Automatic Tool for Searching for Differential Characteristics in ARX Ciphers and Applications

Mingjiang Huang[1,2], Liming Wang[1]

[1] SKLOIS, Institute of Information Engineering, CAS, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences
{huangmingjiang, wangliming}@iie.ac.cn

Abstract. Motivated by the algorithm of differential probability calculation of Lipmaa and Moriai, we revisit the differential properties of modular addition. We propose an efficient approach to generate the input-output difference tuples with non-zero probabilities. A novel construction of combinational DDT, which makes it possible to obtain all valid output differences for fixed input differences. According to the upper bound of differential probability of modular addition, combining the optimization strategies with branch and bound search algorithm, we can reduce the search space of the first round and prune the invalid difference branches of the middle rounds. Applying this tool, the provable optimal differential trails covering more rounds for SPECK32/48/64 with tight probabilities can be found, and the differentials with larger probabilities are also obtained. In addition, the optimal differential trails cover more rounds than exisiting results for SPARX variants are obtained. A 12-round differential with a probability of $2^{-54.83}$ for SPARX-64, and a 11-round differential trail with a probability of 2^{-53} for SPARX-128 are found. For CHAM-64/128 and CHAM-128/*, the 39/63-round differential characteristics we find cover 3/18 rounds more than the known results respectively.

Keywords: SPECK · SPARX · CHAM · ARX · Differential cryptanalysis· Automatic search · Block cipher

1 Introduction

ARX-based ciphers rely on modular addition to provide non-linearity while rotation and XOR provide diffusion, hence the name: Addition, Rotation, XOR [7]. Benefiting from the high efficiency of modular addition in software implementation, the ARX construction is favored by many cryptography designers. In recent years, a large number of primitives based on the ARX construction have emerged, such as HIGHT [15], LEA [14], SPECK [5], SPARX [11], CHAM [18] and the augmented ARX ciphers SIMON [5] and SIMECK [32]. On April 18, 2019, in the Round 1 Candidates of *Lightweight Cryptographic* (LWC) standards announced by NIST [1], the permutations of COMET, Limdolen, SNEIK and SPARKLE [2] etc. also adopt ARX construction (all available online at [1]).

Since the ARX-based primitives are not so well understood as the S-box based ciphers, the security analysis on them is more difficult. And the proof of the rigorous security of the ARX ciphers is still a challenging task. In the cryptographic community, investigations on ARX ciphers are still going on.

Differential cryptanalysis [6,27] is one of the most important means to evaluate the security of ARX ciphers. For differential attack, the first step is to find some differentials with high probabilities, as well as covering enough rounds. Differentials with high probabilities can be used to mount key recovery attack with less data and/or time complexity, and differentials with longer rounds can be ultilized to attack more rounds in the iterative block ciphers. To obtain good differentials of ARX ciphers, an effective method is with the help of automated analysis tools at present. Therefore, constructing efficient automated analysis tools to get the differential characteristics on ARX ciphers worth the effort.

Related works. There are mainly three types of automated analysis tools for ARX ciphers until now. The first one, by characterizing the properties of components in ARX ciphers as a set of satisfiability problems, then use the SAT/SMT solvers (MiniSAT, STP, Boolector, etc.) to search for the characteristics, such as in [3,4,17,22,28,30,31]. The second ones are based on the inequality solving tools, by converting the cryptographic properties into inequalities characterization problems, constructing (mixed) integer linear programming (MILP) models, and solving them by third-party softwares (such as Gurobi, SAGE, etc.). MILP method is also very efficient in searching differential characteristics for ARX ciphers [13,33,34,35]. The third ones, which are constructed directly by the branch and bound search algorithm (Matsui's approach) under the Markov assumption [19]. By investigating the differential propagation properties of the round function, the differential characteristics can be searched according to depth-first [8,10,16,25,26] or breadth-first strategies [9]. The execution efficiency in the search phase of the first two tools depend on the performance of the third-party softwares and the representation of the equalities/inequalities of differential properties, while the third tool mostly depends on the optimizing strategies to reduce the invalid difference branches for improving the search efficiency.

In 2014, Biryukov et al. applied Matsui's approach to the differential analysis on SPECK, and proposed a concept of partial difference distribution table (pDDT) [8,9]. Based on some heuristic strategies, the differential trails they obtained can not be guaranteed as optimal ones. Then, at FSE'16 [10], Biryukov et al. further improved the branch and bound algorithm for SPECK. In the first round, they traversed the input-output difference space by gradually increasing the number of active bits of the input-output difference tuple, according to the monotonicity of the differential probability of modular addition. In the middle rounds, they used the calculation algorithm of Lipmaa and Moriai to compute the differential probability directly. The optimal differential characterstic covering 9-round for SPECK32 with a probability of 2^{-30} was obtained in [10]. Fu et al. applied the MILP method in [13] and Song et al. adopted the SAT method in [30] to search for the optimal differential characteristics of SPECK, and the obtained optimal differential trails cover 9/11-round for SPECK32/48 with probabilities of $2^{-30}/2^{-45}$ respectively. In [13,30], for SPECK64/96/128, the good differential trails were obtained by connecting two or three short trails from the extention of one intermediate difference state with a differential probability weight of 0.

In [23,24], Liu et al. firstly proposed a concept of carry-bit-dependent difference distribution table (CDDT) and a carry-bit-dependent linear approximate table (CLAT), and gave a efficient method to compute the differential probability and linear correlation by looking up CDDT and CLAT. By dividing the differences of a big modular addtion into small chunks, all possible output differences and corresponding probabilities can be traced. Combined with Matsui's method and lookup tables, the optimal differential trails of SPECK32/48/64, and the 8/8-round optimal trails with probabilities of $2^{-30}/2^{-30}$ for SPECK96/128 were found. In particular, the delicated proofs were given. This may prompt us to introduce some optimization conditions for constructing a more refined implementation to improve the search efficiency, especially, for the differential effect.

In [4], Ankele et al. analyzed the differential characteristics of SPARX-64 by suing the SAT method, and they got a 10-round optimal differential trail with a probability of 2^{-42}. Up to now, there are still no third-party differential cryptanalysis results for SPARX-128 and CHAM.

Our Contributions. Firstly, we propose a method to construct the space of the valid input-output difference tuples of certain differential probability weight. We adopt the way to increase the differential probability weight monotonously, which can exclude the search space of impossible large probability weight of the first round. Secondly, in order to quickly obtain the possible output differences with non-zero probabilities correspond to the fixed input differences, we construct a combinational DDT with feasible storage complexity, which can be regarded as an improved implementation of CDDT in [23,24]. Thirdly, we achieve more delicate pruning conditions based on the probability upper bound of modular addition. Finally, combining these optimization strategies, the automatic tool to search for the differential characteristics on ARX ciphers can be constructed.

Better results can be obtained with our tool. For SPECK32/48/64, we get the 10/11/15-round differentials with probabilities of $2^{-31.55}/2^{-42.86}/2^{-60.39}$. And, a new 12-round differential for SPECK48 with probability of $2^{-47.3}$ is found. For SPARX-64, the 11/12-round optimal/good differential trail with probability of $2^{-48}/2^{-56}$, and a 12-round differential with probability of $2^{-54.83}$ are obtained. For SPARX-128, a 10-round differential with probability of $2^{-39.98}$ is obtained. For CHAM-64/128, we find a 39-round optimal differential trail with probability of 2^{-64}. For CHAM-128/*, the 63-round optimal differential trail with probability of 2^{-127} is a good improvement compared to the results already announced.

Outline. The remainder of this paper is organized as follows. In Section 2, we present some preliminaries encountered in this paper. In Section 3, we present the approach to construct the space of input-output difference tuples and the construction method of cDDT. We introduce an automatic search tool for ARX ciphers in Section 4. And we apply the new tool to SPECK, SPARX and CHAM in Section 5. Finally, we conclude our work in Section 6.

2 Preliminaries

2.1 Notation

In this paper, we mainly focus on the XOR-difference probability of modular addition, which is marked by xdp^+. If not specified, the differential probabilities in this paper all represent xdp^+. For modular additon $x \boxplus y = z$ with input difference (α, β) and output difference γ, the XOR-difference probability of modular addition is defined by

$$\mathrm{xdp}^+((\alpha,\beta) \to \gamma) = \frac{\#\{(x,y)|((x \oplus \alpha) \boxplus (y \oplus \beta)) \oplus (x \boxplus y) = \gamma\}}{\#(x,y)}. \quad (1)$$

Modular addition is the only nonlinear component in ARX ciphers that produces differential probabilities. The differential probability of each round is decided by the number of active modular additions (i.e N_A) in it. Let $(\alpha^{i,j}, \beta^{i,j}, \gamma^{i,j})$ be the differences of the j^{th} addition in the i^{th} round, there have,

$$\Pr(\Delta x_{i-1} \to \Delta x_i) = \prod_{j=1}^{N_A} \mathrm{xdp}^+((\alpha^{i,j}, \beta^{i,j}) \to \gamma^{i,j}). \quad (2)$$

Under the *Markov assumption*, when the round keys are choosen uniformly, the probability of a differential trail is the product of the probabilities of each round. For a r-round reduced iterative cipher, with input difference Δx_0 and output difference Δx_r, the probability of the differential trail is denoted by

$$\Pr(\Delta x_0 \xrightarrow{r} \Delta x_r) = \prod_{i=1}^{r} \prod_{j=1}^{N_A} \mathrm{xdp}^+((\alpha^{i,j}, \beta^{i,j}) \to \gamma^{i,j}). \quad (3)$$

For the differential effect, the differential probability (DP) can be counted by the probabilities of the differential trails with the same input and output differences. Let N be the number of trails be counted, it will contribute to get a more compact DP when N is large enough.

$$\mathrm{DP}(\Delta x_0 \xrightarrow{r} \Delta x_r) = \sum_{s=1}^{N} \Pr(\Delta x_0 \xrightarrow{r} \Delta x_r)_s. \quad (4)$$

In this paper, we let $\mathbb{F}_2^n$ be the n dimensional vector space over binary filed $\mathbb{F}_2^1 = \{0,1\}$. We use the symbols $\lll$, $\ggg$ to indicate rotation to the left and right, and $\ll$, $\gg$ to indicate the left and right shift operation, respectively. The binary operator symbols $\oplus$, $\wedge$, $||$, $\neg$ represent XOR, AND, concatenation, and bitwise NOT respectively. For a vector x, its Hamming weight is denoted by $\mathrm{wt}(x)$. x_i represnets the i^{th} bit in vector x, and $x_{[j,i]}$ represents the vector of bits i to j in x. $\Delta x = x \oplus x'$ represents the XOR difference of x and x'. $\mathbf{0}$ represents a zero vector. For a r-round optimal differential trail with probability of Pr, $Bw_r = -\log_2 \Pr$ represents the obtained differential probability weight of it, and $\overline{Bw_{r+1}}$ is the expected differential probability weight of the $(r+1)$-round optimal differential trail.

