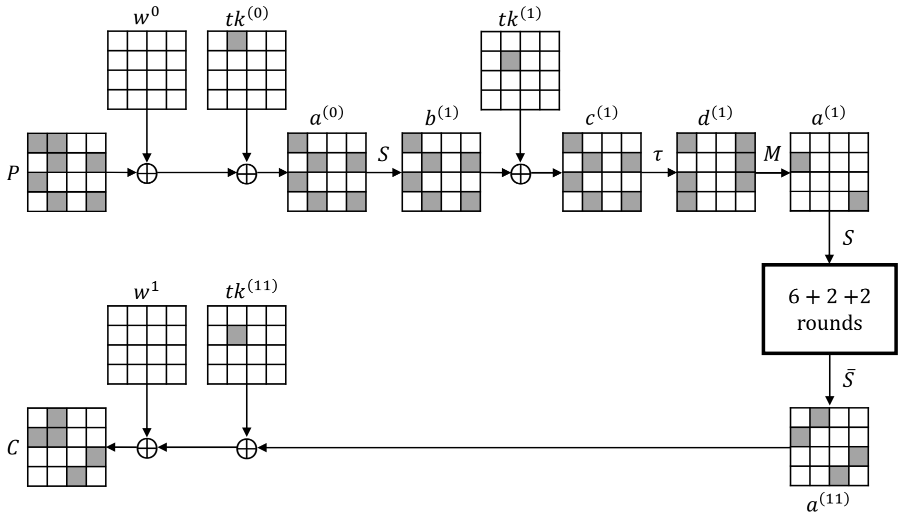


Fig. 5: Propagation of active cells for the 11-round attack on QARMA128 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$).

with the time, data, and memory complexities of $2^{137.0}$, $2^{111.38}$, and $2^{248.38}$, respectively. Moreover, Liu et al. [27] presented a related-tweak meet-in-the-middle attack on 12-round QARMA128 with the time, data, and memory complexities of $2^{156.06}$, $2^{88.00}$, and $2^{244.06}$, respectively. We note that the Liu et al.'s related-tweak impossible differential attack did not consider the outer whitening key. From these reports, our targets are 11 or more rounds of QARMA128 under the RT setting, and our attacks do consider the outer whitening key.

We first focused on the use of the 9-round truncated differential listed in Table 8, but we were faced with the fact that it would be difficult to validate the 11-round attack with this truncated differential. For this reason, we provide here key recovery attacks on the 11-, 12-, and 13-round QARMA128 under the RT setting with the 10-round truncated differential listed in Table 8. We can only append at most three key-recovery rounds to the 10-round truncated differential distinguisher; thus, it must be mentioned that the feasibility of our attacks is up to 13 rounds.

**11-round Attack on QARMA128 under RT Setting.** To launch our 11-round attack, we append only one key-recovery round before the 10-round truncated differential distinguisher. Fig. 5 shows the propagation of active cells for the 11-round attack on QARMA128 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$). We note that $a^{(1)}$, $a^{(11)}$, and $tk^{(2)}$ represent the target input, output, and tweak differentials, respectively. Our attack is based on the following two phases: data collection and key recovery phases.

*Data Collection Phase.* From Fig. 5, the plaintext $P_{\{0,1,5,7,8,13,15\}}$ and the tweak $T^{(0)}_1$ are always active. In addition, $\Delta a^{(0)}_1 = \Delta P_1 \oplus \Delta T^{(0)}_1 = 0$ (i.e., $\Delta P_1 = \Delta T^{(0)}_1$) must hold. Under these conditions, the details of the data collection phase are described as follows.

**Step 1:** We prepare $\mathcal{S}$ structures. In each structure, there are $2^{56}$ plaintext-tweak pairs such that the plaintext $P$ and the tweak $T^{(0)}$ satisfying the above conditions traverse all possible values while the remaining cells are fixed to some random constants. Each structure leads to approximately $(2^{56})^2/2 = 2^{111}$ differential pairs.

**Step 2:** For each structure, we simulate the encryption oracle to get the ciphertexts $C$ from $(P, T^{(0)})$ pairs. Then, we insert every ciphertext with its corresponding plaintext-tweak pair $(P, T^{(0)}, C)$ into a hash table $\mathbb{H}$ at index $C_{\{0,2,3,5,6,7,8,9,10,12,13,15\}} \| tk^{(11)}_5$. Thus, each pair from the same index of $\mathbb{H}$ satisfies the differential pattern of $\Delta C_{\{0,2,3,6,7,8,9,10,12,13,15\}} = 0$. In addition, $\Delta a^{(11)}_5 = \Delta C_5 \oplus \Delta tk^{(11)}_5 = 0$ (i.e., $\Delta C_5 = \Delta tk^{(11)}_5$) must hold. All pairs violating its differential pattern are discarded without further processing because these pairs have no hope to comply with the target truncated differential. This step reduces the differential pair used in the next phase from $2^{111} \times \mathcal{S}$ to $2^{15} \times \mathcal{S}$. Here, we obtain $N = 2^{15} \times \mathcal{S}$ pairs that match the output truncated differential of the 10-round truncated distinguisher.

*Key Recovery Phase.* Hereafter, we denote $wk^0 = w^0 \oplus k^0$ and $wk^1 = w^1 \oplus k^0$. In the key recovery phase, we try to recover all bits of $wk^0$ and $wk^1$. Once these bits are recovered, all of the outer whitening and core keys (i.e., $w^0$, $w^1$, $k^0$, and $k^1$) can be naturally obtained based on the key specialization of QARMA (see Appendix A). The details of the key recovery phase are described as follows.

**Step 1:** For each possible 24-bit value of $wk^0_{\{0,5,15\}}$, we compute $\Delta a^{(1)}_{\{0,4,8,12\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{0,8,12\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{-9} \times \mathcal{S}$.

**Step 2:** For each possible 24-bit value of $wk^0_{\{7,8,13\}}$, we compute $\Delta a^{(1)}_{\{3,7,11,15\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{3,7,11\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{-33} \times \mathcal{S}$. $\Delta a^{(1)}_{\{1,2,5,6,9,10,13,14\}} = 0$ must be derived from the collected data; thus, the key guessing part is completed here.

**Step 3:** For the 48-bit key guesses in total, we prepare counters to confirm the number of right pairs that validate the differential pairs of the 10-round truncated differential distinguisher. The number of right pairs follows a binomial

Table 10: Detailed computation of the time complexity for the 11-round attack.

| Step | Guessed key bits | Time complexity | # of remaining pairs | Note |
|---|---|---|---|---|
| Step 1 | $wk^0_{\{0,5,15\}}$ | $2^{15} \times 2^{24} \times \frac{1}{11} \times \mathcal{S} \approx 2^{35.54} \times \mathcal{S}$ | $2^{15} \times 2^{-24} \times \mathcal{S} = 2^{-9} \times \mathcal{S}$ | 1-round enc |
| Step 2 | $wk^0_{\{7,8,13\}}$ | $2^{-9} \times 2^{48} \times \frac{1}{11} \times \mathcal{S} \approx 2^{35.54} \times \mathcal{S}$ | $2^{-9} \times 2^{-24} \times \mathcal{S} = 2^{-33} \times \mathcal{S}$ | 1-round enc |
| Step 4 | remaining bits | $2^{256} \times \beta \times (1 - 2^{-128})$ | - | 11-round enc |

distribution with parameters $(N, p_0 = 2^{-80.431})$ in the case of the good key and $(N, p_1 = 2^{-112})$ otherwise.

**Step 4:** We fix the threshold as $\Upsilon$, and the key guess will be accepted as a candidate if the counter of right pairs is no less than $\Upsilon$. For all surviving candidates for the 48-bit key, we exhaustively search for the remaining key bits with at most two plaintext-ciphertext pairs.

*Complexity Analysis.* We use the same method for the complexity analysis as described in Sect. 5.2, but the non-detection error probability $\alpha$ cannot be calculated correctly using Equation (3). This is likely due to the very large value of $N$ and the very small value of $p_0$. For this reason, to correctly calculate $\alpha$, we use the well-known de Moivre–Laplace theorem to approximate the binomial distribution to the normal distribution, and then directly exploit the probability density function of the normal distribution. Table 10 lists the detailed computation of the time complexity for the 11-round attack. When we set the threshold as $\Upsilon = 5$ and the number of structures as $\mathcal{S} = 2^{68.05}$, we derive the success probability of $P_S = 80.3\%$ and the false alarm error probability of $\beta = 2^{-158.84}$. To summarize, the 11-round attack on QARMA128 under the RT setting is feasible with the time, data, and memory complexities of $2^{104.60}$, $2^{124.05}$, and $2^{48.00}$, respectively.

*Discussion.* Regarding the designer's security claim, the multiplication of time and data complexities (TD) for QARMA128 should be less than $2^{256-\epsilon}$ for a small $\epsilon$ (e.g., $\epsilon = 2$). From this point of view, our 11-round attack is valid because $\mathsf{TD} = 2^{104.60} \times 2^{124.05} = 2^{228.65} < 2^{254}$. In addition, our 11-round attack outperforms the existing best one with the outer whitening key [26] since its performance is $\mathsf{TD} = 2^{126.1} \times 2^{126.1} = 2^{252.2} > 2^{228.65}$. Therefore, we provide the best key recovery attack on the 11-round QARMA128.

**12/13-round Attack on QARMA128 under RT Setting.** Our 12- and 13-round attacks can be launched in a similar way as described in Sect. 5.3. We provide the details of the attacks in Appendices F and G.

