Automatic Search for the Linear (hull) Characteristics of ARX Ciphers: Applied to SPECK, SPARX, Chaskey and CHAM-64

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

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

Abstract. Linear cryptanalysis is an important evaluation method for cryptographic primitives against key recovery attack. In this paper, we revisit the Walsh transformation for linear correlation calculation of modular addition, and an efficient algorithm is proposed to construct the input-output mask space of specified correlation weight. By filtering out the impossible large correlation weights in the first round, the search space of the first round can be substantially reduced. We introduce a concept of combinational linear approximation table (cLAT) for modular addition with two inputs. When one input mask is fixed, another input mask and the output mask can be obtained by the *Spliting-Lookup-Recombination* approach. We first split the n-bit fixed input mask into several sub-vectors, then, to find the corresponding bits of other masks, and in the recombination phase, pruning conditions can be used. By this approach, a large number of search branches in the middle rounds can be pruned. With the combination of the optimization strategies and the branch-and-bound search algorithm, we can improve the search efficiency for linear characteristics on ARX ciphers. The linear hulls for SPECK32/48/64 with higher average linear potential (ALP) than existing results have been obtained. For SPARX variants, a 11-round linear trail and a 10-round linear hull have been found for SPARX-64, a 10-round linear trail and a 9-round linear hull are obtained for SPARX-128. For Chaskey, a 5-round linear trail with correlation of 2^{-61} have been obtained. For CHAM-64, the 34/35-round optimal linear characteristics with correlation of $2^{-31}/2^{-33}$ are found.

Keywords: SPECK · SPARX · ARX · Linear cryptanalysis· Linear Hull · Automatic search · Block cipher

1 Introduction

The three components: Modular Addition, Rotation, XOR, constitute the basic operations in ARX cryptographic primitives [2]. In ARX ciphers, modular additions provide non-linearity diffusion with efficient software implementation and low dependencies on computing resources. Compared with S-box based ciphers, ARX ciphers do not need to store S-box in advance, which can reduce the occupation of storage resources, especially in resource-constrained devices. In addition, ARX ciphers do not need to query S-boxes in the encryption and decryption process, which can reduce a lot of query operations. Therefore, ARX construction is prefered by many designers of lightweight ciphers. At present, there are many primitives used this construction, such as HIGHT [10], SPECK [1], LEA [9], Chaskey [21], SPARX [6], CHAM [11] et al..

Until now, cryptanalysis on ARX ciphers is still not well understood as S-box based ciphers, the security analysis on them are relatively lagging behind [25]. Linear cryptanalysis is very important for evaluating the security margin of symmetric cryptographic primitives [17,18]. The linear approximation tables of S-box based ciphers mostly can be constructed and stored directly, however, the full linear approximation table of modular addition will be too large to store when the word length of modular addition is large.

For linear cryptanalysis of ARX ciphers, one crucial step is to calculate the linear correlation of modular addition. In [12,23,26,27], the linear properties of the modular addition have been carefully studied. In [26], a method to calculate the linear correlation of modular addition recursively was proposed, but the calculation process that based on the state transition in bit level leads to high complexity. Based on this method, only the optimal linear characterstics for the variants of SPECK32 [28] and SPECK32/48 [8] were found.

In 2013, Schulte-Geers used CCZ-equivalence to improve the explicit formula for the calculation of linear correlation of modular addition [24]. Based on the improved formula and SAT solver model, Liu et al. obtained better linear characteristics for SPECK [14], the optimal linear trails for SPECK32/48/64 with correlation close to the security boundary $(2^{-\frac{n}{2}})$ were obtained, and the 9/10 round linear hull with potential of $2^{-29.1}/2^{-32.1}$ for SPECK32 were obtained.

According to the position of the starting round of the search algorithm, there are currently 3 types of automatic search technologies for linear/differential cryptanalysis on ARX primitives. These are the bottom-up techniques [28], top-down techniques [3,15,16], and the method of extending from the middle to the ends [19]. In these methods, the linear correlations are directly calculated based on the input-output masks, or by looking up the pre-computed partial linear approximation table (pLAT) [13]. For addition modulo 2^n with two inputs, the correlations need to be calculated based on the known input-output masks. However, in the search process for linear characteristics of ARX cipers, due to the existence of three-forked branches, in most case, for the input-output masks $((v, w) \to u)$ of modular addition, only one input mask v is determined, another input mask w and the output mask u are unknown. Although, all 2^{2n} space of (w, u) can be traversed in a trivial way, it's very time consuming.

High efficiency query operations can be achieved by constructing a linear approximation table of reasonable storage size. The pLAT can store the input-output masks whose linear correlation are greater than a certain threshold [28]. When the branches cannot be queried in pLAT, and that need to be calculated by the input-output masks, the calculation process will lead to a significant reduction in search efficiency. Although heuristic method can speed up the search, it can not guarantee the results will be the best [4].