2.2 Differential Probability Calculation for Modular Addition

In [20], Lipmaa and Moriai proposed an algorithm to compute the XOR-difference probability of modular addition, which can be rewriten by *Theorem 1*.

Theorem 1. *(Algorithm 2 in [20]) Let* α, β *be the two* n*-bit input differences and* γ *is the* n*-bit ouput difference of addition modulo* 2^n, $x, x', y, y' \in \mathbb{F}_2^n$, $f(x,y) = x \boxplus y$, $x = x' \oplus \alpha$, $y = y' \oplus \beta$, *and* $\gamma = f(x,y) \oplus f(x',y')$. *For arbitrary* α, β *and* γ, *let* $\mathrm{eq}(\alpha,\beta,\gamma) := (\bar{\alpha} \oplus \beta) \wedge (\bar{\alpha} \oplus \gamma)$, $\mathrm{mask}(n) := 2^n - 1$, *and* $g(\alpha,\beta,\gamma) := \mathrm{eq}(\alpha \ll 1, \beta \ll 1, \gamma \ll 1) \wedge (\alpha \oplus \beta \oplus \gamma \oplus (\beta \ll 1))$. *The differential probability of* (α, β) *propagate to* γ *is denoted by*

$$\Pr\{(\alpha,\beta) \to \gamma\} = \begin{cases} 2^{-\mathrm{wt}(\neg \mathrm{eq}(\alpha,\beta,\gamma) \wedge \mathrm{mask}(n-1))}, & \text{if } g(\alpha,\beta,\gamma) = \mathbf{0}; \\ 0, & \text{else.} \end{cases}$$

Theorem 2. *Let* α, β *be the two* n*-bit input differences and* γ *is the* n*-bit ouput difference of addition modulo* 2^n, *the number of input-output difference tuples with probability of* 2^{-w} *is* $4 \cdot 6^w \cdot \binom{n-1}{w}$, *for any* $0 \leq w < n$ *(Theorem 6 in [21], which is derived from Theorem 2 in [20]).*

2.3 Liu's Theorem to Compute Differential Probability of Modular Addition.

In [23,24], Liu et al. proposed the concept of carry-bit-dependent S-box and carry-bit-dependent difference distribution table (CDDT), which represent the truth value table and difference distribution table of addition with carry bit. With carry-bit-dependent S-box, they divided a modulo addition into sequential small modulo additions with carry bit, which turned an ARX cipher into an S-box-like cipher. They also proposed an efficient method to compute the differential probability of modular addition with CDDTs. CDDTs can be constructed based on the algorithm of Lipmaa and Moriai. For the given input differences, the unknown output differences can be obtained by looking up the tables, and the corresponding differential probabilities can be calculated according to the following theorem.

Theorem 3 ([23,24]). *Let* $n = mt$, $\alpha, \beta, \gamma \in \mathbb{F}_2^n$, $A_k = \alpha[(k+1)t - 1 : kt]$, $B_k = \beta[(k+1)t - 1 : kt]$, *and* $\Gamma_k = \gamma[(k+1)t - 1 : kt]$, $0 \leq k \leq m-1$. *For* $A, B, \Gamma \in \mathbb{F}_2^n$, $a, b, c, d \in \mathbb{F}_2$, *let*

$$PS_d^{a,b,c}(A, B \to \Gamma) = \begin{cases} 2^{-\mathrm{wt}(\overline{\mathrm{eq}(\alpha,\beta,\gamma)} \wedge \mathrm{mask}(t_d))}, & \text{if } f^{a,b,c}(A,B,\Gamma) = 0_t; \\ 0, & \text{else.} \end{cases}$$

where $t_0 = t$, $t_1 = t - 1$. *Let* $d_k = 0$, $0 \leq k \leq m-2$, d_{m-1}, $A_{-1} = B_{-1} = \Gamma_{-1} = 0_t$, *then*

$$\Pr\{(\alpha,\beta) \to \gamma\} = \prod_{k=0}^{m-1} PS_{d_k}^{h(A_{k-1}),h(B_{k-1},h(\Gamma_{k-1}}(A_k, B_k, C_k).$$

3 The Input-Output Differences and the Differential Probabilities of Modular Addition

3.1 The Input-Output Difference Tuples of Non-zero Probability

In branch-bound search strategy, a naive method is to traverse the full space of the input-output difference tuples for each modular addition in the first round. However, it will lead to very large time complexity, when the word size n is too large. To address this, it's possible to reduce the search complexity by removing those impossible tuples of modular addition at the startting of the search. Here, we will introduce an efficient algorithm to achieve this goal.

Lemma 1. *Let α, β be the two n-bit input differences and γ is the n-bit ouput difference of modular addition with non-zero differential probability. Let δ be a n-bit auxiliary vector , for $0 \leq i \leq n-1$, the i^{th} bit of δ is denoted by*

$$\delta_i = \begin{cases} 0, \text{if } \alpha_i = \beta_i = \gamma_i; \\ 1, \text{else}. \end{cases}$$

Therefore, there have $\delta = \neg \text{eq}(\alpha, \beta, \gamma)$, and

$$\Pr\{(\alpha, \beta) \to \gamma\} = 2^{-\text{wt}(\delta \wedge \text{mask}(n-1))}.$$

Let $w = \text{wt}(\delta \wedge \text{mask}(n-1))$ be the differential probability weight, there should be $0 \leq w \leq n-1$. The Hamming weight of the vector $\delta_{[n-2,0]}$ equals to the differential probability weight w.

Definition 1. *For $w \geq 1$, we define an array $\Lambda := \{\lambda_w, \cdots, \lambda_1\}$, which contains w elements. The elements in Λ record the subscripts of the non-zero bits of vector $\delta_{[n-2,0]}$, called as the probability weight active positions. For $1 \leq j \leq w$, each element is denoted by $\lambda_j = i$, when $\delta_i \neq 0$, for $i = 0$ to $n-2$. For example, $\Lambda = \{3, 2, 0\}$, when $\delta_{[6,0]} = (0001101)_2$.*

Definition 2. *Let (α, β, γ) be the input-output difference tuples of addition modulo 2^n with non-zero probability. Let's define an array $D := \{d_{n-1}, \cdots, d_0\}$, which contains n elements. Where $d_i = \alpha_i || \beta_i || \gamma_i = 4d_{i,2} + 2d_{i,1} + d_{i,0}$, $d_i \in \mathbb{F}_2^3$, and $d_{i,2}, d_{i,1}, d_{i,0} \in \mathbb{F}_2^1$, for $0 \leq i \leq n-1$.*

Definition 3. *Let's define four sets to represent the possible values that d_i might belongs to, i.e. $U_0 = \{0, 3, 5, 6\}$, $U_0^* = \{3, 5, 6\}$, $U_1 = \{1, 2, 4, 7\}$, $U_1^* = \{1, 2, 4\}$.*

Corollary 1. *Let (α, β, γ) be the input-output difference tuples of addition modulo 2^n with probability weight of w. For $1 \leq j \leq w$, $1 \leq w \leq n-1$ and let $\lambda_0 = 0$ when $\lambda_1 > 0$, there should have,*
- for every element λ_j in Λ, the λ_j-th octal word in D should s.t. $d_{\lambda_j} \notin \{0, 7\}$;
- the elements between d_{λ_j} and $d_{\lambda_{j-1}}$ should be all 0, if and only if $d_{\lambda_j} \in U_0^*$;
- the elements between d_{λ_j} and $d_{\lambda_{j-1}}$ should be all 7, if and only if $d_{\lambda_j} \in U_1^*$;
- and $d_{\lambda_1} \in U_0^*$ in any case.

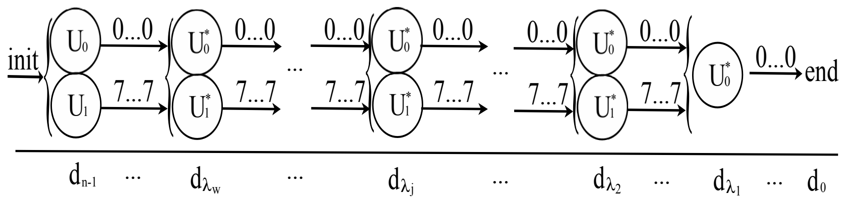

Fig. 1. The state transition diagram of the octal word sequence in D.

Algorithm 1: $Gen(w)$. Generating the input-output difference tuples of differential probability weight w for modular addition, $0 \le w \le n-1$.

Input: The patterns of the *probability weight active positions* can be calculated from the combinations algorithm in [12], i.e. $\Lambda := \{the\ patterns\ of\ \binom{n-1}{w}\}$.

Func_MSB: // Constructing the most significant bits of α, β, γ.
for *each* $d_{n-1} = d_{n-1,2}||d_{n-1,1}||d_{n-1,0} \in \mathbb{F}_2^3$ **do**
  if $d_{n-1} \in U_0$ **then**
    $\alpha = d_{n-1,2}||\overbrace{0\cdots0}^{all\ 0s}$, $\beta = d_{n-1,1}||\overbrace{0\cdots0}^{all\ 0s}$, $\gamma = d_{n-1,0}||\overbrace{0\cdots0}^{all\ 0s}$;
    If $w \ge 1$, call Func_Middle(w); else output each tuple (α, β, γ);
  else
    $\alpha = d_{n-1,2}||\overbrace{1\cdots1}^{all\ 1s}$, $\beta = d_{n-1,1}||\overbrace{1\cdots1}^{all\ 1s}$, $\gamma = d_{n-1,0}||\overbrace{1\cdots1}^{all\ 1s}$; $//d_{n-1} \in U_1$.
    If $w \ge 1$, call Func_Middle(w); else output each tuple (α, β, γ);
  end
end

Func_Middle(j): // Constructing the middle bits of α, β, γ.
if $j \le 1$ **then**
  call Fun_LSB;
end
for *each* $d_{\lambda_j} \in U_0^* \cup U_1^*$ **do**
  $\alpha_{\lambda_j} = d_{\lambda_j,2}$, $\beta_{\lambda_j} = d_{\lambda_j,1}$, $\gamma_{\lambda_j} = d_{\lambda_j,0}$;
  if $d_{\lambda_j} \in U_0^*$ **then**
    Set the bit strings of α, β, γ with subscripts $\lambda_{j-1} \to \lambda_j - 1$ to all 0;
  else
    Set the bit strings of α, β, γ with subscripts $\lambda_{j-1} \to \lambda_j - 1$ to all 1; // $d_{\lambda_j} \in U_1^*$.
  end
  call Func_Middle($j-1$);
end

Func_LSB: // Constructing the bits of α, β, γ with subscripts $0 \to \lambda_1$.
if $\lambda_1 > 0$ **then**
  Set the bit strings of (α, β, γ) with subscripts $0 \to \lambda_1 - 1$ to all 0;
end
for *each* $d_{\lambda_1} \in U_0^*$ **do**
  $\alpha_{\lambda_1} = d_{\lambda_1,2}$, $\beta_{\lambda_1} = d_{\lambda_1,1}$, $\gamma_{\lambda_1} = d_{\lambda_1,0}$;
  Output each tuple (α, β, γ);
end

Corollary 1 can be derived directly from Theorem 1. Inspired by the idea of finite-state machine (FSM) in [29], we take the most significant octal word d_{n-1} as the initial state to construct the state transition process of the elements in array D. The state transition diagram of octal word sequence that satisfy Corollary 1 is shown in Fig. 1. According to the distribution patterns of *probability weight active positions*, we introduce Algorithm 1 (marked as $Gen(w)$) to construct the $4 \cdot 6^w \cdot \binom{n-1}{w}$ input-output difference tuples of a certain differential probability weight w. All combinations of $\binom{n-1}{w}$ are produced by only single bit exchanges [12]. The output tuples do not need to be stored. The element d_i in D correspond to the bit values $(\alpha_i, \beta_i, \gamma_i)$ of the input-output difference tuples. Algorithm 1 traverses the values of the n elements in D and assigns them to the bits $(\alpha_i, \beta_i, \gamma_i)$, the total complexity of it will not be greater than $4 \cdot 6^w \cdot \binom{n-1}{w} \cdot 3n$.