The 12-round attack on QARMA128 under the RT setting is feasible with the time, data, and memory complexities of $2^{154.53}$, $2^{108.52}$, and $2^{144.00}$, respectively. Our 12-round attack appears to be valid when compared to the exhaustive search for the secret key because $2^{154.53} < 2^{256}$, but in fact, the attack is invalid in terms of the designer's security claim because $\mathsf{TD} = 2^{154.53} \times 2^{108.52} = 2^{263.05} > 2^{254}$.

Also, our 12-round attack cannot outperform the existing best one [27] since its performance is $\mathsf{TD} = 2^{156.06} \times 2^{88.00} = 2^{244.06} < 2^{263.05}$. In contrast, the 13-round attack on QARMA128 under the RT setting is feasible with the time, data, and memory complexities of $2^{238.02}$, $2^{106.63}$, and $2^{211.00}$, respectively. Our 13-round attack is also invalid in terms of the designer's security claim because $\mathsf{TD} = 2^{238.02} \times 2^{106.63} = 2^{344.65} > 2^{254}$. We note that this is the first key recovery attack on the 13-round QARMA128 under the RT setting.

To summarize, our 12- and 13-round attacks cannot outperform the designer's security claim, but we provide the attack that outperforms the standard security notion (i.e., security against the exhaustive search for the secret key). In addition, our work provides the first key recovery attack on the 13-round QARMA128 under the RT setting with the outer whitening key.

## 6 Difficulty of Applying the Key-Recovery Attack on PRINCE

In this section, we examine the key-recovery attack on PRINCE and PRINCEv2 using the (truncated) differentials discovered by our method. Similarly to QARMA, the security claim of PRINCE relies on the concept of the tradeoff security, i.e, the attacker is restricted to exploiting the time/data complexity of up to $2^{(time \times data)} < 2^{126}$. For PRINCEv2, the designers claim its security to be below $< 2^{112}$ and $< 2^{47}$ of time and data complexities, respectively.

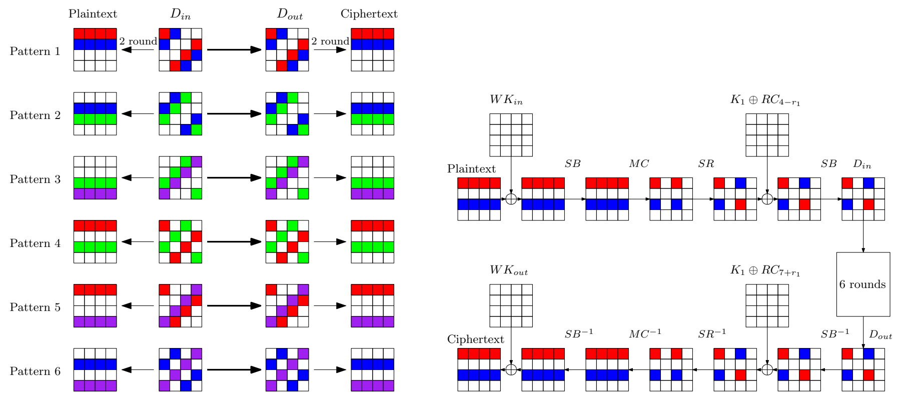
### 6.1 Key-Recovery Attack on PRINCE

As demonstrated in the multiple differential attacks proposed by Canteaut et al. [14], we can append the key-recovery rounds up to 2 rounds to both encryption and decryption sides by searching differentials started from the output of $MC$ to the input of $SB^{-1}$. To improve the known best attack, we adopt the following two approaches:

**Approach 1.** We attempt to mount the key-recovery attack on the 6- or 7-round truncated differentials to attack the 10- or 11-round PRINCE with a similar approach as described in Sect. 5.

**Approach 2.** We attempt to enhance the probability of the multiple differentials utilized in the attack proposed by Canteaut et al. to improve their time and data complexities.

*Approach 1.* To explore the differentials for approach 1, we endeavor to construct the truncated differentials with a probability being greater than $2^{-64}$ on the 6- and 7- round PRINCE by Algorithm 1. To minimize the time and data complexities, we place limitations on the truncated differentials: only the same two rows are affected by active cells in the input and output differences after appending the key-recovery rounds. Fig. 6a displays the six patterns conforming to these restrictions, and we search for such differentials with the highest



(a) Differentials for approach 1. (b) Differentials for approach 2.

Fig. 6: Key-recovery-friendly (truncated) differentials for PRINCE and PRINCEv2. In Fig. 6b, $WK_{in/out}$, $K_1$, and $RC_i$ denote the whitening keys in the input/output, round key, and round constant, respectively. The index of the cells follows that of QARMA.

probability. As a result, we successfully discovered several truncated differentials that fit these patterns with a probability greater than $2^{-64}$ in both 6 and 7 rounds. For 6 rounds, we identified a truncated differential with a probability of $2^{-52}$, with four active cells in both $D_{in}$ and $D_{out}$ in the pattern 1, 2, and 5. We here take pattern 1 as an example and attempt to apply the key-recovery attack to the 10-round PRINCE. We first prepare $\mathcal{S}$ structures, each of which contains $2^{32}$ plaintext with the same selection manner as in Sect. 5. Since the corresponding ciphertext differences in row 1 and 2 must hold 0, we obtain the $2^{63} \times 2^{-32} \times \mathcal{S} = 2^{31} \times \mathcal{S}$ pairs that possibly match the truncated differential. In the key recovery phase from the encryption side, $2^{31} \times \mathcal{S}$ pairs are reduced to $2^{15} \times \mathcal{S}$ by a 16-bit filter because the truncated differential that we found has four active cells in $D_{in}$. As we can construct at most $\mathcal{S} \leq 2^{32}$ structures, this attack does not work due to $2^{15} \times 2^{32} \leq 2^{52}$. The same situation occurs with the 7-round truncated differential because its probability[10] is much lower than that of the 6-round truncated differential. Therefore, we conclude that improving the known best attack to PRINCE by the truncated differential attack is hard.

*Approach 2.* We search for the differentials illustrated in Fig. 6b by Algorithm 1, which is the ones used in the known best key recovery attack [14]. As a result, we discover four differentials with the probability $2^{-57.678}$, as well as some differentials with the probability $2^{-58.607}$. However, since Canteaut et al. employ twelve differentials with a probability of $2^{-56.42}$ in their multiple differential attacks, our differentials do not allow for any improvements to their attack. This

[10] We found the 7-round truncated differential with the probability $2^{-62.985}$ in the pattern 1.

gap in the probabilities comes from the difference in the method of construction differentials. Canteaut et al. theoretically construct differentials based on the transition matrix while our method does it by directly gathering multiple differential characteristics with the weight from 62 to 81. We expect that it is possible to further improve the probability of our differentials by taking the wider range of the clustering effect into account. However, it takes more computational cost, and we cannot do it with our computational environment. Therefore, our tool confirms that the existing differentials by Canteaut et al. [14] are the best ones for the purpose of key recovery attacks.

### 6.2 Key-Recovery Attack on PRINCEv2

The known best attack is proposed by the designers which is the 8-round attack by the integral attack. Hence, we aim to carry out a key-recovery attack on the 9-round PRINCEv2. As approach 2 to PRINCEv2 has already been tried by the designers and showed that it did not work, we focus solely on approach 1.

We cannot employ the 6-round truncated differentials discovered in Sect. 6.1 for the key-recovery attack on PRINCEv2 because of the limitation of the data complexity being less than $2^{47}$. Therefore, we search for truncated differentials in 5 rounds under the same limitation as depicted in Fig. 6a. As a result, we found the truncated differentials with the probability $2^{-43.977}$ in pattern 2. However, we found applying them to the key-recovery attack impossible because the same situation, as in the case of PRINCE, occurs if a more than 4-bit filter is applied in the key-recovery phase. To avoid this issue, we construct truncated differentials that the pairs of the plaintext are unfiltered in the key-recovery phase, but this leads to the probability of the wrong-key suggestion $p_1$ being considerably greater than the probability of the right-key suggestion $p_0$, thus rendering the recovery of the correct key impossible. Thus, we conclude that it is difficult to mount the key-recovery attack on the 9-round PRINCEv2.

## 7 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 for 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.
– We specify the conditions for finding the best differentials for performing the key recovery attacks and conduct experiments using our framework. As a result, our attacks cannot outperform the previous best attack for almost all variants of the target ciphers, but we demonstrated that our framework is valid in evaluating the tight security of the symmetric-key primitives against key-recovery attacks based on straightforward differential cryptanalysis.

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. Blondeau, C., Gérard, B., Tillich, J.: Accurate estimates of the data complexity and success probability for various cryptanalyses. Des. Codes Cryptogr. **59**(1-3), 3–34 (2011)
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. 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)
13. Brayton, R.K., Hachtel, G.D., McMullen, C.T., Sangiovanni-Vincentelli, A.L.: Logic Minimization Algorithms for VLSI Synthesis, The Kluwer International Series in Engineering and Computer Science, vol. 2. Springer (1984)
14. 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)
15. Cook, S.A.: The complexity of theorem-proving procedures. In: STOC. pp. 151–158. ACM (1971)
16. Ding, Y., Zhao, J., Li, L., Yu, H.: Impossible differential analysis on round-reduced PRINCE. J. Inf. Sci. Eng. **33**(4), 1041–1053 (2017)
17. Dobraunig, C., Eichlseder, M., Kales, D., Mendel, F.: Practical key-recovery attack on MANTIS5. IACR Trans. Symmetric Cryptol. **2016**(2), 248–260 (2016)
18. 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)
19. 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)
20. Guo, H., Sun, S., Shi, D., Sun, L., Sun, Y., Hu, L., Wang, M.: Differential attacks on CRAFT exploiting the involutory s-boxes and tweak additions. IACR Trans. Symmetric Cryptol. **2020**(3), 119–151 (2020)
21. Hamadi, Y., Jabbour, S., Sais, L.: Manysat: a parallel SAT solver. J. Satisf. Boolean Model. Comput. **6**(4), 245–262 (2009)
22. 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)
23. 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)
24. 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)
25. 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 proces-