3.2 The Combinational DDT

Generating a DDT that can be looked up is an efficient method to obtain the valid output differences for fixed input difference. For addition modulo 2^n, when n is too large, the full DDT will be too large to store. Hence, an intuitive idea is to store only a part of it. In [9], pDDT was introduced to precompute and store the difference tuples with probabilities above a fixed threshold. However, for the tuples that cannot be looked up in pDDT, their probabilities need to be calculated by the algorithm of Lipmaa and Moriai. In [23,24], the concept of CDDT was introduced with detailed proofs, by it, the differential probability of modular addition with n-bit word size can be obtained by the probabilities of the m-bit chunks after splitting. When considering to fully mine the useful information in DDT, here, we will introduce an improved different construction of combinational DDT (cDDT). cDDT represents the difference distribution tables for m-bit chunks of the n-bit words. By cDDT, the full DDT can be dynamically reconstructed on-the-fly during search, and the prunning conditions can be bound. cDDT stores fine-grained index entries and probability weights, while CDDT stores probabilities. The detailed theorems for calculating the differential probabilities of input-output difference tuples were described in [23,24], here, it can also be denoted by Lemma 2.

Lemma 2. *Let α, β, γ be the input-output differences of addition modulo 2^n, $\alpha' = \alpha \ll 1$, $\beta' = \beta \ll 1$, $\gamma' = \gamma \ll 1$, $\alpha, \alpha', \beta, \beta', \gamma, \gamma' \in \mathbb{F}_2^n$ and $n = mt$. Spliting $\alpha, \alpha', \beta, \beta', \gamma, \gamma'$ into t m-bit sub-vectors. If the equations*

$$\mathrm{eq}(\alpha'_{[(j+1)m-1,jm]}, \beta'_{[(j+1)m-1,jm]}, \gamma'_{[(j+1)m-1,jm]}) \wedge$$
$$(\alpha_{[(j+1)m-1,jm]} \oplus \beta_{[(j+1)m-1,jm]} \oplus \gamma_{[(j+1)m-1,jm]} \oplus \beta'_{[(j+1)m-1,jm]}) = \mathbf{0}$$

are satisfied for $0 \leq j \leq t-1$, there should be

$$-\log_2 \Pr = \sum_{j=0}^{t-2} \mathrm{wt}(\neg \mathrm{eq}(\alpha_{[(j+1)m-1,jm]}, \beta_{[(j+1)m-1,jm]}, \gamma_{[(j+1)m-1,jm]}) \wedge \mathrm{mask}(m))$$
$$+ \mathrm{wt}(\neg \mathrm{eq}(\alpha_{[n-1,n-m]}, \beta_{[n-1,n-m]}, \gamma_{[n-1,n-1m]}) \wedge \mathrm{mask}(m-1)).$$

Proof. When $\Pr \neq 0$, $g(\alpha, \beta, \gamma) = \mathbf{0}$ should be satisfied, which is equivalent to each m-bit sub-vector of $g(\alpha, \beta, \gamma)$ should be zero vector. As $-\log_2 \Pr = \mathrm{wt}(\delta_{[n-2,0]})$, the Hamming weight of vector $\delta_{[n-2,0]}$ can be split into $\mathrm{wt}(\delta_{[n-2,0]}) = \sum_{j=0}^{t-2} \mathrm{wt}(\delta_{[(j+1)m-1,jm]}) + \mathrm{wt}(\delta_{[n-2,n-m]})$. Hence, the probability weight is the sum of the weights of each m-bit sub-vector of $\delta_{[n-2,0]}$, when all m-bit sub-vectors of $g(\alpha, \beta, \gamma)$ are zero vectors. □

For each sub-vector tuple $(\alpha_{[(j+1)m-1,jm]}, \beta_{[(j+1)m-1,jm]}, \gamma_{[(j+1)m-1,jm]}$, or called as *sub-block*, its corresponding probability weight also depends on bits α_{jm-1}, β_{jm-1}, and γ_{jm-1}. Let $c[j] = \alpha_{jm-1}||\beta_{jm-1}||\gamma_{jm-1} \in \mathbb{F}_2^3$ (called as *carry bits*), and $\alpha_{[(j+1)m-1,jm]}$, $\beta_{[(j+1)m-1,jm]}$, $\gamma_{[(j+1)m-1,jm]} \in \mathbb{F}_2^m$, by traversing the 2^{3m+3} bits, a m-bit difference distribution table with non-zero probabilities can be pre-computed.

During the search process, the input differences (α, β) of modular addition are known, while the output difference γ and corresponding probability are unknown. For each m-bit sub-block $(\alpha_{[(j+1)m-1,jm]}, \beta_{[(j+1)m-1,jm]}, \gamma_{[(j+1)m-1,jm]})$, where $(\alpha_{[(j+1)m-1,jm]}, \beta_{[(j+1)m-1,jm]})$ are known. Considerring the $\lll 1$ operator, the bits $\alpha'_0, \beta'_0, \gamma'_0$ should be all zeros. By traversing the m-bit sub-vector $\gamma_{[m-1,0]}$, the possible probability weights of the least significant sub-block can be generated. And for a definite $\gamma_{[m-1,0]}$, the bits $\alpha_{m-1}||\beta_{m-1}||\gamma_{m-1}$ can also be obtained.

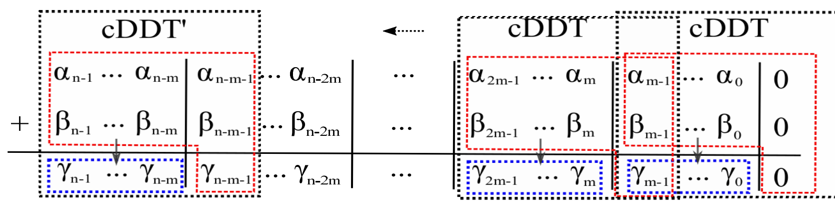

Fig. 2. The process of generating γ by looking up the difference distribution table.

Recursively, by traversing the other $t-1$ sub-vectors of γ, the corresponding probability weight of each sub-block can also be generated. Therefore, all valid n-bit output differences γ can be concatenated by the t sub-vectors of of γ, and the probability weight of this modular addition is the sum of probability weight of each sub-block. The dynamic generation process of γ is shown in Fig. 2.

For fixed input differences (α, β), the possible output difference γ with non-zero probability can be combined recursively by (5), where $c[0] = 0$ and $0 \leq j \leq t-1$. For each sub-block, the mapping can be pre-computed and stored by Algorithm 2, called as combinational DDT (cDDT) of modular addition. For each m-bit sub-vector of γ, it can be indexed by α, β, *carry bits* $c[j]$, corresponding probability weight w and the number of counts $N[w]$. It should be noted that, from the LSB to MSB direction, the *carry bits* $c[j]$ are obtained by the highest bits of the adjacent lower sub-block.

$$\begin{cases} c[j] = \alpha_{jm-1}||\beta_{jm-1}||\gamma_{jm-1}; \\ \gamma_{[(j+1)m-1,jm]} := \text{cDDT}(\alpha_{[(j+1)m-1,jm]}, \beta_{[(j+1)m-1,jm]}, c[j], w, N[w]). \end{cases} \quad (5)$$

For fixed word size n, when m is large, the number of sub-blocks t should be small, and less times of queries in the combination phase. However, when m is too large, the complexity of the pre-computing time and storage space of Algorithm 2 will also be too large. After the trade-off in storage size and lookup times, we choose $m = 8$. Before the procedure to search for the differential characteristics,

Algorithm 2: Pre-computing the m-bit combinational DDTs.

```
for each α, β ∈ F_2^m do
    α' = α ≪ 1, β' = β ≪ 1, AB = α||β;
    for each c = c_2||c_1||c_0 ∈ F_2^3 do
        Assign arrays N and N' with all zero;
        for each γ ∈ F_2^m do
            γ' = γ ≪ 1, α'_0 = c_2, α* = ¬α', β'_0 = c_1, γ'_0 = c_0;
            eq = (α* ⊕ β') ∧ (α* ⊕ γ') ∧ (α ⊕ β ⊕ γ ⊕ β');
            if eq = 0 then
                w = wt(¬((¬α ⊕ β) ∧ (¬α ⊕ γ)));
                cDDT[AB][c][w][N[w]] = γ;  // 0 ≤ w ≤ m.
                N[w] + +; // Number of γ with probability weight of w.
                w' = wt(¬((¬α ⊕ β) ∧ (¬α ⊕ γ)) ∧ mask(m − 1));
                cDDT'[AB][c][w'][N'[w']] = γ; // 0 ≤ w' ≤ m − 1.
                N'[w'] + +; // Number of γ with probability weight of w'.
            end
        end
        for 0 ≤ i ≤ m do
            cDDT_num[AB][c][i] = N[i]; // The number of γ with probability weight of i.
        end
        cDDT_wt_min[AB][c] = min{i|N[i] ≠ 0};  // The minimum probability weight.
        for 0 ≤ i ≤ m − 1 do
            cDDT'_num[AB][c][i] = N'[i];
        end
        cDDT'_wt_min[AB][c] = min{i|N'[i] ≠ 0};
    end
end
```

we first run Algorithm 2 to generate cDDT and cDDT$'$, where cDDT$'$ is used for the most significant sub-block. Algorithm 2 takes about several seconds[1] and about 16GB of storage space when $m = 8$. Analogously, when only input difference α is fixed, the input difference β and output difference γ can also be indexed by a similar construction method, this variant of cDDT is omitted here.

3.3 Probability Upper Bound and Pruning Conditions

The exact probability upper bound can be used to prune the branches in the intermediate rounds and reduce the unnecessary search space.