sor architectures. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(4), 510–545 (2021)
26. 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)
27. 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)
28. Liu, Y., Wang, Q., Rijmen, V.: Automatic search of linear trails in ARX with applications to SPECK and chaskey. In: ACNS. Lecture Notes in Computer Science, vol. 9696, pp. 485–499. Springer (2016)
29. 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)
30. McCluskey, E.J.: Minimization of Boolean functions. The Bell System Technical Journal **35**(6), 1417–1444 (1956)
31. Quine, W.V.: The problem of simplifying truth functions. The American mathematical monthly **59**(8), 521–531 (1952)
32. Quine, W.V.: A way to simplify truth functions. The American mathematical monthly **62**(9), 627–631 (1955)
33. 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)
34. Sun, L., Wang, W., Wang, M.: More accurate differential properties of LED64 and midori64. IACR Trans. Symmetric Cryptol. **2018**(3), 93–123 (2018)
35. 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)
36. Wang, S., Feng, D., Hu, B., Guan, J., Shi, T., Zhang, K.: The simplest SAT model of combining matsui's bounding conditions with sequential encoding method. IACR Cryptol. ePrint Arch. p. 626 (2022)
37. Warners, J.P.: A linear-time transformation of linear inequalities into conjunctive normal form. Inf. Process. Lett. **68**(2), 63–69 (1998)
38. Yang, D., Qi, W., Chen, H.: Impossible differential attack on QARMA family of block ciphers. IACR Cryptol. ePrint Arch. p. 334 (2018)
39. 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)
40. Zong, R., Dong, X.: Meet-in-the-middle attack on QARMA block cipher. IACR Cryptol. ePrint Arch. p. 1160 (2016)
41. 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 [10] and PRINCEv2[11] [12] are two families of a block cipher with a 64-bit block and 128-bit key. Both ciphers have the

[11] 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 11: 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 12: 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 |

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 11. $SR$ is the shift row operation that applies the same permutation used in AES as shown in Table 12. $MC$ is the MixColumns operation composed from the following four basic $4 \times 4$ 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 $16 \times 16$ 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 $64 \times 64$ matrix $M'$ applied in $MC$ as follows:

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

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 [10, 12].

**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 $4 \times 4$ 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 $4 \times 4$ 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 $\mathsf{IS}$.

`ShuffleCells`. $(\tau(\mathsf{IS}))_i = s_{\tau(i)}$ for $0 \leq i \leq 15$, where $\tau$ 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 $\mathsf{IS}$ 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 $\rho^i$ 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 $\sigma$ 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:

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

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 $\omega$ updates the tweak cells with indexes 0, 1, 3, 4, 8, 11, and 13. For QARMA64, $\omega$ 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 Modeling for Each Operation

We provide a detailed description of how to construct a SAT model for each operation by referring to the existing method proposed by Sun et al. [34,35] and Wang et al. [36].

### B.1 Modeling for an XOR [34, 35]

Let $(a_0, a_1, \dots, a_{i-1})$ and $b$ be the input and output differences of an XOR operation with $i$ inputs, respectively. Hence, $a_0 \oplus a_1 \oplus \cdots \oplus a_{i-1} = b$. Let $X$ be the set $\{(x_0, x_1, \dots, x_i) \in \mathbb{F}_2^{i+1} \mid (x_0 \oplus x_1 \oplus \cdots \oplus x_i) = 1\}$. The following variables and clauses are sufficient to express the differential propagation for an XOR operation with $i$ inputs:

$$\mathcal{M}_{var} \leftarrow (a_0, a_1, \dots, a_{i-1}, b, x_0, x_1, \dots, x_i),$$
$$\mathcal{M}_{cla.xor} \leftarrow (a_0 \oplus x_0) \vee (a_1 \oplus x_1) \vee \cdots \vee (a_{i-1} \oplus x_{i-1}) \vee (b \oplus x_i) \in X.$$

### B.2 Modeling for a Copy [34, 35]

Let $a$ and $(b_0, b_1, \dots, b_{i-1})$ be the input and output differences of a copy operation, respectively. Hence, $a = b_0 = b_1 = \cdots = b_{i-1}$. The following variables and clauses are sufficient to express the differential propagation for a copy operation:

$$\mathcal{M}_{var} \leftarrow (a, b_0, b_1, \dots b_{i-1}),$$
$$\mathcal{M}_{cla.copy} \leftarrow \begin{cases} a \vee \overline{b_j} & \textit{for } 0 \le j \le i-1, \\ \overline{a} \vee b_j & \textit{for } 0 \le j \le i-1. \end{cases}$$

### B.3 Modeling for a Matrix

Let $(a_0, a_1, \dots, a_{i-1})$ and $(b_0, b_1, \dots, b_{i-1})$ be the input and output differences of an $i \times i$ matrix, respectively. As a matrix can be decomposed to XOR and copy operations, the differential propagation for an $i \times i$ matrix can be expressed by $\mathcal{M}_{cla.xor}$ and $\mathcal{M}_{cla.copy}$ with some auxiliary Boolean variables. Let $(x_0, x_1, \dots, x_{j-1})$ be the auxiliary Boolean variables, where $j$ is equal to the total number of the input terms in all equations decomposed from an $i \times i$ matrix. The following variables and clauses are sufficient to express the differential propagation for an $i \times i$ matrix:

$$\mathcal{M}_{var} \leftarrow (a_0, a_1, \dots, a_{i-1}, b_0, b_1, \dots b_{i-1}, x_0, x_1, \dots, x_{j-1}),$$
$$\mathcal{M}_{cla.matrix} \leftarrow \begin{cases} \mathcal{M}_{cla.xor} & \textit{for all XORs to be decomposed from a matrix,} \\ \mathcal{M}_{cla.copy} & \textit{for all copies to be decomposed from a matrix.} \end{cases}$$

For a better understanding, we take the following $3 \times 3$ matrix as an example:

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}.$$

This $3 \times 3$ matrix can be decomposed as follows:

$$\begin{aligned} a_1 \oplus a_2 &= b_0, \\ a_0 \oplus a_2 &= b_1, \\ a_0 \oplus a_1 &= b_2. \end{aligned}$$

Because these are six input terms in total, we prepare six auxiliary Boolean variables $(x_{i,0}, x_{i,1})$ for $0 \leq i \leq 2$. Then, we can construct the model for this matrix as follows:

$$\mathcal{M}_{var} \leftarrow (a_0, a_1, a_2, b_0, b_1, b_2, x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}),$$

$$\mathcal{M}_{cla.xor} \leftarrow \begin{cases} \overline{x_{1,0}} \vee \overline{x_{2,0}} \vee \overline{b_0}, \\ \overline{x_{1,0}} \vee x_{2,0} \vee b_0, \\ x_{1,0} \vee \overline{x_{2,0}} \vee b_0, \\ x_{1,0} \vee x_{2,0} \vee \overline{b_0}, \end{cases} \begin{cases} \overline{x_{0,0}} \vee \overline{x_{2,1}} \vee \overline{b_1}, \\ \overline{x_{0,0}} \vee x_{2,1} \vee b_1, \\ x_{0,0} \vee \overline{x_{2,1}} \vee b_1, \\ x_{0,0} \vee x_{2,1} \vee \overline{b_1}, \end{cases} \begin{cases} \overline{x_{0,1}} \vee \overline{x_{1,1}} \vee \overline{b_2}, \\ \overline{x_{0,1}} \vee x_{1,1} \vee b_2, \\ x_{0,1} \vee \overline{x_{1,1}} \vee b_2, \\ x_{0,1} \vee x_{1,1} \vee \overline{b_2}, \end{cases}$$

$$\mathcal{M}_{cla.copy} \leftarrow \begin{cases} a_i \vee \overline{x_{i,j}} & 0 \leq i \leq 2,\ 0 \leq j \leq 1, \\ \overline{a_i} \vee x_{i,j} & 0 \leq i \leq 2,\ 0 \leq j \leq 1, \end{cases}$$

$$\mathcal{M}_{cla.matrix} \leftarrow \begin{cases} \mathcal{M}_{cla.copy}, \\ \mathcal{M}_{cla.xor}. \end{cases}$$

Apart from the above method, another method exists to model a matrix more efficiently. Instead of introducing the auxiliary Boolean variable, we can directly place each input variable to each equation which is decomposed from a $3 \times 3$ matrix, as follows:

$$\mathcal{M}_{var} \leftarrow (a_0, a_1, a_2, b_0, b_1, b_2),$$

$$\mathcal{M}_{cla.xor} \leftarrow \begin{cases} \overline{a_1} \vee \overline{a_2} \vee \overline{b_0}, \\ \overline{a_1} \vee a_2 \vee b_0, \\ a_1 \vee \overline{a_2} \vee b_0, \\ a_1 \vee a_2 \vee \overline{b_0}, \end{cases} \begin{cases} \overline{a_0} \vee \overline{a_2} \vee \overline{b_1}, \\ \overline{a_0} \vee a_2 \vee b_1, \\ a_0 \vee \overline{a_2} \vee b_1, \\ a_0 \vee a_2 \vee \overline{b_1}, \end{cases} \begin{cases} \overline{a_0} \vee \overline{a_1} \vee \overline{b_2}, \\ \overline{a_0} \vee a_1 \vee b_2, \\ a_0 \vee \overline{a_1} \vee b_2, \\ a_0 \vee a_1 \vee \overline{b_2}, \end{cases}$$

$$\mathcal{M}_{cla.matrix} \leftarrow \mathcal{M}_{cla.xor}.$$

The above model does not contain a model for a copy operation by removing the auxiliary Boolean variable. In fact, a model for a copy operation must not be used through a whole SAT model, because all copy operations can be realized by inputting the proper Boolean variables into the suitable clauses. Because keeping the number of Boolean variables and clauses in a whole SAT model as few as possible contributes to reducing runtime to solve a SAT problem, we do not use a model for a copy operation in our modeling.