Corollary 2. *Let α, β be the two input differences of addition modulo 2^n, for any n-bit output difference γ with differential probability* $\Pr \neq 0$*, the upper bound of the probability should s.t.* $\mathrm{wt}((\alpha \oplus \beta) \wedge \mathrm{mask}(n-1)) \leq -\log_2 \Pr$.

Proof. When $\Pr \neq 0$, it's easy to get that the elements in array D should s.t. $d_i \in U_0^* \cup U_1^*$. When $d_i \in \{2, 3, 4, 5\}$, there have $\delta_i = \alpha_i \oplus \beta_i$, and for $d_i \in \{1, 6\}$ there should be $\delta_i > \alpha_i \oplus \beta_i$. Therefore, $\mathrm{wt}(\delta \wedge \mathrm{mask}(n-1)) \geq \mathrm{wt}((\alpha \oplus \beta) \wedge \mathrm{mask}(n-1))$ always hold when $\Pr \neq 0$. □

For fixed input difference (α, β), the probability weight correspond to all valid output difference γ can be obtained by summing the probability weights

[1]The time cost depends on the ability of the computation environment. On a 2.5 GHz CPU, it takes about 9 seconds.

of all sub-blocks. The possible probability weight should subject to (6).

$$\begin{aligned} -\log_2 \Pr \geq & \operatorname{wt}((\alpha_{[n-1,n-m]} \oplus \beta_{[n-1,n-m]}) \wedge \operatorname{mask}(m-1)) \\ & + \sum_{j=0}^{t-2} \operatorname{wt}(\alpha_{[(j+1)m-1,jm]} \oplus \beta_{[(j+1)m-1,jm]}). \end{aligned} \tag{6}$$

Let probability weights of each sub-block be $W_{XOR}[j] = \operatorname{wt}(\alpha_{[(j+1)m-1,jm]} \oplus \beta_{[(j+1)m-1,jm]})$ for $0 \leq j \leq t-2$, and $W_{XOR}[t-1] = \operatorname{wt}(\alpha_{[n-2,n-m]} \oplus \beta_{[n-2,n-m]})$. For fixed input differences (α, β), $0 \leq j \leq t-1$, the probability weight of each valid γ should also subject to (7).

$$\begin{aligned} -\log_2 \Pr \geq & \sum_{l=j+1}^{t-1} W_{XOR}[l] + \\ & \sum_{k=0}^{j} -\log_2 \Pr((\alpha_{[(k+1)m-1,km]}, \beta_{[(k+1)m-1,km]}) \to \gamma_{[(k+1)m-1,km]}). \end{aligned} \tag{7}$$

Expressions (6) and (7) can be adopted as the pruning conditions to prune the branches delicately in the process of combine the n-bit γ, which can eliminate a large number of γ that will not be the intermediate difference states of the optimal differential trails.

4 Automatic Search Tool for ARX ciphers

In [9,10], Biryukov et al. proposed the framework of *threshold search* for ARX ciphers. In which, difference tuples belonging to pDDT can be found by looking up the pre-computed tables, while tuples with small probabilities that can't be found by looking up the pDDT which should be calculated by Theorm 1. The choosen of the threshold in this framework depends on the analyzed primitives. There is a good point that at the beginning of the search, instead of traversing all the input space, they gradually increase the active bits. Even though, due to the Hamming weight of the differences is limited by a heuristic method, the result of this search framework can not always be guaranteed to be optimal.

In [25], Liu et al. proposed a search framework to gradually reduce the expected differential probability and applied it to SIMON's optimal differential trail search. In [23,24], Liu et al. also proposed an automatic search algorithm for optimal differential characteristics in ARX ciphers. Their algorithm is based on Matsui's branch-and-bound approach, by looking up CDDTs to get all possible output differences and their probabilities when computing the differential probability of modular addition. The algorithm is applied to search the optimal differential characteristics of SPECK, and the optimal trails of SPECK32/48/64 can be obtained by their search framwork. However, the optimization strategies were not introduced, and the search efficiency can still to improve.

Inspired by the these search frameworks, we can improve their algorithms by indtroducing our optimization strategies to build an efficient automatic tool for optimal differential characteristics on general iterated ARX ciphers. The core idea of our tool is to prune the difference branches with impossible small probabilities by gradually increasing the probability weights of each modular addition, or called as *probability-weight-driven search*.

Assuming w_1 is the probability weight of the first round in the r-round optimal differential trail, there should be $w_1 + Bw_{r-1} \leq \overline{Bw_r}$. Hence, the total search space of the first round is no more than $\sum_{w_1=0}^{\overline{Bw_r}-Bw_{r-1}} 4 \cdot 6^{w_1} \cdot \binom{n-1}{w_1}$. By gradually increasing the probability weight w_1 of the first round and traversing all input-output difference tuples correspond to it, the search space with probability weight be greater than w_1 can be excluded.

In the intermediate rounds, we firstly split the input differences (α, β) of each modular additon into t m-bit sub-vectors respectively. Then, according to (6), verifying whether the minimum probability weight correspond to (α, β) satisfies the condition or not. For valid possible (α, β), call $Cap(\alpha, \beta)$. By looking up cDDTs and pruning the branches by (7), the valid γ and possible probability weight will be generated dynamically. The pseudo code given by Algorithm 3 which is applied to SPECK as an example.

In the subroutine $Cap(\alpha, \beta)$, the least significant $t-2$ sub-blocks will look up the cDDT. And the pruning condition $\sum_{s=1}^{i-1} w_s + \sum_{l=k+1}^{t-1} W_{XOR}[l] + \sum_{j=0}^{k} w_i^j + Bw_{r-i} \leq \overline{Bw_r}$ should be satisfied, in which w_i^j increases monotonously. For the most significant sub-block, to get all possible outputs of it by querying cDDT′. Then combinining all sub-blocks' outputs to reconstruct the n-bit output difference with probability weight of $w_i = \sum_{j=0}^{t-1} w_i^j$, and $\sum_{s=1}^{i-1} w_s + w_i + Bw_{r-i} \leq \overline{Bw_r}$, where $\gamma = \gamma_{[n-1,n-m]}||\cdots||\gamma_{[m-1,0]}$. Nevertheless, the delicate pruning condition $\sum_{s=1}^{i-1} w_s + \sum_{l=k+1}^{t-1} W_{XOR}[l] + \sum_{j=0}^{k} w_i^k + Bw_{r-i} \leq \overline{Bw_r}$ will exclude most branches with small probabilities.

Formula (8) is adopted to count the probability of differential effect. In this tool, the pruning condition can be modified as $\sum_{s=1}^{i-1} w_s + w_i + Bw_{r-i} \leq w_{max}$ (*statistical condition*) to filter out the trails with probability weights be larger than w_{max}. w_{min} is the probability weight of the optimal differential trail be selected. The DP is counted by all trails with probability weights between w_{min} and w_{max}. When the probabilities of corresponding trails are too small, these trails cannot or need not to be searched, as their contribution to the DP can be ignored. $\#Trails[w]$ is the number of differential trails with probability of 2^{-w}.

$$\text{DP} = \sum_{w=w_{min}}^{w_{max}} 2^{-w} \times \#\text{Trails}[w] \tag{8}$$

5 Applications and Results

5.1 Differential Characteristics for SPECK32/48/64

The SPECK [5] family ciphers are typical ARX ciphers that proposed by NSA in 2013, which have five variants, i.e. SPECK32/48/64/96/128. The state of the

Algorithm 3: Searching for the optimal differential trails of ARX ciphers, and taking the application to SPECK as an example, where $n = mt$, $r > 1$.

Input: The cDDTs are pre-computed by Algorithm 2. $Bw_1, \cdots, Bw_{r-1}$ have been recorded;

Program entry: //Bw_1 can be derived manually for most ARX ciphers.
Let $\overline{Bw_r} = Bw_{r-1} - 1$, and Bw_r = null;
while $\overline{Bw_r} \neq Bw_r$ **do**
    $\overline{Bw_r}++$; //The r-round expected weight increases monotonously from Bw_{r-1}.
    Call Procedure Round-1;
end
Exit the program and record the differential trail be found.;

Round-1: //w_1 increases monotonously.
for $w_1 = 0$ *to* $n - 1$ **do**
    if $w_1 + Bw_{r-1} > \overline{Bw_r}$ **then**
        Return to the upper procedure with FALSE state;
    end
    Call Algorithm 1 $Gen(w_1)$ and traverse each tuple (α, β, γ); //$Gen(w_1) \Rightarrow (\alpha, \beta, \gamma)$.
    if *call Round-I(2,γ,β) and the return value is TRUE* **then**
        Break from $Gen(w_1)$ and return TRUE;
    end
end
Return to the upper procedure with FALSE state;

Round-$I(i, \alpha, \beta)$: //Intermediate rounds, $2 \leq i \leq r$.
$\alpha' = \alpha \ggg r_a$, $\beta' = \alpha \oplus (\beta \lll r_b)$; // (r_a, r_b): rotation parameters.
Let $W_{XOR}[t-1] = \mathrm{wt}((\alpha'_{[n-1,n-m]} \oplus \beta'_{[n-1,n-m]}) \wedge \mathrm{mask}(m-1))$;
Let $W_{XOR}[j] = \mathrm{wt}(\alpha'_{[(j+1)m-1,jm]} \oplus \beta'_{[(j+1)m-1,jm]}$, for $0 \leq j \leq t-2$;
if $w_1 + ... + w_{i-1} + \sum_{j=0}^{t-1} W_{XOR}[j] + Bw_{r-i} > \overline{Bw_r}$ **then**
    Return to the upper procedure with FALSE state; //Condition (6).
end
Let $AB[j] = \alpha'_{[(j+1)m-1,jm]} || \beta'_{[(j+1)m-1,jm]}$, for $0 \leq j \leq t-1$;
Call $Cap(\alpha', \beta')$, and traverse each possible γ; //Where $w_i = -\log_2 xdp^+((\alpha', \beta') \to \gamma)$.
if $i = r$ *and* $w_1 + ... + w_{i-1} + w_i = \overline{Bw_r}$ **then**
    Let $Bw_r = \overline{Bw_r}$, break from $Cap(\alpha', \beta')$ and return TRUE; //The last round.
end
if *call Round-I$(i+1, \gamma, \beta')$ and the return value is TRUE,* **then**
    Break from $Cap(\alpha', \beta')$ and return TRUE;
end
Return to the upper procedure with FALSE state;