### B.4 Modeling for an S-box [34–36]

Let $\boldsymbol{a} = (a_0, a_1, \ldots, a_{i-1})$ and $\boldsymbol{b} = (b_0, b_1, \ldots, b_{i-1})$ be the input and output differences of an $i$-bit S-box, respectively. Additionally, we must introduce additional Boolean variables $\boldsymbol{p} = (p_0, p_1, \ldots, p_{j-1})$, where $j$ is the maximum weight of the differential propagation through an $i$-bit S-box and $p_q \in \{0, 1\}$ for $1 \leq q \leq j-1$, to count the differential probability as an S-box is a probabilistic operation. With these Boolean variables, we construct the following Boolean formula:

$$f(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{p}) = \begin{cases} 1 & \textit{if } \Pr(\boldsymbol{a} \rightarrow \boldsymbol{b}) = 2^{-\sum_{q=0}^{j-1} p_q}, \\ 0 & \textit{otherwise.} \end{cases} \tag{4}$$

Notably, these are multiple choices of $\boldsymbol{p}$ for each possible propagation $\boldsymbol{a} \rightarrow \boldsymbol{b}$. For example, we have three choices $(p_0, p_1, p_2) \in \{(0, 1, 1), (1, 1, 0), (1, 0, 1)\}$ when $2^{-\sum_{q=0}^{2} p_q} = 2^{-2}$. In Eq. (4), we chose only one arbitrary choice, namely $f(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{p})$ can be one if and only if $\boldsymbol{p}$ is this one arbitrary choice. Then, we extract a set $A$, which contains all vectors satisfying $f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = 0$ as follows:

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

where $\boldsymbol{x} = (x_0, x_1, \ldots, x_{i-1})$, $\boldsymbol{y} = (y_0, y_1, \ldots, y_{i-1})$, and $\boldsymbol{z} = (z_0, z_1, \ldots, z_{j-1})$. Because $A$ is a set of invalid patterns in a model of an S-box, we ban these patterns by the following clauses:

$$\bigvee_{c=0}^{i-1} (a_c \oplus x_c) \vee \bigvee_{d=0}^{i-1} (b_d \oplus y_d) \vee \bigvee_{e=0}^{j-1} (p_e \oplus z_e) = 1, \ (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in A. \tag{5}$$

The remaining vectors that are identical to $\overline{A}$ are a set of valid patterns. Therefore, these clauses extract the differential propagation with corresponding weight over an $i$-bit S-box. Note that the solution space of $|A|$ clauses about $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{p})$ in Eq. (5) is identical to that of the following Boolean function:

$$g(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{p}) = \bigwedge_{\eta=0}^{|A|-1} \left( \bigvee_{c=0}^{i-1} (a_c \oplus x_c^\eta) \vee \bigvee_{d=0}^{i-1} (b_d \oplus y_d^\eta) \vee \bigvee_{e=0}^{j-1} (p_e \oplus z_e^\eta) \right) = 1.$$

This Boolean function is equivalent to

$$g(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{p}) = \bigwedge_{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathbb{F}_2^{2i+j}} \left( g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \vee \bigvee_{c=0}^{i-1} (a_c \oplus x_c^\eta) \vee \bigvee_{d=0}^{i-1} (b_d \oplus y_d^\eta) \vee \bigvee_{e=0}^{j-1} (p_e \oplus z_e^\eta) \right).$$

This equation is called the *product-of-sum* of $g$. The issue of reducing the number of clauses in $g$ is turned into the issue of simplifying the product-of-sum representation of the Boolean function. Owing to previous works [28, 34, 35], we know that this can be solved by the Quine-McCluskey algorithm [30–32] and Espresso algorithm [13]. When this problem is relatively small, we can solve it by software, such as *Logic Friday*[12], although it is NP-complex. In this study, we

[12] `http://www.sontrak.com/`

construct a model for an S-box by Logic Friday to reduce the number of clauses in $g$. Therefore, the following variables and clauses are sufficient to express the differential propagation with corresponding weight over an $i$-bit S-box:

$$\begin{aligned}
&\mathcal{M}_{var} \leftarrow (a_0, a_1, \dots, a_{i-1}, b_0, b_1, \dots, b_{i-1}, p_0, p_1, \dots, p_{j-1}),\\
&\mathcal{M}_{cla.sbox} \leftarrow min\left(g(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{p})\right).
\end{aligned}$$

## B.5 Modeling for a Sequential Encoding Counter [33]

Let $(p_0^{(r)}, p_1^{(r)}, \dots, p_{i-1}^{(r)})$ be the Boolean variables to explain the weight of an S-box, where $r$ and $i$ denote the number of rounds and the total number of such variables in a single round, respectively. For a better understanding, we redefine them as the sequential Boolean variables like $(p_0, p_1, \dots, p_{r\cdot i-1})$. Additionally, we introduce the auxiliary Boolean variables $(s_{0,k}, s_{1,k}, \dots, s_{r\cdot i-1,k})$ for $0 \le k \le B-1$, where $B$ denotes a certain bound. The following variables and clauses are sufficient to express the sequential encoding counter $\sum_{j=0}^{r\cdot i-1} p_j \le B$:

$$\mathcal{M}_{var} \leftarrow (p_0, p_1, \dots, p_{r\cdot i-1}, s_{0,0}, s_{1,0}, \dots, s_{r\cdot i-1,B-1}),$$

$$\mathcal{M}_{cla.sec(B)} \leftarrow \begin{cases}
\overline{p_0} \vee s_{0,0},\\
\overline{s_{0,m}},\ 1 \le m \le B-1,\\
\left.\begin{array}{l}
\overline{p_n} \vee s_{n,0},\\
\overline{s_{n-1,0}} \vee s_{n,0},\\
\left.\begin{array}{l}
\overline{p_n} \vee \overline{s_{n-1,l-1}} \vee s_{n,l},\\
\overline{s_{n-1,l}} \vee s_{n,l},
\end{array}\right\} 1 \le l \le B-1\\
\overline{p_n} \vee \overline{s_{n-1,B-1}},
\end{array}\right\} 1 \le n \le r\cdot i-2\\
\overline{p_{r\cdot i-1}} \vee \overline{s_{r\cdot i-2,B-1}}.
\end{cases}$$

## B.6 Modeling for Constraint of Input Differences

Let $(a_0, a_1, \dots, a_{i-1})$ be Boolean variables to express the differences at the first operation in a cipher (basically, $i$ matches the block size of a cipher). The following clauses are sufficient to avoid trivial differential characteristics, as all input differences are zero at the same time:

$$\mathcal{M}_{cla.input} \leftarrow a_0 \vee a_1 \vee \dots \vee a_{i-1}.$$

## B.7 Modeling for Evaluation of a Clustering Effect

Let $(a_{j,0}, a_{j,1}, \dots, a_{j,i-1})$ 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 $\boldsymbol{C} = (\boldsymbol{c_0}, \boldsymbol{c_1}, \dots, \boldsymbol{c_r})$, where $\boldsymbol{c_m} = (c_{m,0}, c_{m,1}, \dots, c_{m,i-1})$, we can fix the input and output differences to $\boldsymbol{c_0}$ and $\boldsymbol{c_r}$, respectively, by the following clauses:

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

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

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

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

## C Basic Algorithms

We describe the basic algorithms employed to construct our framework with the notations in Sect. 2.2. Sun et al. [34,35] already briefly presented the algorithm to evaluate the clustering effect of 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 $W_{min}$.

**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 `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

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

```
1  Function $\mathtt{SAT}_{\mathtt{diff.min}}(T, r)$
2  begin
3      $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \mathtt{SET}_{\mathtt{model}}(T, r)$
4      while $\mathtt{SAT}_{\mathtt{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var})$ = ("UNSAT", $\emptyset$) do
5          $T \leftarrow T + 1$
6          $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \mathtt{SET}_{\mathtt{model}}(T, r)$
7      $W_{min} \leftarrow T$
8      return $(\boldsymbol{C_r}, W_{min})$

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

14 Function $\mathtt{SAT}_{\mathtt{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var})$
15 begin
16     if $\mathtt{Solver}(\mathcal{M}_{SAT}, \mathcal{M}_{var})$ = "SAT" then
17         for $i = 0$ to $r$ do
18             $\boldsymbol{c_i} \leftarrow \boldsymbol{v_i}$
19         return ("SAT", $\boldsymbol{C_r}$)
20     else
21         return ("UNSAT", $\emptyset$)
```

$\boldsymbol{C_r}$, whereas $\mathtt{SAT}_{\mathtt{diff.char}}()$ 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 $\mathtt{SAT}_{\mathtt{diff.min}}()$, the clustering effect of $\boldsymbol{C_r}$ is evaluated by Algorithm 4, namely, this algorithm find the differential characteristics 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 $\sum_{i=T_{low}}^{T_{upp}} N_{i-T_{low}} \cdot 2^{-i}$.