$Cap(\alpha, \beta)$: //Combining all possible γ correspond to (α, β).
for $k = 0$ *to* $t - 2$, *and let* $k' = t - 1$, $c[0] = 0$ **do**
    for $w_i^k = \mathrm{cDDT}_{\mathrm{wt}_{min}}[AB[k]][c[k]]$ *to* m **do**
        if $\sum_{s=1}^{i-1} w_s + \sum_{l=k+1}^{t-1} W_{XOR}[l] + \sum_{j=0}^{k} w_i^j + Bw_{r-i} \leq \overline{Bw_r}$; *//Condition (7).* **then**
            for $x = 0$ *to* $\mathrm{cDDT}_{num}[AB[k]][c[k]][w_i^k] - 1$ **do**
                $\gamma_{[km+m-1,km]} = \mathrm{cDDT}[AB[k]][c[k]][w_i^k][x]$;
                $c[k+1] = \alpha_{km+m-1} || \beta_{km+m-1} || \gamma_{km+m-1}$;
                if $k = t - 2$ **then**
                    for $w_i^{k'} = \mathrm{cDDT}'_{\mathrm{wt}_{min}}[AB[k']][c[k']]$ *to* $m - 1$ **do**
                        if $\sum_{s=1}^{i-1} w_s + \sum_{j=0}^{t-1} w_i^j + Bw_{r-i} \leq \overline{Bw_r}$ **then**
                            for $y = 0$ *to* $\mathrm{cDDT}'_{num}[AB[k']][c[k']][w_i^{k'}] - 1$ **do**
                                $\gamma_{[n-1,n-m]} = \mathrm{cDDT}'[AB[k']][c[k']][w_i^{k'}][y]$;
                                Output each $\gamma = \gamma_{[n-1,n-m]} || \cdots || \gamma_{[m-1,0]}$ and $w_i = \sum_{j=0}^{t-1} w_i^j$;
                            end
                        end
                    end
                end
            end
        end
    end
end

i^{th} round can be divided into two parts according to Feistel structure, i.e. X_r^i and X_l^i. Therefore, the round function transition process can be denoted by $X_l^{i+1} = ((X_r^i \ggg r_a) \boxplus X_l^i) \oplus rk^i$ and $X_r^{i+1} = X_l^{i+1} \oplus (X_r^i \lll r_b)$, in which the rk^i is the round subkey of the i^{th} round, and (r_a, r_b) are the rotation parameters of left and right part respectively. $(r_a, r_b) = (7{,}2)$ for SPECK32, and $(r_a, r_b) = (8{,}3)$ for other variants.

Property 1. For SPECK variants, let $(\alpha^i, \beta^i, \gamma^i)$ be the input-output differences of modular addition in the i^{th} round, $(\Delta X_l^i, \Delta X_r^i)$ and $(\Delta X_l^{i+1}, \Delta X_r^{i+1})$ are the input and output difference of i^{th} round. There are $\alpha^i \lll r_a = \Delta X_l^i$, $\beta^i = \Delta X_r^i$, $\gamma^i = \Delta X_l^{i+1}$, and $\gamma^i \oplus (\beta^i \lll r_b) = \Delta X_r^{i+1}$.

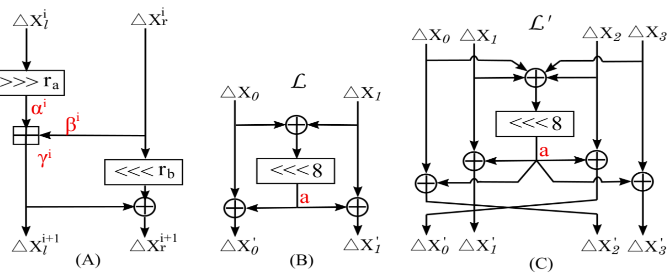

Fig. 3. The differential propagation of SPECK/SPECKEY is shown in (A), and the differential propagation of $\mathcal{L}/\mathcal{L}'$ are shown in (B) and (C).

By Algorithm 3, the optimal differential trails we obtained are shown in Table 1,2. The runtime[2] and the differential probabilities are slightly improved comparing to the existing results, and the obtained optimal differential trails can cover more rounds with less search time. A new 12-round differential for SPECK48 is obtained, shown in Table 3. For SPECK96/128, due to the large word size, the time complexity is still too large to directly search for the optimal differential trails covering more rounds with probabilities close to the security bound ($\Pr = 2^{-n}$).

5.2 Differential Characteristics for SPARX Variants

SPARX [11] was introduced by Dinu et al. at ASIACRYPT'16, which is designed according to the *long trail strategy* with provable bound. The SPECKEY component in SPARX, or called as ARX-Box, which is modified from the round function of SPECK32. The differential properties of SPECKEY are similar to that of the round function in SPECK32, see *Property 1*. For the 3 variants of S-PARX, we mark them as SPARX-64 and SPARX-128 according to the block size. For the linear layer functions $\mathcal{L}/\mathcal{L}'$ (shown in Fig. 3), their differential properties are listed in *Property 2,3*.

Table 1. Runtime and the probabilities of the optimal differential trails for SPECK variants. In the following tables, $w = -\log_2 \Pr$, the 's','m','h','d' represent the time in seconds, minutes, hours, and days respectively. 'u' means search time is unknown, and 't' for time. The columns of 'tw' indicate this work, and the time for pre-calculating the cDDTs are not counted.

r	SPECK32				SPECK48					SPECK64				
	w	[10] t	[24] t	tw t	[10] w	[10] t	w	[24] t	tw t	[10] w	[10] t	w	[24] t	tw t
1	0	0s	u	0s	0	0s	0	u	0s	0	0s	0	u	0s
2	1	0s	u	0s	1	0s	1	u	0s	1	0s	1	u	0s
3	3	0s	u	0s	3	0s	3	u	0s	3	0s	3	u	0s
4	5	0s	u	0s	6	0s	6	u	0s	6	0s	6	u	0s
5	9	0s	u	0s	10	1s	10	u	0s	10	1m	10	u	8s
6	13	1s	u	1s	14	3s	14	u	0s	15	26m	15	u	10m
7	18	1m	u	7s	19	1m	19	u	17s	21	4h	21	u	19m
8	24	34m	u	35s	26	9m	26	u	77s	27	22h	29	u	18h
9	30	12m	u	3m	33	7d	33	u	6h	>31	1d	34	u	1h
10	34	6m	u	2m	>34	3h	40	u	16h			38	u	40m
11							45	u	2h			42	u	11m
12							49	u	40m			46	u	5m
13												50	u	5m
14												56	u	20m
15												62	u	1h
16												70	u	91h

r	SPECK96						SPECK128					
	[10] w	[10] t	[24] w	[24] t	tw w	tw t	[10] w	[10] t	[24] w	[24] t	tw w	tw t
1	0	0s	0	u	0	0s	0	0s	0	u	0	0s
2	1	0s	1	u	1	0s	1	0s	1	u	1	0s
3	3	0s	3	u	3	0s	3	0s	3	u	3	0s
4	6	6s	6	u	6	0s	6	22s	6	u	6	2s
5	10	5m	10	u	10	2s	10	26m	10	u	10	13m
6	15	5h	15	u	15	11m	15	2d	15	u	15	80m
7	21	5d	21	u	21	18m	21	3h	21	u	21	2h
8	>27	3d	30	u	30	162h	>26	2d	30	u	≤30	32d
9			≤39	u	≤39	32d			≤39	u	≤39	28d
10			≤49	u					≤49	u		
11												
12												
13												
14												
15												
16												

Table 2. The 9/11/15-round optimal differential trails for SPECK32/48/64.

	SPECK32		SPECK48		SPECK64	
r	ΔX_r	w	ΔX_r	w	ΔX_r	w
0	8054A900	3	080048080800	3	4000409210420040	5
1	0000A402	3	400000004000	1	8202000000120200	4
2	A4023408	8	000000020000	1	0090000000001000	2
3	50C080E0	4	020000120000	3	0000800000000000	1
4	01810203	5	120200820200	4	0000008000000080	1
5	000C0800	3	821002920006	9	8000008080000480	3
6	20000000	1	918236018202	12	0080048000802084	6
7	00400040	1	0C1080000090	4	80806080848164A0	13
8	80408140	2	800480800000	2	040F240020040104	8
9	00400542	-	008004008000	3	2000082020200001	4
10			048080008080	3	0000000901000000	2
11			808400848000	-	0800000000000000	1
12					0008000000080000	2
13					0008080000480800	4
14					0048000802084008	6
15					0A0808081A4A0848	-

Table 3. The differentials for SPECK32/48/64.

2n	r	Δin	Δout	w_{min}	w_{max}	DP	Reference
32	9	8054,A900	0040,0542	30	N/A	2^{-30}	[8]
	9	8054,A900	0040,0542	30	N/A	$2^{-29.47}$	[30]
	10	2040,0040	0800,A840	35	N/A	$2^{-31.99}$	[30]
	10	0040,0000	0814,0844	36	48	$2^{-31.55}$	This paper.
48	11	202040,082921	808424,84A905	47	N/A	$2^{-46.48}$	[8]
	11	504200,004240	202001,202000	46	N/A	$2^{-44.31}$	[30]
	11	001202,020002	210020,200021	45	54	$2^{-43.44}$	This paper.
	11	080048,080800	808400,848000	45	54	$2^{-42.86}$	This paper.
	12	080048,080800	840084,A00080	49	52	$2^{-47.3}$	This paper.
64	14	00000009,01000000	00040024,04200D01	60	N/A	$2^{-59.02}$	[8]
	15	04092400,20040104	808080A0,A08481A4	62	N/A	$2^{-60.56}$	[30]
	15	40004092,10420040	0A080808,1A4A0848	62	71	$2^{-60.39}$	This paper.

Property 2. For SPARX-64, $(X_0', X_1') = \mathcal{L}(X_0, X_1)$, let $a = (\Delta X_0 \oplus \Delta X_1) \lll 8$, there should be $\Delta X_0' = \Delta X_0 \oplus a$, and $\Delta X_1' = \Delta X_1 \oplus a$.

Property 3. For SPARX-128, $(X_0', X_1', X_2', X_3') = \mathcal{L}'(X_0, X_1, X_2, X_3)$, let $a = (\Delta X_0 \oplus \Delta X_1 \oplus \Delta X_2 \oplus \Delta X_3) \lll 8$, there should be $\Delta X_0' = \Delta X_2 \oplus a$, $\Delta X_1' = \Delta X_1 \oplus a$, $\Delta X_2' = \Delta X_0 \oplus a$, and $\Delta X_3' = \Delta X_3 \oplus a$.

To obtain the optimal differential trails of SPARX, there should make some modifications to Algorithm 3. In the first round, it is necessary to call Algorithm 1 for each addition modulo 2^{16} to generate its input-output difference tuples with probability weight increase monotonously. There should be nested call Algorithm 1 2/4 times for SPARX-64/SPARX-128 respectively. For every modular additions in each intermediate round, $Cap(\alpha, \beta)$ needs to be nested multiple times to produce its valid output differences. The *Property 2/3* of linear layer functions $\mathcal{L}/\mathcal{L}'$ will be used to replace the linear properties of SPECK. The optimal differential trails and differentials for SPARX-64 are listed in Table 4[3] and Table 5. The 12-round optimal differential trail for SPARX-64 cover 2 more rounds than the existing results in [3,4]. The 12-round good differential trail is obtained by taking the input difference of the 11-round optimal differential trail as a fixed value. Refer to expression (8), if the searched w_{max} is large enough, the time complexity and the differential probability also should be larger[4].