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

```
1  Function $\mathtt{SAT}_{\mathtt{diff.clust}}(T_{low}, T_{upp}, r, (\boldsymbol{c_0}, \boldsymbol{c_r}))$
2  begin
3      $\boldsymbol{N_{T_{upp}-T_{low}+1}} \leftarrow (N_0, N_1, \ldots, N_{T_{upp}-T_{low}})$
4      for $i = T_{low}$ to $T_{upp}$ do
5          $(\mathcal{M}_{SAT}, \mathcal{M}_{var}) \leftarrow \mathtt{SET}_{\mathtt{model}}(i, r)$
6          add $\mathcal{M}_{cla.clust}$ to $\mathcal{M}_{SAT}$
7          add auxiliary Boolean variables of $\mathcal{M}_{cla.\overline{sec}(i)}$ to $\mathcal{M}_{var}$
8          add $\mathcal{M}_{cla.\overline{sec}(i)}$ to $\mathcal{M}_{SAT}$
9          $N_{i-T_{low}} \leftarrow 0$
10         while $\mathtt{SAT}_{\mathtt{diff.char}}(\mathcal{M}_{SAT}, \mathcal{M}_{var})$ = ("SAT", $\boldsymbol{C_r}$) do
11             $N_{i-T_{low}} \leftarrow N_{i-T_{low}} + 1$
12             add $\mathcal{M}_{cla.\overline{clust}}$ to $\mathcal{M}_{SAT}$
13     return $\boldsymbol{N_{T_{upp}-T_{low}+1}}$
```

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 $W$.

## D 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 13. To investigate an effect

**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 $\boldsymbol{D}$
```

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 13 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.

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,

Table 13: 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 |

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

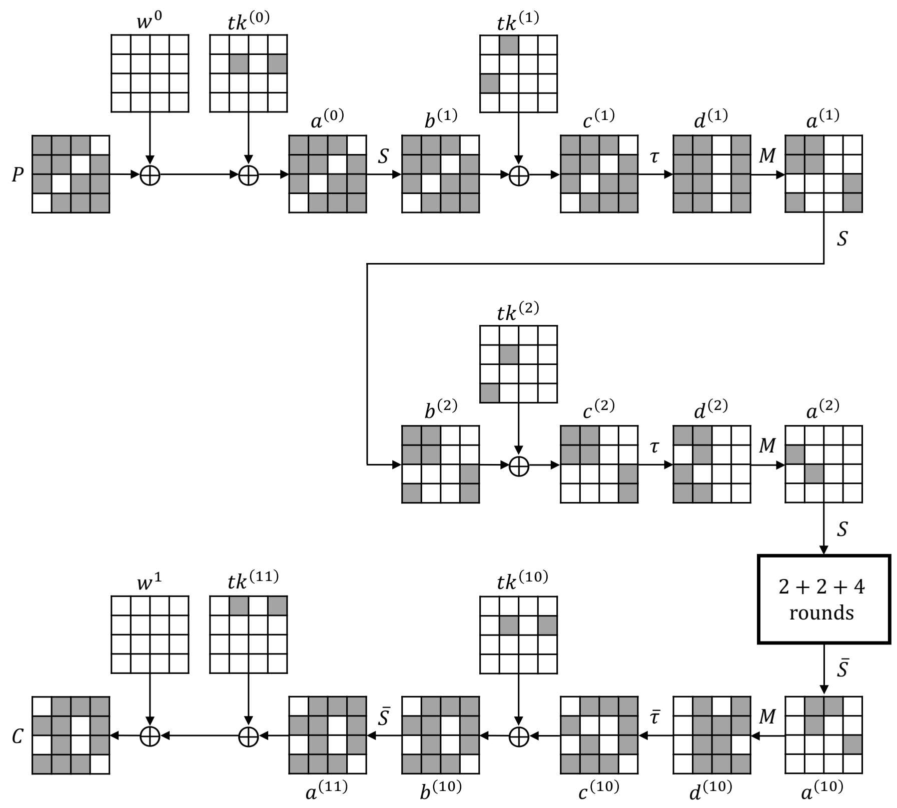


Fig. 7: Propagation of active cells for the 11-round attack on QARMA64 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$).

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.

## E 11-round Attack on QARMA64 under RT Setting

To launch our 11-round attack, we append two and one key-recovery rounds before and after the 8-round truncated differential distinguisher, respectively. Fig. 7 shows the propagation of active cells for the 11-round attack on QARMA64 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$). We note that $a^{(2)}$, $a^{(10)}$, and $tk^{(3)}$ represent the target input, output, and tweak differentials, respectively. Our attack is based on the following two phases: data collection and key recovery phases.

*Data Collection Phase.* From Fig 7, the plaintext $P_{\{0,1,2,4,5,7,8,10,11,13,14,15\}}$ and the tweak $T^{(0)}_{\{5,7\}}$ are always active. Under this condition, the details of the data collection phase are described as follows.

**Step 1:** We prepare $\mathcal{S}$ structures. In each structure, there are $2^{56}$ plaintext-tweak pairs such that the plaintext $P$ and the tweak $T^{(0)}$ satisfying the above condition traverse all possible values while the remaining cells are fixed to some random constants. Each structure leads to approximately $(2^{56})^2/2 = 2^{111}$ differential pairs.

**Step 2:** For each structure, we simulate the encryption oracle to get the ciphertexts $C$ from $(P, T^{(0)})$ pairs. Then, we insert every ciphertext with its corresponding plaintext-tweak pair $(P, T^{(0)}, C)$ into a hash table $\mathbb{H}$ at index $C_{\{0,6,8,10,15\}}$. Thus, each pair from the same index of $\mathbb{H}$ satisfies the differential pattern of $\Delta C_{\{0,6,8,10,15\}} = 0$. All pairs violating its differential pattern are discarded without further processing because these pairs have no hope to comply with the target truncated differential. This step reduces the differential pair used in the next phase from $2^{111} \times \mathcal{S}$ to $2^{91} \times \mathcal{S}$.

*Key Recovery Phase.* Hereafter, we use the following notations. We denote $wk^0 = w^0 \oplus k^0$ and $wk^1 = w^1 \oplus k^0$. When representing the bit position of each cell, we denote it as $s_i^{(r)}[0], \ldots, s_i^{(r)}[3]$, where $i$ represents the cell number, $r$ represents the number of rounds, $s_i^{(r)}[0]$ represents the most significant bit, and $s_i^{(r)}[3]$ represents the least significant bit.

In the key recovery phase, we try to recover all bits of $wk^0$ and $wk^1$. Once these bits are recovered, all of the outer whitening and core keys (i.e., $w^0$, $w^1$, $k^0$, and $k^1$) can be naturally obtained based on the key specialization of QARMA (see Appendix A). The details of the key recovery phase are described as follows.

**Step 1:** For each possible 4-bit value of $wk^1_5$, we compute $\Delta c_5^{(10)}$ from the ciphertexts $C$ and check whether $\Delta c_5^{(10)} = 0$. This step leads to a 4-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{87} \times \mathcal{S}$.

**Step 2:** For each possible 16-bit value of $wk^1_{\{1,4,11,14\}}$, we compute $\Delta a^{(10)}_{\{1,5,9,13\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(10)}_{\{9,13\}} = 0$. This step leads to an 8-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{79} \times \mathcal{S}$.

**Step 3:** For each possible 12-bit value of $wk^1_{\{3,9,12\}}$, we compute $\Delta a^{(10)}_{\{2,6,10,14\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(10)}_{\{6,10,14\}} = 0$. This step leads to a 12-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{67} \times \mathcal{S}$.

**Step 4:** For each possible 12-bit value of $wk^1_{\{2,7,14\}}$, we compute $\Delta a^{(10)}_{\{3,7,11,15\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(10)}_{\{3,7,15\}} = 0$. This step leads to a 12-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{55} \times \mathcal{S}$. Here, we obtain $N = 2^{55} \times \mathcal{S}$ pairs that match the output truncated differential of the 8-round truncated distinguisher.

**Step 5:** For each possible 16-bit value of $wk^0_{\{1,4,11,14\}}$, we compute $\Delta a^{(1)}_{\{1,5,9,13\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{9,13\}} = 0$. This step leads to an 8-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{47} \times \mathcal{S}$.

**Step 6:** For each possible 16-bit value of $wk^0_{\{2,7,8,13\}}$, we compute $\Delta a^{(1)}_{\{3,7,11,15\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{3,7\}} = 0$. This step leads to an 8-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{39} \times \mathcal{S}$.

**Step 7:** For each possible 16-bit value of $wk^0_{\{0,5,10,15\}}$, we compute $\Delta a^{(1)}_{\{0,4,8,12\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{8} = 0$. This step leads to a 4-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{35} \times \mathcal{S}$.

**Step 8:** Hereafter, we guess $k^0$ using the previously guessed key bits. When guessing each possible 5-bit value of $k^0_{\{1,4,7,11,13\}}[0]$, we can uniquely determine the 36-bit value of $k^0_{\{1,2,4,5,7,11,13,14\}}$, $k^0_{\{3,12\}}[0]$, and $k^0_{\{0,10\}}[3]$ in total. To clarify this reason, we consider the case of guessing $k^0_1[0]$ as an example. From the relationship between $wk^0$ and $wk^1$, we can derive $wk^0_1[0] = w^0_1[0] \oplus k^0_1[0]$ and $wk^1_1[0] = w^0_0[3] \oplus k^0_1[0]$. Since we have already guessed $wk^0_{\{0,1,2\}}$ and $wk^1_{\{1,2,3\}}$ (i.e., we have the values of $w^0_1[0] \oplus k^0_1[0]$ and $w^0_0[3] \oplus k^0_1[0]$), once the value of $k^0_1[0]$ is guessed, we can uniquely determine the values of $w^0_1[0]$ and $w^0_0[3]$. Based on these values, we can also uniquely determine the value of $k^0_0[3]$, $k^0_1[1]$, and $w^0_1[1]$ since we have the values of $w^0_0[3] \oplus k^0_0[3]$, $w^0_1[0] \oplus k^0_1[1]$, and $w^0_1[1] \oplus k^0_1[1]$, respectively. By repeating such a procedure, we can uniquely determine the 10-bit value of $k^0_{\{1,2\}}$, $k^0_3[0]$, and $k^0_0[3]$ when guessing the 1-bit value of $k^0_1[0]$. Getting back to track, for each possible 5-bit value of $k^0_{\{1,4,7,11,13\}}[0]$, we compute $\Delta a^{(2)}_{\{6,13\}}$ from the plaintexts $P$ and check whether $\Delta a^{(2)}_{13} = 0$. This step leads to a 4-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{31} \times \mathcal{S}$.

**Step 9:** For each possible 4-bit value of $k^0_8$, we compute $\Delta a^{(2)}_0$ from the plaintexts $P$ and check whether $\Delta a^{(2)}_0 = 0$. This step leads to a 4-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{27} \times \mathcal{S}$.

**Step 10:** For each possible 6-bit value of $k^0_{\{0,10\}}[0,1,2]$, we compute $\Delta c^{(2)}_{12}$ from the plaintexts $P$ and check whether $\Delta c^{(2)}_{12} = 0$. When $\Delta c^{(2)}_{12} = 0$ holds, $\Delta a^{(2)}_{\{2,6,10,14\}} = 0$ also always holds. This step leads to a 4-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{23} \times \mathcal{S}$.

**Step 11:** For each possible 4-bit value of $k^0_{15}$, we compute $\Delta a^{(2)}_{\{1,4,5,8,9,12\}}$ from the plaintexts $P$ and check whether $\Delta a^{(2)}_{\{1,5,8,12\}} = 0$. This step leads to a 16-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^7 \times \mathcal{S}$.

Table 14: Detailed computation of the time complexity for the 11-round attack.

| Step | Guessed key bits | Time complexity | # of remaining pairs | Note |
|---|---|---|---|---|
| Step 1 | $wk_5^1$ | $2^{91} \times 2^4 \times \frac{1}{11} \times \mathcal{S} \approx 2^{91.54} \times \mathcal{S}$ | $2^{91} \times 2^{-4} \times \mathcal{S} = 2^{87} \times \mathcal{S}$ | 1-round dec |
| Step 2 | $wk_{\{1,4,11,14\}}^1$ | $2^{87} \times 2^{20} \times \frac{1}{11} \times \mathcal{S} \approx 2^{103.54} \times \mathcal{S}$ | $2^{87} \times 2^{-8} \times \mathcal{S} = 2^{79} \times \mathcal{S}$ | 1-round dec |
| Step 3 | $wk_{\{3,9,12\}}^1$ | $2^{79} \times 2^{32} \times \frac{1}{11} \times \mathcal{S} \approx 2^{107.54} \times \mathcal{S}$ | $2^{79} \times 2^{-12} \times \mathcal{S} = 2^{67} \times \mathcal{S}$ | 1-round dec |
| Step 4 | $wk_{\{2,7,14\}}^1$ | $2^{67} \times 2^{44} \times \frac{1}{11} \times \mathcal{S} \approx 2^{107.54} \times \mathcal{S}$ | $2^{67} \times 2^{-12} \times \mathcal{S} = 2^{55} \times \mathcal{S}$ | 1-round dec |
| Step 5 | $wk_{\{1,4,11,14\}}^0$ | $2^{55} \times 2^{60} \times \frac{1}{11} \times \mathcal{S} \approx 2^{111.54} \times \mathcal{S}$ | $2^{55} \times 2^{-8} \times \mathcal{S} = 2^{47} \times \mathcal{S}$ | 1-round enc |
| Step 6 | $wk_{\{2,7,8,13\}}^0$ | $2^{47} \times 2^{76} \times \frac{1}{11} \times \mathcal{S} \approx 2^{119.54} \times \mathcal{S}$ | $2^{47} \times 2^{-8} \times \mathcal{S} = 2^{39} \times \mathcal{S}$ | 1-round enc |
| Step 7 | $wk_{\{0,5,10,15\}}^0$ | $2^{39} \times 2^{92} \times \frac{1}{11} \times \mathcal{S} \approx 2^{127.54} \times \mathcal{S}$ | $2^{39} \times 2^{-4} \times \mathcal{S} = 2^{35} \times \mathcal{S}$ | 1-round enc |