Table 4. Probabilities of the optimal differential trails for SPARX-64.

r	$-\log_2 \Pr$	Δin	Δout	Time
1	0	0040 0000 0000 0000	8000 8000 0000 0000	0s
2	1	0040 0000 0000 0000	8100 8102 0000 0000	0s
3	3	0040 0000 0000 0000	8A04 8E0E 8000 840A	0s
4	5	0000 0000 2800 0010	8000 840A 0000 0000	0s
5	9	0000 0000 2800 0010	850A 9520 0000 0000	1s
6	13	0000 0000 0211 0A04	AF1A BF30 850A 9520	2s
7	24	0000 0000 1488 1008	8000 8C0A 8000 840a	2h38m
8	29	0000 0000 0010 8402	0040 0542 0040 0542	4h16m
9	35	2800 0010 2800 0010	D761 9764 D221 9224	4h54m
10	42	2800 0010 2800 0010	0204 0A04 0204 0A04	80h
11	48	2800 0010 2800 0010	0200 2A10 0200 2A10	194h35m
12	$\leq$56	2800 0010 2800 0010	0291 0291 2400 B502	-

The differential characteristics for SPARX-128 are shown in Table 6, and the 12/11-round good differential trail for SPARX-64/SPARX-128 are shown in Table 7. T_{opt}, T_{diff} are the time cost for searching the optimal differential trails and differntials respectively. The 9/10/11-round good differential trail with probability weight of 34/41/53 are obtained by limiting the probability weight $w_1 \leq 1$ of the first round, and T_{opt} is the corresponding time cost.

[2]All experiments in this paper are carried out serially on a HPC with Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz. All differences are represented in hexadecimal.

[3]For the 7-round optimal differential trail with probability weight of 24, we limit the first round probability weight $w_1 \leq 5$ to speed up the search process.

[4]When the *statistical condition* is omitted in the last round, #Trails will perhaps be greater than the sum of the number of trail with probability weight $\leq w_{max}$.

Table 5. Comparison of the differentials for SPARX-64.

r	Δin	Δout	w_{min}	w_{max}	DP	#Trails	Time	Reference
7	000000007448B0F8	80048C0E8000840A	24	60	$2^{-23.95}$	56301	28m	[3][4]
	0000000014881008	80008C0A8000840A	24	30	$2^{-23.82}$	4	12s	This paper.
8	0000000000508402	0040054200400542	29	60	$2^{-28.53}$	37124	17m	[3][4]
	0000000000108402	0040054200400542	29	46	$2^{-28.54}$	194	48m	This paper.
9	2800001028000010	5761176452211224	35	58	$2^{-32.87}$	233155	7h42m	[3][4]
	2800001028000010	D7619764D2219224	35	47	$2^{-32.96}$	399	12h19m	This paper.
10	2800001028000010	8081828380008002	42	73	$2^{-38.12}$	1294158	35h18m	[3][4]
	2800001028000010	02040A0402040A04	42	49	$2^{-38.05}$	362	17h18m	This paper.
11	2800001028000010	02002A1002002A10	48	53	$2^{-43.91}$	922	98h21m	This paper.
12	2800001028000010	029102912400B502	56	58	$2^{-54.83}$	9	17h37m	This paper.

Table 6. The differential characteristics for SPARX-128.

r	w_{opt}	T_{opt}	Δin	Δout	w_{min}	w_{max}	DP	#Trails	T_{diff}
4	5	0s	0000 0000 0000 0000 0000 0000 2800 0010	0000 0000 0000 040A 0000 0000 0000 0000	5	6	2^{-3}	63	16s
5	9	3m25s	0000 0000 0000 0000 0000 0000 2800 0010	0000 0000 850A 9520 0000 0000 0000 0000	9	12	2^{-9}	1	15s
6	13	7m	0000 0000 0000 0000 0000 0000 0211 0A04	0000 0000 850A 9520 0000 0000 0000 0000	13	16	2^{-13}	1	14s
7	18	17h18m	0000 0000 0000 0000 0000 0000 0a20 4205	0000 0000 850A 9520 0000 0000 0000 0000	18	22	2^{-18}	1	15s
8	24	24d17h	0000 0000 0000 0000 0000 0000 1488 1008	AF1A 2A10 2A10 BF30 0000 0000 850A 9520	24	28	$2^{-23.83}$	2	9s
9	≥ 29 ≤ 34	27m	0000 0000 0000 0000 0000 0000 2040 0040	0010 0010 0800 2800 0000 0000 0810 2810	34	42	$2^{-31.17}$	238	2h31m
10	≥ 38 ≤ 41	16h31m	0000 0000 0000 0000 0000 0000 0050 A000	8040 8140 A040 2042 0000 0000 2000 A102	41	48	$2^{-39.98}$	40	45h22m
11	≤ 53	17d19h	0000 0000 0000 0000 0000 0000 0050 A000	0040 0542 A102 200A 0000 0000 6342 E748	53	53	2^{-53}	1	-

Table 7. The 12/11-round good differential trail for SPARX-64 and SPARX-128.

12-round trail for SPARX-64					11-round trail for SPARX-128						
r	$\Delta X_r^0 \| \cdots \| \Delta X_r^3$	w_r^0	w_r^1	w_r	r	$\Delta X_r^0 \| \Delta X_r^1 \| \cdots \| \Delta X_r^6 \| \Delta X_r^7$	w_r^0	w_r^1	w_r^2	w_r^3	w_r
1	2800001028000010	2	2	4	1	0000000000000000000000000050A000	0	0	0	1	1
2	0040000000400000	0	0	0	2	00000000000000000000000000008002	0	0	0	2	2
3	8000800080008000	2	1	3	3	0000000000000000000000008006800C	0	0	0	6	6
$\mathcal{L}$	8300830281008102	-	-	-	4	0000000000000000000000009D0C9D3E	0	0	0	7	7
4	0000000083008302	0	5	5	$\mathcal{L}'$	0000000000000000000000008478F082	-	-	-	-	-
5	000000008404880E	0	6	6	5	000000008478F0820000000000000000	0	6	0	0	6
6	00000000911AB120	0	8	8	6	00000000C08A02810000000000000000	0	7	0	0	7
$\mathcal{L}$	00000000C4060084	-	-	-	7	000000000A0000040000000000000000	0	2	0	0	2
7	C406008400000000	8	0	8	8	00000000001000000000000000000000	0	1	0	0	1
8	0A14080400000000	4	0	4	$\mathcal{L}'$	00000000200020000000000000000000	-	-	-	-	-
9	2010000000000000	2	0	2	9	20000000000020000000000020002000	1	1	0	2	4
$\mathcal{L}$	2040204000000000	-	-	-	10	004000402000A000000000002040A040	1	2	0	2	5
10	2040204020402040	2	2	4	11	80408140A0402042000000002000A102	2	4	0	6	12
11	A0002100A0002100	3	3	6	12	00400542A102200A000000006342E748	-	-	-	-	-
12	2040A4402040A440	3	3	6							
$\mathcal{L}$	2400B5022400B502	-	-	-							
13	029102912400B502	-	-	-							

5.3 Differential Characteristics for CHAM Variants

CHAM [18] is a family of lightweight block ciphers that proposed by Koo et al. at ICISC'17, which combines the good design features of SIMON and SPECK. CHAM adopts a 4-branch generalized Feistel structure, and contains three variants which are denoted by CHAM-n/k with a block size of n-bit and a key size of k-bit. For CHAM-64/128, the word size w of each branch is 16 bits, and for CHAM-128/*, $w = 32$. The rotation parameters of every two consecutive rounds are (1,8) and (8,1) respectively, and it iterates over $R = 80/80/96$ rounds for the three variants.

Let $X_{r+1} = f_r(X_r, K)$ be the round function of the r^{th} round of CHAM, $1 \leq r \leq R$. Let's divide the input state $X_r \in \mathbb{F}_2^n$ of the r^{th} round into four w-bit words, i.e. $X_r = X_r[0]||X_r[1]||X_r[2]||X_r[3]$. The state transformation of the round function can be represented by

$$X_{r+1}[3] = ((X_r[0] \oplus (r-1)) \boxplus ((X_r[1] \lll r_a) \oplus RK[(r-1)\ mod\ 2k/w])) \lll r_b,$$

$$X_{r+1}[j] = X_r[j+1],\ for\ 0 \leq j \leq 2.$$

When $r\ mod\ 2 = 1$, there have $(r_a, r_b) = (1, 8)$, otherwise $(r_a, r_b) = (8, 1)$.

For a master key $K \in \mathbb{F}_2^k$ of CHAM, the key schedule process will generate $2k/w$ w-bit round keys, i.e. $RK[0], RK[1], \cdots, RK[2k/w-1]$. For $0 \leq i < k/w$, Let $K = K[0]||K[1]||\cdots||K[k/w-1]$, the round keys can be generated by

$$RK[i] = K[i] \oplus (K[i] \lll 1) \oplus (K[i] \lll 8),$$

$$RK[(i + k/w) \oplus 1] = K[i] \oplus (K[i] \lll 1) \oplus (K[i] \lll 11).$$

The input difference $\Delta X_r = X_r \oplus X'_r$ of the r^{th} round can be denoted by $\Delta X_r = \Delta X_r[0]||\Delta X_r[1]||\Delta X_r[2]||\Delta X_r[3]$, where $\Delta X_r[j] \in \mathbb{F}_2^w$, for $0 \leq j \leq 3$. Therefore, the differential propagation property of the round function of CHAM can be denoted by *Property 4*. The differential propagation process of the first 4 consecutive rounds of CHAM is shown in Fig. 4.

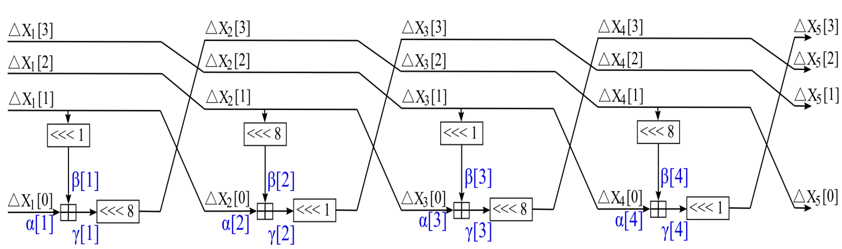

Fig. 4. The difference propagation for the first 4 rounds of CHAM.

Property 4. Let ΔX_r, ΔX_{r+1} be the input and output difference of the r^{th} round of CHAM, there are $\Delta X_{r+1}[0] = \Delta X_r[1]$, $\Delta X_{r+1}[1] = \Delta X_r[2]$, $\Delta X_{r+1}[2] = \Delta X_r[3]$, and $\Delta X_{r+1}[3] := \delta_{\mathrm{Pr}}(\Delta X_r[0], \Delta X_r[1] \lll r_a) \lll r_b$. Where $\gamma := \delta_{\mathrm{Pr}}(\alpha, \beta)$ represents the output difference γ of modular addition that generated by input differences (α, β) with differential probability of Pr.