| Step 8 | $k_{\{1,4,7,11,13\}}^0[0]$ | $2^{35} \times 2^{97} \times \frac{2}{11} \times \mathcal{S} \approx 2^{129.54} \times \mathcal{S}$ | $2^{35} \times 2^{-4} \times \mathcal{S} = 2^{31} \times \mathcal{S}$ | 2-round enc |
| Step 9 | $k_8^0$ | $2^{31} \times 2^{101} \times \frac{2}{11} \times \mathcal{S} \approx 2^{129.54} \times \mathcal{S}$ | $2^{31} \times 2^{-4} \times \mathcal{S} = 2^{27} \times \mathcal{S}$ | 2-round enc |
| Step 10 | $k_{\{0,10\}}^0[0,1,2]$ | $2^{27} \times 2^{107} \times \frac{2}{11} \times \mathcal{S} \approx 2^{131.54} \times \mathcal{S}$ | $2^{27} \times 2^{-4} \times \mathcal{S} = 2^{23} \times \mathcal{S}$ | 2-round enc |
| Step 11 | $k_{15}^0$ | $2^{23} \times 2^{111} \times \frac{2}{11} \times \mathcal{S} \approx 2^{131.54} \times \mathcal{S}$ | $2^{23} \times 2^{-16} \times \mathcal{S} = 2^{7} \times \mathcal{S}$ | 2-round enc |
| Step 13 | remaining bits | $2^{128} \times \beta \times (1 - 2^{-64})$ | - | 11-round enc |

**Step 12:** For the 111-bit key guesses in total, we prepare counters to confirm the number of right pairs that validate the differential pairs of the 8-round truncated differential distinguisher. The number of right pairs follows a binomial distribution with parameters $(N, p_0 = 2^{-31.565})$ in the case of the good key and $(N, p_1 = 2^{-56})$ otherwise[13].

**Step 13:** We fix the threshold as $\Upsilon$, and the key guess will be accepted as a candidate if the counter of right pairs is no less than $\Upsilon$. For all surviving candidates for the 111-bit key, we exhaustively search for the remaining key bits with at most two plaintext-ciphertext pairs.

*Complexity Analysis.* We use the same method for the complexity analysis as described in Sect. 5.2. Table 14 lists the detailed computation of the time complexity for the 11-round attack. When we set the threshold as $\Upsilon = 2$ and the number of structures as $\mathcal{S} = 2^{-21.74}$, we derive the success probability of $P_S = 80.0\%$ and the false alarm error probability of $\beta = 2^{-46.42}$. To summarize, the 11-round attack on QARMA64 under the RT setting is feasible with the time, data, and memory complexities of $2^{111.16}$, $2^{34.26}$, and $2^{111.00}$, respectively.

## F 12-round Attack on QARMA128 under RT Setting

To launch our 12-round attack, we append one key-recovery round before and after the 10-round truncated differential distinguisher, respectively. Fig. 8 shows the propagation of active cells for the 12-round attack on QARMA128 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key

[13] This attack uses the 8-round truncated differential distinguisher from the output to the input directions.

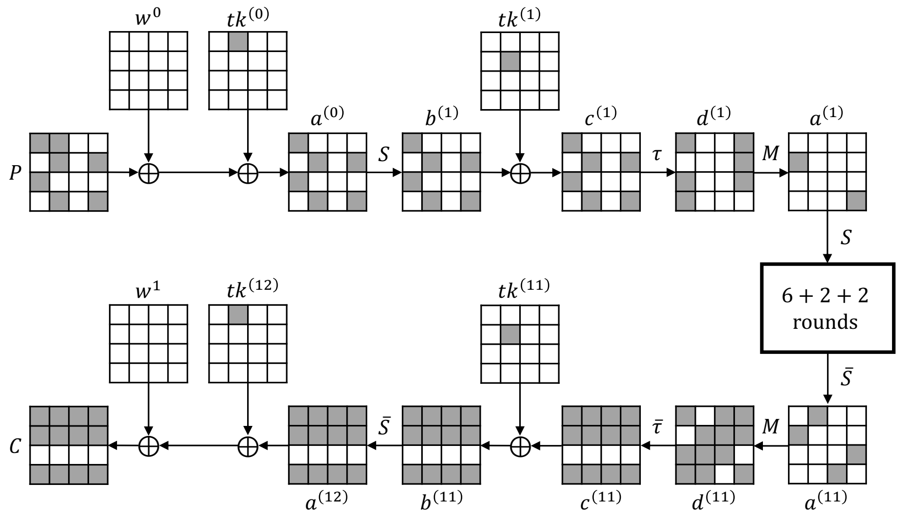


Fig. 8: Propagation of active cells for the 12-round attack on QARMA128 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$).

$k^0$). We note that $a^{(1)}$, $a^{(11)}$, and $tk^{(2)}$ represent the target input, output, and tweak differentials, respectively. Our attack is based on the following two phases: data collection and key recovery phases.

*Data Collection Phase.* From Fig. 8, the plaintext $P_{\{0,1,5,7,8,13,15\}}$ and the tweak $T_1^{(0)}$ are always active. In addition, $\Delta a_1^{(0)} = \Delta P_1 \oplus \Delta T_1^{(0)} = 0$ (i.e., $\Delta P_1 = \Delta T_1^{(0)}$) must hold. Under these conditions, the details of the data collection phase are described as follows.

**Step 1:** We prepare $\mathcal{S}$ structures. In each structure, there are $2^{56}$ plaintext-tweak pairs such that the plaintext $P$ and the tweak $T^{(0)}$ satisfying the above conditions traverse all possible values while the remaining cells are fixed to some random constants. Each structure leads to approximately $(2^{56})^2/2 = 2^{111}$ differential pairs.

**Step 2:** For each structure, we simulate the encryption oracle to get the ciphertexts $C$ from $(P, T^{(0)})$ pairs. Then, we insert every ciphertext with its corresponding plaintext-tweak pair $(P, T^{(0)}, C)$ into a hash table $\mathbb{H}$ at index $C_{\{8,9,10,11\}}$. Thus, each pair from the same index of $\mathbb{H}$ satisfies the differential pattern of $\Delta C_{\{8,9,10,11\}} = 0$. All pairs violating its differential pattern are discarded without further processing because these pairs have no hope to comply with the target truncated differential. This step reduces the differential pair used in the next phase from $2^{111} \times \mathcal{S}$ to $2^{79} \times \mathcal{S}$.

*Key Recovery Phase.* Hereafter, we denote $wk^0 = w^0 \oplus k^0$ and $wk^1 = w^1 \oplus k^0$. In the key recovery phase, we try to recover all bits of $wk^0$ and $wk^1$. Once these bits are recovered, all of the outer whitening and core keys (i.e., $w^0$, $w^1$, $k^0$, and $k^1$) can be naturally obtained based on the key specialization of QARMA (see Appendix A). The details of the key recovery phase are described as follows.

**Step 1:** For each possible 24-bit value of $wk^0_{\{0,5,15\}}$, we compute $\Delta a^{(1)}_{\{0,4,8,12\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{0,8,12\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{55} \times \mathcal{S}$.

**Step 2:** For each possible 24-bit value of $wk^0_{\{7,8,13\}}$, we compute $\Delta a^{(1)}_{\{3,7,11,15\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{3,7,11\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{31} \times \mathcal{S}$. $\Delta a^{(1)}_{\{1,2,5,6,9,10,13,14\}} = 0$ must be derived from the collected data; thus, we obtain here $N = 2^{31} \times \mathcal{S}$ pairs that match the input truncated differential of the 8-round truncated distinguisher.

**Step 3:** For each possible 24-bit value of $wk^1_{\{0,5,15\}}$, we compute $\Delta a^{(11)}_{\{0,4,8,12\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(11)}_{\{0,8,12\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{7} \times \mathcal{S}$.

**Step 4:** For each possible 24-bit value of $wk^1_{\{1,4,14\}}$, we compute $\Delta a^{(11)}_{\{1,5,9,13\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(11)}_{\{5,9,13\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{-17} \times \mathcal{S}$.

**Step 5:** For each possible 24-bit value of $wk^1_{\{3,6,12\}}$, we compute $\Delta a^{(11)}_{\{2,6,10,14\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(11)}_{\{2,6,10\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{-41} \times \mathcal{S}$.

**Step 6:** For each possible 24-bit value of $wk^1_{\{2,7,13\}}$, we compute $\Delta a^{(11)}_{\{3,7,11,15\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(11)}_{\{3,7,15\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{-65} \times \mathcal{S}$.

**Step 7:** For the 144-bit key guesses in total, we prepare counters to confirm the number of right pairs that validate the differential pairs of the 10-round truncated differential distinguisher. The number of right pairs follows a binomial distribution with parameters $(N, p_0 = 2^{-80.431})$ in the case of the good key and $(N, p_1 = 2^{-96})$ otherwise.

**Step 8:** We fix the threshold as $\Upsilon$, and the key guess will be accepted as a candidate if the counter of right pairs is no less than $\Upsilon$. For all surviving candidates for the 144-bit key, we exhaustively search for the remaining key bits with at most two plaintext-ciphertext pairs.

Table 15: Detailed computation of the time complexity for the 12-round attack.

| Step | Guessed key bits | Time complexity | # of remaining pairs | Note |
|---|---|---|---|---|
| Step 1 | $wk^0_{\{0,5,15\}}$ | $2^{79} \times 2^{24} \times \frac{1}{12} \times \mathcal{S} \approx 2^{99.41} \times \mathcal{S}$ | $2^{79} \times 2^{-24} \times \mathcal{S} = 2^{55} \times \mathcal{S}$ | 1-round enc |
| Step 2 | $wk^0_{\{7,8,13\}}$ | $2^{55} \times 2^{48} \times \frac{1}{12} \times \mathcal{S} \approx 2^{99.41} \times \mathcal{S}$ | $2^{55} \times 2^{-24} \times \mathcal{S} = 2^{31} \times \mathcal{S}$ | 1-round enc |
| Step 3 | $wk^1_{\{0,5,15\}}$ | $2^{31} \times 2^{72} \times \frac{1}{12} \times \mathcal{S} \approx 2^{99.41} \times \mathcal{S}$ | $2^{31} \times 2^{-24} \times \mathcal{S} = 2^{7} \times \mathcal{S}$ | 1-round dec |
| Step 4 | $wk^1_{\{1,4,14\}}$ | $2^{7} \times 2^{96} \times \frac{1}{12} \times \mathcal{S} \approx 2^{99.41} \times \mathcal{S}$ | $2^{7} \times 2^{-24} \times \mathcal{S} = 2^{-17} \times \mathcal{S}$ | 1-round dec |
| Step 5 | $wk^1_{\{3,6,12\}}$ | $2^{-17} \times 2^{120} \times \frac{1}{12} \times \mathcal{S} \approx 2^{99.41} \times \mathcal{S}$ | $2^{-17} \times 2^{-24} \times \mathcal{S} = 2^{-41} \times \mathcal{S}$ | 1-round dec |
| Step 6 | $wk^1_{\{2,7,13\}}$ | $2^{-41} \times 2^{144} \times \frac{1}{12} \times \mathcal{S} \approx 2^{99.41} \times \mathcal{S}$ | $2^{-41} \times 2^{-24} \times \mathcal{S} = 2^{-65} \times \mathcal{S}$ | 1-round dec |
| Step 8 | remaining bits | $2^{256} \times \beta \times (1 - 2^{-128})$ | - | 12-round enc |

*Complexity Analysis.* We use the same method for the complexity analysis as described in Sect. 5.3. Table 15 lists the detailed computation of the time complexity for the 12-round attack. When we set the threshold as $\Upsilon = 7$ and the number of structures as $\mathcal{S} = 2^{-52.52}$, we derive the success probability of $P_S = 80.0\%$ and the false alarm error probability of $\beta = 2^{-109.74}$. To summarize, the 12-round attack on QARMA128 under the RT setting is feasible with the time, data, and memory complexities of $2^{154.53}$, $2^{108.52}$, and $2^{144.00}$, respectively.