In the search process, the input-output difference tuples $(\alpha[1], \beta[1], \gamma[1])$ can be generated by Algorithm 1 directly. Then $(\beta[2], \gamma[2])$ can be obtained by querying a variant of cDDT based on $\alpha[2] = \beta[1] \ggg 1$. And, $(\beta[3], \gamma[3])$ can also be queried by $\alpha[3] = \beta[2] \ggg 8$. When $r \geq 4$, the input differences $\Delta X_r[0]||\Delta X_r[1]||\Delta X_r[2]||\Delta X_r[3]$ can be determined, so, $\Delta X_{r+1}[3]$ can be obtained by querying cDDT based on $(\Delta X_r[0], \Delta X_r[1] \lll r_a)$. The probability weights of each splitted sub-blocks of the input-output difference tuples increase monotonously, and the *Property 4* should also be introduced, for $r \geq 2$.

It should be noted that, the rotation parameters in two consecutive rounds of CHAM are different. Let Bw_r^* be the probability weights of the truncated optimal differential trails that starting with rotation parameter $(r_a, r_b) = (8, 1)$. Hence, when searching for the optimal differential trail of CHAM, in the pruning condition $\sum_{s=1}^{i-1} w_s + w_i + Bw_{r-i} \leq \overline{Bw_r}$, if current round i is odd, the pruning condition should be replaced with $\sum_{s=1}^{i-1} w_s + w_i + Bw_{r-i}^* \leq \overline{Bw_r}$. Correspondingly, when searching for Bw_r^*, if current round i is even, the pruning condition should be $\sum_{s=1}^{i-1} w_s + w_i + Bw_{r-i}^* \leq \overline{Bw_r^*}$, otherwise $\sum_{s=1}^{i-1} w_s + w_i + Bw_{r-i} \leq \overline{Bw_r^*}$.

For CHAM variants, the differential characteristics with a probability of $P \geq 2^{-n}$ we obtained are listed in Table 8 and Table 9. The details of the differential characteristics are shown in Table 11. Compared to the results given by the authors of CHAM, our results can cover more rounds, shown in Table 10. For CHAM-128/*, we get an interesting observation from the differential characteristics obtained, shown in *Observation 1*.

Observation 1. For CHAM-128/*, let $\Delta X_0^1||\cdots||\Delta X_3^1 \xrightarrow{16} \Delta X_0^{17}||\cdots||\Delta X_3^{17}$ be a 16-round differential trail Υ_1 with a probability of P_1, and $\Delta X_j^{17} = \Delta X_j^1 \lll 4$ for $0 \leq j \leq 3$. Hence, for consecutive $16t$-round reduced CHAM-128/*, there have such a differential trail, i.e. $\Delta X_0^1||\cdots||\Delta X_3^1 \xrightarrow{r=16t} \Delta X_0^{r+1}||\cdots||\Delta X_3^{r+1}$ with a probability of $P = P_1 \times \cdots \times P_t$, $t \geq 1$. Where $P_2, \cdots, P_t$ can be derived from the probability of Υ_1, the input differences of each round of the differential trail can be denoted by $\Delta X_j^i = \Delta X_j^{i \bmod 16} \lll (4\lfloor \frac{i}{16} \rfloor)$, for $0 \leq j \leq 3$ and $i > 16$.

Table 8. The probability weights of the best differential trails for CHAM-64.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Bw_r	0	0	0	0	1	1	2	3	4	5	6	7	8	9	11	14	15	16	19	22
Bw_r^*	0	0	0	0	1	1	2	3	4	5	6	7	8	9	11	13	15	16	18	22
Round	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	
Bw_r	23	26	29	30	32	35	38	39	41	44	46	48	49	51	55	56	58	61	64	
Bw_r^*	23	25	29	31	34	36	38	40	42	45	47	48	50	52	54	57	58	60	64	

Table 9. The probability weights of the best differential trails for CHAM-128/*.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Bw_r	0	0	0	1	1	2	2	3	5	6	7	8	9	11	13	16	17	18	21	24	26	28
Bw_r^*	0	0	0	1	1	2	2	3	5	6	7	8	9	11	13	16	17	18	21	24	26	28
Round	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
Bw_r	31	33	35	39	43	46	48	53	57	61	65	67	70	72	73	75	78	80	81	83	86	87
Bw_r^*	31	34	36	39	43	46	49	51	55	62	64	67	69	72	74	76	78	81	82	83	85	88
Round	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64		
Bw_r	88	90	93	96	97	99	102	104	105	107	110	113	114	116	119	121	122	124	127	130		
Bw_r^*	90	92	94	96	99	100	102	105	107	108	110	113	115	116	118	121	123	125	127	130		

Table 10. Comparison of the differential characteristics on CHAM.

Variants	r	Pr	Δin	Δout	Reference
CHAM-64/128	36	2^{-63}	0004 0408 0A00 0000	0005 8502 0004 0A00	[18]
	39	2^{-64}	0020 0010 1020 2800	1008 0010 2000 1000	This paper.
CHAM-128/*	45	2^{-125}	01028008 08200080 04000040 42040020	00000000 00110004 04089102 00080010	[18]
	63	2^{-127}	80000000 40000000 00408000 00200080	00400010 00008000 00004000 80000040	This paper.

Let $(\Delta X_0^1||\cdots||\Delta X_3^1) = (80000000400000000040800000200080)$, the probabilities of the 16-round differential trails $\Upsilon_1/\Upsilon_2/\Upsilon_3/\Upsilon_4$ are $P_1 = 2^{-32}$, $P_2 = 2^{-33}$, $P_3 = 2^{-31}$, and $P_4 = 2^{-34}$. We can experimentally deduce the probabilities of the additional two 16-round differential trail Υ_5 and Υ_6, where $P_5 = 2^{-33}$, $P_6 = 2^{-32}$. Therefore, for the full round of CHAM-128/128 and CHAM-128/256, we can get the differential characteristics $\Upsilon_1 \to \cdots \to \Upsilon_5$ and $\Upsilon_1 \to \cdots \to \Upsilon_6$ of 80/96-round with probabilities of 2^{-163} and 2^{-195} respectively.

Υ_1 : 80000000400000000040800000200080 → 00000008000000040408000002000800
Υ_2 : 00000008000000040408000002000800 → 00000800000000404080000020008000
Υ_3 : 00000800000000404080000020008000 → 00008000000004000800000400080002
Υ_4 : 00008000000004000800000400080002 → 00080000000040008000004000800020
Υ_5 : 00080000000040008000004000800020 → 000800000000400000000040808000200
Υ_6 : 000800000000400000000040808000200 → 008000000040000000000408080002000

6 Conclusions

In this paper, we revisit the differential properties of modular addition. An algorithm to obtain all input-output difference tuples of specific probability weight, an improved construction of cDDT, and the delicate pruning conditions are proposed. Combining these optimization strategies, we can construct the automatic tool to achieve efficient search for the differential characteristics on ARX ciphers. As appling, more tight differential probabilities for SPECK32/48/64 have been obtained. The differential characteristics obtained for SPARX variants are the best so far, although it does not threaten the claimed security. When considering key recovery attacks on CHAM-128/128 and CHAM-128/256 based on the differential characteristics of CHAM we obtained, and as its authors claimed that

Table 11. The best differential trails for CHAM-64/128 and CHAM-128/*.

39-round trail for CHAM-64/128			64-round trail for CHAM-128/*		
r	$\Delta X_0^r \| \cdots \| \Delta X_3^r$	w_r	r	$\Delta X_0^r \| \cdots \| \Delta X_3^r$	w_r
1	0020 0010 1020 2800	1	1	80000000 40000000 00408000 00200080	0
2	0010 1020 2800 0000	2	2	40000000 00408000 00200080 00000000	2
3	1020 2800 0000 4000	3	3	00408000 00200080 00000000 01000000	3
4	2800 0000 4000 2040	2	4	00200080 00000000 01000000 00810000	2
5	0000 4000 2040 5000	0	5	00000000 01000000 00810000 00400100	1
6	4000 2040 5000 0080	2	6	01000000 00810000 00400100 00000002	1
7	2040 5000 0080 0040	2	7	00810000 00400100 00000002 00000001	3
8	5000 0080 0040 4080	2	8	00400100 00000002 00000001 01020000	3
9	0080 0040 4080 A000	1	9	00000002 00000001 01020000 00800200	1
10	0040 4080 A000 0000	1	10	00000001 01020000 00800200 00000000	2
11	4080 A000 0000 0001	3	11	01020000 00800200 00000000 04000000	3
12	A000 0000 0001 8100	1	12	00800200 00000000 04000000 02040000	2
13	0000 0001 8100 4001	1	13	00000000 04000000 02040000 01000400	1
14	0001 8100 4001 0200	2	14	04000000 02040000 01000400 00000008	2
15	8100 4001 0200 0100	2	15	02040000 01000400 00000008 00000004	3
16	4001 0200 0100 0201	3	16	01000400 00000008 00000004 04080000	3
17	0200 0100 0201 8003	1	17	00000008 00000004 04080000 02000800	1
18	0100 0201 8003 0000	2	18	00000004 04080000 02000800 00000000	2
19	0201 8003 0000 0004	4	19	04080000 02000800 00000000 10000000	3
20	8003 0000 0004 0402	2	20	02000800 00000000 10000000 08100000	2
21	0000 0004 0402 0007	1	21	00000000 10000000 08100000 04001000	1
22	0004 0402 0007 0800	2	22	10000000 08100000 04001000 00000020	2
23	0402 0007 0800 0400	4	23	08100000 04001000 00000020 00000010	3
24	0007 0800 0400 0004	4	24	04001000 00000020 00000010 10200000	3
25	0800 0400 0004 0002	1	25	00000020 00000010 10200000 08002000	1
26	0400 0004 0002 0000	1	26	00000010 10200000 08002000 00000000	2
27	0004 0002 0000 0000	1	27	10200000 08002000 00000000 40000000	3
28	0002 0000 0000 0000	1	28	08002000 00000000 40000000 20400000	2
29	0000 0000 0000 0004	0	29	00000000 40000000 20400000 10004000	0
30	0000 0000 0004 0000	0	30	40000000 20400000 10004000 00000080	2
31	0000 0004 0000 0000	1	31	20400000 10004000 00000080 00000040	3
32	0004 0000 0000 0800	1	32	10004000 00000080 00000040 40800000	3
33	0000 0000 0800 0008	0	33	00000080 00000040 40800000 20008000	1
34	0000 0800 0008 0000	1	34	00000040 40800000 20008000 00000000	1
35	0800 0008 0000 0010	2	35	40800000 20008000 00000000 00000001	3
36	0008 0000 0010 1008	1	36	20008000 00000000 00000001 81000000	2
37	0000 0010 1008 0010	1	37	00000000 00000001 81000000 40010000	1
38	0010 1008 0010 2000	2	38	00000001 81000000 40010000 00000200	2
39	1008 0010 2000 1000	3	39	81000000 40010000 00000200 00000100	2
40	0010 2000 1000 2810	-	40	40010000 00000200 00000100 02000001	3
			41	00000200 00000100 02000001 80020000	1
			42	00000100 02000001 80020000 00000000	2
			43	02000001 80020000 00000000 00000004	3
			44	80020000 00000000 00000004 04000002	1
			45	00000000 00000004 04000002 00040001	1
			46	00000004 04000002 00040001 00000800	2
			47	04000002 00040001 00000800 00000400	3
			48	00040001 00000800 00000400 08000004	3
			49	00000800 00000400 08000004 00080002	1
			50	00000400 08000004 00080002 00000000	2
			51	08000004 00080002 00000000 00000010	3
			52	00080002 00000000 00000010 10000008	2
			53	00000000 00000010 10000008 00100004	1
			54	00000010 10000008 00100004 00002000	2
			55	10000008 00100004 00002000 00001000	3
			56	00100004 00002000 00001000 20000010	3
			57	00002000 00001000 20000010 00200008	1
			58	00001000 20000010 00200008 00000000	2
			59	20000010 00200008 00000000 00000040	3
			60	00200008 00000000 00000040 40000020	2
			61	00000000 00000040 40000020 00400010	1
			62	00000040 40000020 00400010 00008000	2
			63	40000020 00400010 00008000 00004000	3
			64	00400010 00008000 00004000 80000040	3
			65	00008000 00004000 80000040 00800020	-

one can attack at most $4 + 2(k/w - 4) + 3$ rounds more than that of the differential characteristics obtained, therefore, the security margin of CHAM-128/* will be less than 20%. It can be believed that, our tool can also be ultilized to differential cryptanalysis on other ARX-based primitives.