## G 13-round Attack on QARMA128 under RT Setting

To launch our 13-round attack, we append two and one key-recovery rounds before and after the 10-round truncated differential distinguisher, respectively. Fig. 9 shows the propagation of active cells for the 13-round attack on QARMA128 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$). We note that $a^{(2)}$, $a^{(12)}$, and $tk^{(3)}$ represent the target input, output, and tweak differentials, respectively. Our attack is based on the following two phases: data collection and key recovery phases.

*Data Collection Phase.* From Fig. 9, the plaintext $P_{\{0,1,2,4,5,7,8,10,11,13,14,15\}}$ and the tweak $T^{(0)}_5$ are always active. Under this condition, the details of the data collection phase are described as follows.

**Step 1:** We prepare $\mathcal{S}$ structures. In each structure, there are $2^{104}$ plaintext-tweak pairs such that the plaintext $P$ and the tweak $T^{(0)}$ satisfying the above conditions traverse all possible values while the remaining cells are fixed to some random constants. Each structure leads to approximately $(2^{104})^2/2 = 2^{207}$ differential pairs.

**Step 2:** For each structure, we simulate the encryption oracle to get the ciphertexts $C$ from $(P, T^{(0)})$ pairs. Then, we insert every ciphertext with its corresponding plaintext-tweak pair $(P, T^{(0)}, C)$ into a hash table $\mathbb{H}$ at index $C_{\{8,9,10,11\}}$. Thus, each pair from the same index of $\mathbb{H}$ satisfies the differential pattern of $\Delta C_{\{8,9,10,11\}} = 0$. All pairs violating its differential pattern

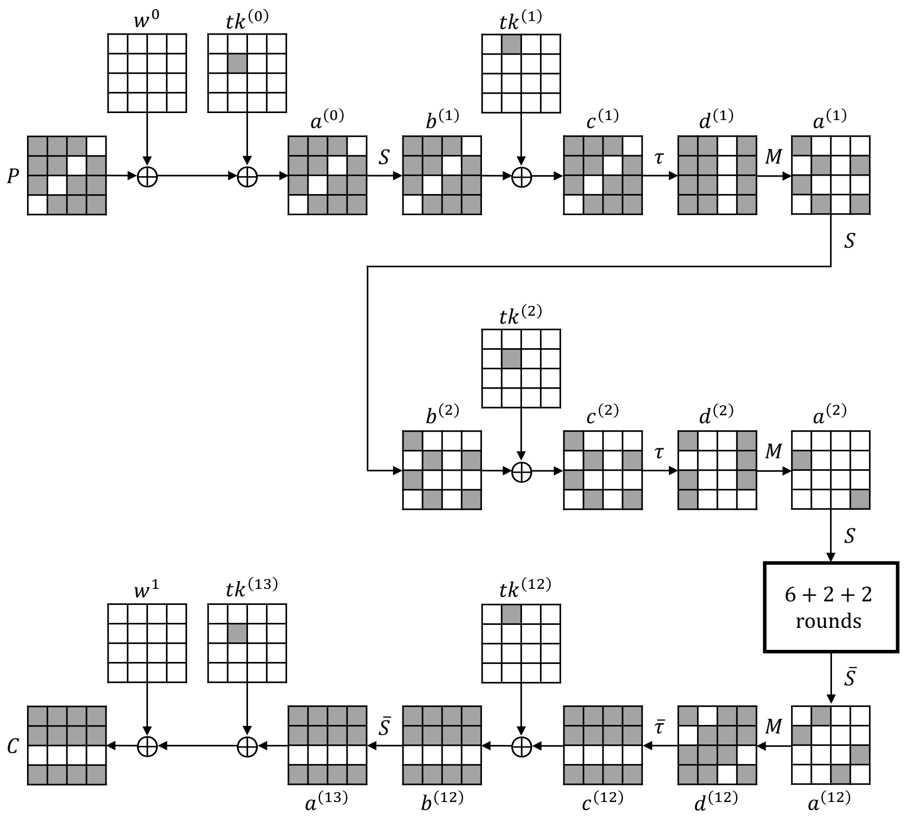


Fig. 9: Propagation of active cells for the 13-round attack on QARMA128 under the RT setting, where the gray areas represent the active cells and $tk_i$ denotes the $i$-th tweakey (i.e., the combination of the $i$-th round tweak $T^{(i)}$ and the core key $k^0$).

are discarded without further processing because these pairs have no hope to comply with the target truncated differential. This step reduces the differential pair used in the next phase from $2^{207} \times \mathcal{S}$ to $2^{175} \times \mathcal{S}$.

*Key Recovery Phase.* Hereafter, we use the following notations. We denote $wk^0 = w^0 \oplus k^0$ and $wk^1 = w^1 \oplus k^0$. When representing the bit position of each cell, we denote it as $s_i^{(r)}[0], \ldots, s_i^{(r)}[3]$, where $i$ represents the cell number, $r$ represents the number of rounds, $s_i^{(r)}[0]$ represents the most significant bit, and $s_i^{(r)}[3]$ represents the least significant bit.

In the key recovery phase, we try to recover all bits of $wk^0$ and $wk^1$. Once these bits are recovered, all of the outer whitening and core keys (i.e., $w^0$, $w^1$, $k^0$, and $k^1$) can be naturally obtained based on the key specialization of QARMA (see Appendix A). The details of the key recovery phase are described as follows.

**Step 1:** For each possible 24-bit value of $wk^1_{\{0,5,15\}}$, we compute $\Delta a^{(12)}_{\{0,4,8,12\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(12)}_{\{0,8,12\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{151} \times \mathcal{S}$.

**Step 2:** For each possible 24-bit value of $wk^1_{\{1,4,14\}}$, we compute $\Delta a^{(12)}_{\{1,5,9,13\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(12)}_{\{5,9,13\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{127} \times \mathcal{S}$.

**Step 3:** For each possible 24-bit value of $wk^1_{\{3,6,12\}}$, we compute $\Delta a^{(12)}_{\{2,6,10,14\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(10)}_{\{2,6,10\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{103} \times \mathcal{S}$.

**Step 4:** For each possible 24-bit value of $wk^1_{\{2,7,13\}}$, we compute $\Delta a^{(12)}_{\{3,7,11,15\}}$ from the ciphertexts $C$ and check whether $\Delta a^{(12)}_{\{3,7,15\}} = 0$. This step leads to a 24-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{79} \times \mathcal{S}$. Here, we obtain $N = 2^{79} \times \mathcal{S}$ pairs that match the output truncated differential of the 10-round truncated distinguisher.

**Step 5:** For each possible 32-bit value of $wk^0_{\{0,5,10,15\}}$, we compute $\Delta a^{(1)}_{\{0,4,8,12\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{4,12\}} = 0$. This step leads to a 16-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{63} \times \mathcal{S}$.

**Step 6:** For each possible 32-bit value of $wk^0_{\{1,4,11,14\}}$, we compute $\Delta a^{(1)}_{\{1,5,9,13\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{1,9\}} = 0$. This step leads to a 16-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{47} \times \mathcal{S}$.

**Step 7:** For each possible 32-bit value of $wk^0_{\{2,7,8,13\}}$, we compute $\Delta a^{(1)}_{\{3,7,11,15\}}$ from the plaintexts $P$ and check whether $\Delta a^{(1)}_{\{3,11\}} = 0$. This step leads to a 16-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{31} \times \mathcal{S}$.

**Step 8:** Hereafter, we guess $k^0$ using the previously guessed key bits. When guessing each possible 3-bit value of $k^0_{\{0,4,7\}}[0]$, we can uniquely determine the 82-bit value of $k^0_{\{0,1,2,4,5,7,8,13,14,15\}}$ and $k^0_{\{3,6\}}[0]$ in total. This is the same technique used in Step 8 in the key recovery phase of the 11-round attack on QARMA64 (see Appendix E for details). In addition, for each possible 8-bit value of $k^0_{10}$, we compute $\Delta a^{(2)}_{\{3,8\}}$ from the plaintexts $P$ and check whether $\Delta a^{(2)}_{3,8} = 0$. This step leads to a 16-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{15} \times \mathcal{S}$.

**Step 9:** For each possible 8-bit value of $k^0_{11}$, we compute $\Delta a^{(2)}_{\{0,4,7,11,12,15\}}$ from the plaintexts $P$ and check whether $\Delta a^{(2)}_{\{0,7,11,12\}} = 0$. This step leads to a 32-bit filter; thus, the number of surviving differential pairs after this step is approximately $2^{-17} \times \mathcal{S}$.

Table 16: Detailed computation of the time complexity for the 13-round attack.

| Step | Guessed key bits | Time complexity | # of remaining pairs | Note |
|---|---|---|---|---|
| Step 1 | $wk^1_{\{0,5,15\}}$ | $2^{175} \times 2^{24} \times \frac{1}{13} \times \mathcal{S} \approx 2^{195.29} \times \mathcal{S}$ | $2^{175} \times 2^{-24} \times \mathcal{S} = 2^{151} \times \mathcal{S}$ | 1-round dec |
| Step 2 | $wk^1_{\{1,4,14\}}$ | $2^{151} \times 2^{48} \times \frac{1}{13} \times \mathcal{S} \approx 2^{195.29} \times \mathcal{S}$ | $2^{151} \times 2^{-24} \times \mathcal{S} = 2^{127} \times \mathcal{S}$ | 1-round dec |
| Step 3 | $wk^1_{\{3,6,12\}}$ | $2^{127} \times 2^{72} \times \frac{1}{13} \times \mathcal{S} \approx 2^{195.29} \times \mathcal{S}$ | $2^{127} \times 2^{-24} \times \mathcal{S} = 2^{103} \times \mathcal{S}$ | 1-round dec |
| Step 4 | $wk^1_{\{2,7,13\}}$ | $2^{103} \times 2^{96} \times \frac{1}{13} \times \mathcal{S} \approx 2^{195.29} \times \mathcal{S}$ | $2^{103} \times 2^{-24} \times \mathcal{S} = 2^{79} \times \mathcal{S}$ | 1-round dec |
| Step 5 | $wk^0_{\{0,5,10,15\}}$ | $2^{79} \times 2^{128} \times \frac{1}{13} \times \mathcal{S} \approx 2^{203.29} \times \mathcal{S}$ | $2^{79} \times 2^{-16} \times \mathcal{S} = 2^{63} \times \mathcal{S}$ | 1-round enc |
| Step 6 | $wk^0_{\{1,4,11,14\}}$ | $2^{63} \times 2^{160} \times \frac{1}{13} \times \mathcal{S} \approx 2^{219.29} \times \mathcal{S}$ | $2^{63} \times 2^{-16} \times \mathcal{S} = 2^{47} \times \mathcal{S}$ | 1-round enc |
| Step 7 | $wk^0_{\{2,7,8,13\}}$ | $2^{47} \times 2^{192} \times \frac{1}{13} \times \mathcal{S} \approx 2^{235.29} \times \mathcal{S}$ | $2^{47} \times 2^{-16} \times \mathcal{S} = 2^{31} \times \mathcal{S}$ | 1-round enc |
| Step 8 | $k^0_{\{0,4,7\}}[0] \Vert k^0_{10}$ | $2^{31} \times 2^{203} \times \frac{2}{13} \times \mathcal{S} \approx 2^{231.29} \times \mathcal{S}$ | $2^{31} \times 2^{-16} \times \mathcal{S} = 2^{15} \times \mathcal{S}$ | 2-round enc |
| Step 9 | $k^0_{11}$ | $2^{15} \times 2^{211} \times \frac{2}{13} \times \mathcal{S} \approx 2^{223.29} \times \mathcal{S}$ | $2^{15} \times 2^{-32} \times \mathcal{S} = 2^{-17} \times \mathcal{S}$ | 2-round enc |
| Step 11 | remaining bits | $2^{256} \times \beta \times (1 - 2^{-128})$ | - | 13-round enc |

**Step 10:** For the 211-bit key guesses in total, we prepare counters to confirm the number of right pairs that validate the differential pairs of the 10-round truncated differential distinguisher. The number of right pairs follows a binomial distribution with parameters $(N, p_0 = 2^{-80.431})$ in the case of the good key and $(N, p_1 = 2^{-112})$ otherwise[14].

**Step 11:** We fix the threshold as $\Upsilon$, and the key guess will be accepted as a candidate if the counter of right pairs is no less than $\Upsilon$. For all surviving candidates for the 211-bit key, we exhaustively search for the remaining key bits with at most two plaintext-ciphertext pairs.

*Complexity Analysis.* We use the same method for the complexity analysis as described in Sect. 5.3. Table 16 lists the detailed computation of the time complexity for the 13-round attack. When we set the threshold as $\Upsilon = 2$ and the number of structures as $\mathcal{S} = 2^{2.63}$, we derive the success probability of $P_S = 80.2\%$ and the false alarm error probability of $\beta = 2^{-64.56}$. To summarize, the 13-round attack on QARMA128 under the RT setting is feasible with the time, data, and memory complexities of $2^{238.02}$, $2^{106.63}$, and $2^{211.00}$, respectively.

[14] This attack uses the 8-round truncated differential distinguisher from the output to the input directions.