Acknowledgements. The authors will be very grateful to the anonymous reviewers for their insightful comments. And we are especially thankful to Qingju Wang and Vesselin Velichkov for their helpful suggestions. We also appreciate the discussion with Zhengbin Liu and the theoretical supports. This work was supported by the National Key Research and Development Program of China (No. 2017YFB0801900).

A. How to Apply to Other ARX Ciphers

For an iterated ARX cipher, assumming that there are N_A additions modulo 2^n in each round, for example, $N_A = 1/2/4/1$ for SPECK/SPARX-64/SPARX-128/CHAM respectively. And the difference propagation properties of the linear layer between adjacent rounds can also be deduced, for example, as shown in *Property 1/2/3/4*. The following four steps demonstrate how to model the search strategy for the r-round optimal differential trail of an ARX cipher.

Step 1. Pre-compute and store cDDT. Call **Program entry** and gradually increase the expected probability weight $\overline{Bw_r}$.

Step 2. Gradually increasing the probability weights w_i ($1 \leq i \leq r_1$) of each round for the front r_1 rounds. Simultaneously, generating the input-output difference tuples $(\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j})$ for each addition by $Gen(w_{i,j})$. Where $w_{i,j} = 0$ to $n-1$, and $w_i = \sum_{j=1}^{N_A} w_{i,j}$. Make sure all input differences $(\alpha_{r_1+1,j}, \beta_{r_1+1,j})$ of each modular addition in the (r_1+1)-round can be determined after the propagation. For example, $r_1 = 1/1/3$ for SPECK/SPARX/CHAM respectively.

Step 3. In the middle rounds ($r_1 < r_m \leq r$), for each addition, spliting its input differences $(\alpha_{r_m,j}, \beta_{r_m,j})$ into n/m m-bit sub-blocks and verifying the pruning condition (7). Call $Cap(\alpha_{r_m,j}, \beta_{r_m,j})$ for fine-grained pruning, and get the possible $\gamma_{r_m,j}$ and probability weight $w_{r_m,j}$, where $w_{r_m} = \sum_{j=1}^{N_A} w_{r_m,j}$.

Step 4. Iteratively call **Step 3** till the last round. Checking whether the expected probability weight $\overline{Bw_r} = \sum_{s=1}^{r} w_s$ or not. If it is, record the trail and stop, otherwise the execution should continue.

References

1. https://csrc.nist.gov/Projects/Lightweight-Cryptography
2. https://www.cryptolux.org/index.php/Sparkle
3. Ankele, R., Kölbl, S.: Mind the Gap - A Closer Look at the Security of Block Ciphers against Differential Cryptanalysis. In: Cid, C., Jacobson Jr., M.J. (eds.) Selected Areas in Cryptography – SAC 2018. pp. 163–190. Springer, Cham (2019)
4. Ankele, R., List, E.: Differential Cryptanalysis of Round-Reduced SPARX-64/128. In: Applied Cryptography and Network Security, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings. pp. 459–475 (2018)

5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), https://eprint.iacr.org/2013/404
6. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of CRYPTOLOGY **4**(1), 3–72 (1991)
7. Biryukov, A., Perrin, L.: State of the Art in Lightweight Symmetric Cryptography. IACR Cryptology ePrint Archive **2017**, 511 (2017)
8. Biryukov, A., Roy, A., Velichkov, V.: Differential Analysis of Block Ciphers SIMON and SPECK. In: International Workshop on Fast Software Encryption. pp. 546–570. Springer (2014), https://doi.org/10.1007/978-3-662-46706-0_28
9. Biryukov, A., Velichkov, V.: Automatic Search for Differential Trails in ARX Ciphers. In: Benaloh, J. (ed.) Topics in Cryptology – CT-RSA 2014. pp. 227–250. Springer International Publishing, Cham (2014), http://eprint.iacr.org/2013/853
10. Biryukov, A., Velichkov, V., Le Corre, Y.: Automatic Search for the Best Trails in ARX: Application to Block Cipher SPECK. In: International Conference on Fast Software Encryption. pp. 289–310. Springer (2016)
11. Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., Biryukov, A.: Design Strategies for ARX with Provable Bounds: SPARX and LAX. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 484–513. Springer (2016), https://doi.org/10.1007/978-3-662-53887-6_18
12. Ehrlich, G.: Loopless Algorithms for Generating Permutations, Combinations, and Other Combinatorial Configurations. J. ACM **20**(3), 500–513 (1973), https://doi.org/10.1145/321765.321781
13. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In: Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. pp. 268–288 (2016), https://doi.org/10.1007/978-3-662-52993-5_14
14. Hong, D., Lee, J., Kim, D., Kwon, D., Ryu, K.H., Lee, D.: LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors. In: Information Security Applications - 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers. pp. 3–27 (2013)
15. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.S., Lee, C., Chang, D., Lee, J., Jeong, K., et al.: HIGHT: A New Block Cipher Suitable for Low-resource Device. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 46–59. Springer (2006), https://doi.org/10.1007/11894063_4
16. Huang, M., Wang, L., Zhang, Y.: Improved Automatic Search Algorithm for Differential and Linear Cryptanalysis on SIMECK and the Applications. In: Information and Communications Security - 20th International Conference, ICICS 2018, Lille, France, October 29-31, 2018, Proceedings. pp. 664–681 (2018)
17. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON Block Cipher Family. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology – CRYPTO 2015. pp. 161–185. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
18. Koo, B., Roh, D., Kim, H., Jung, Y., Lee, D., Kwon, D.: CHAM: A family of lightweight block ciphers for resource-constrained devices. In: Information Security and Cryptology - ICISC 2017 - 20th International Conference, Seoul, South Korea, November 29 - December 1, 2017, Revised Selected Papers. pp. 3–25 (2017)
19. Lai, X., Massey, J.L., Murphy, S.: Markov Ciphers and Differential Cryptanalysis. In: Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings. pp. 17–38 (1991), https://doi.org/10.1007/3-540-46416-6_2

20. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama,Japan, April 2-4, 2001, Revised Papers. pp. 336–350 (2001)
21. Lipmaa, H., Wallén, J., Dumas, P.: On the Additive Differential Probability of Exclusive-Or. In: Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers. pp. 317–331 (2004)
22. Liu, Y., Wang, Q., Rijmen, V.: Automatic Search of Linear Trails in ARX with Applications to SPECK and Chaskey. In: Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings. pp. 485–499 (2016), https://doi.org/10.1007/978-3-319-39555-5_26
23. Liu, Z.: Automatic Tools for Differential and Linear Cryptanalysis of ARX Ciphers. University of Chinese Academy of Science, Phd Thesis. **In Chinese** (2017)
24. Liu, Z., Li, Y., Jiao, L., Wang, M.: A new method for searching optimal differential and linear trails in arx ciphers. Cryptology ePrint Archive, Report 2019/1438 (2019), https://eprint.iacr.org/2019/1438
25. Liu, Z., Li, Y., Wang, M.: Optimal Differential Trails in SIMON-like Ciphers. IACR Trans. Symmetric Cryptol. **2017**(1), 358–379 (2017)
26. Liu, Z., Li, Y., Wang, M.: The Security of SIMON-like Ciphers Against Linear Cryptanalysis. IACR Cryptology ePrint Archive **2017**, 576 (2017)
27. Matsui, M.: On Correlation Between the Order of S-boxes and the Strength of DES. In: De Santis, A. (ed.) Advances in Cryptology — EUROCRYPT'94. pp. 366–375. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
28. Mouha, N., Preneel, B.: Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Report 2013/328 (2013)
29. Mouha, N., Velichkov, V., Cannière, C.D., Preneel, B.: The Differential Analysis of S-Functions. In: Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers. pp. 36–56 (2010), https://doi.org/10.1007/978-3-642-19574-7_3
30. Song, L., Huang, Z., Yang, Q.: Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA. In: Australasian Conference on Information Security and Privacy. pp. 379–394. Springer (2016)
31. Sun, L., Wang, W., Wang, M.: Automatic Search of Bit-Based Division Property for ARX Ciphers and Word-Based Division Property. In: Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. pp. 128–157 (2017), https://doi.org/10.1007/978-3-319-70694-8_5
32. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The Simeck Family of Lightweight Block Ciphers. In: Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings. pp. 307–329 (2015), https://eprint.iacr.org/2015/612
33. Yin, J., Ma, C., Lyu, L., Song, J., Zeng, G., Ma, C., Wei, F.: Improved Cryptanalysis of an ISO Standard Lightweight Block Cipher with Refined MILP Modelling. In: Information Security and Cryptology - 13th International Conference, Inscrypt 2017, Xi'an, China, November 3-5, 2017, Revised Selected Papers. pp. 404–426 (2017), https://doi.org/10.1007/978-3-319-75160-3_24
34. Zhang, Y., Sun, S., Cai, J., Hu, L.: Speeding up MILP Aided Differential Characteristic Search with Matsui's Strategy. In: Information Security - 21st International Conference, ISC 2018, Guildford, UK, September 9-12, 2018, Proceedings. pp. 101–115 (2018), https://doi.org/10.1007/978-3-319-99136-8_6

35. Zhou, C., Zhang, W., Ding, T., Xiang, Z.: Improving the MILP-based Security Evaluation Algorithms against Differential Cryptanalysis Using Divide-and-Conquer Approach. IACR Cryptology ePrint Archive **2019**, 19 (2019)