Therefore, constructing a search model based on the precise correlation calculation formula, and realizing an efficient search for linear characteristics on ARX ciphers is still a study worth working on. The motivation of this paper is to investigate how to speed up the search algorithm in order to realize the search for linear (hull) characteristics on typical ARX ciphers.

Our Contributions. In this paper, we first revisit the linear correlation calculation of modular addition, and introduce an algorithm to construct the input-out masks of specific correlation weight. Then, we propose a novel concept of combinational linear approximation table (cLAT), and introduce an algorithm to generate the lookup tables. Combining with these two optimization algorithms, we propose an automatic algorithm to search for the optimal linear characteristics on ARX ciphers. In the first round, we can exclude the search space of the non-optimal linear trails by increasing the correlation weight of each modular addition monotonically. In the middle rounds, the undetermined masks and the corresponding correlation weight of each modular addition can be obtained by querying the cLAT, and a large number of non-optimal branches can be filtered out during the recombination phase. Also, the algorithm can be appropriately modified for the heuristic search.

As applications, for SPECK32/48/64, the 9/11/14-round linear hulls are obtained. For SPARX-64, the 11-round linear trail with correlation of 2^{-28}, and a 10-round linear hull with ALP of $2^{-40.92}$ are found. For SPARX-128, we can experimentally get the optimal linear trails of the first eight rounds, and we get a 10-round linear trail with correlation of 2^{-23}. For Chaskey, the linear characteristics cover more rounds are updated, and a 5-round linear trail with correlation of 2^{-61} is found. For CHAM-64, we find a new 34-round optimal linear trail with correlation of 2^{-31}. A summary table is shown in Table 1.

Roadmap. This paper is organized as follows. We first present some preliminaries used in this paper in Section 2. In Section 3, we introduce the algorithm for constructing the space of input-output mask tuples, the algorithm for constructing cLAT, and the improved automatic search algorithm for linear cryptanalysis on ARX ciphers. In Section 4, we apply the new tool to several typical ARX ciphers. Finally, we conclude our work in Section 5.

2 Preliminaries

2.1 Notation

For additon modulo 2^n, i.e. $x \boxplus y = z$, 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 $\oplus$, $\vee$, $\wedge$, $||$, $\neg$ represent XOR, OR, AND, concatenation, and bitwise NOT respectively. For a vector x, $wt(x)$ represents its Hamming weight, x_i is the i^{th} bit of it. $\mathbf{0}$ is a zero vector.

Table 1. Summary of the linear characteristics on SPECK, SPARX, Chaskey and CHAM-64, where ‘s’,‘m’,‘h’,‘d’ represent seconds, minutes, hours, and days respectively.

Variants	Round	Cor	T_{Cor}	ALP	T_{ALP}	Reference
SPECK32	9	2^{-14}	N/A	$2^{-29.1}$	N/A	[14]
	9	2^{-14}	N/A	2^{-28}	N/A	[8]
	9	2^{-14}	9s	$2^{-27.78}$	25s	This paper.
SPECK48	10	2^{-22}	N/A	2^{-44}	N/A	[14]
	10	2^{-22}	N/A	2^{-44}	N/A	[8]
	10	2^{-22}	3.2h	$2^{-43.64}$	157.3h	This paper.
SPECK64	13	2^{-30}	N/A	2^{-60}	N/A	[14]
	13	2^{-30}	N/A	2^{-60}	N/A	[8]
	13	2^{-30}	8.6h	$2^{-55.29}$	7.3h	This paper.
	14	2^{-33}	25.6h	$2^{-61.24}$	5.8h	This paper.
SPARX-64	10	2^{-22}	3d	$2^{-40.92}$	1h	This paper.
	11	2^{-28}	5m	2^{-56}	-	This paper.
SPARX-128	9	2^{-18}	27m	$2^{-35.22}$	6h	This paper.
	10	2^{-23}	4.4d	2^{-46}	-	This paper.
Chaskey	3	2^{-9}	N/A	2^{-18}	N/A	[14]
	4	2^{-29}	15.7m	2^{-58}	-	This paper.
	5	2^{-61}	6.6h	2^{-122}	-	This paper.
CHAM-64	34	2^{-31}	N/A	2^{-62}	N/A	[11]
	34	2^{-31}	1.1d	2^{-62}	-	This paper.
	35	2^{-33}	4.8d	-	-	This paper.

2.2 Linear Correlation Calculation for Modular Addition

Let $\mathbb{F}_2^n$ be the n dimensional vector space over binary field $\mathbb{F}_2 = \{0, 1\}$, for boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and $h : \mathbb{F}_2^n \to \mathbb{F}_2$, $x \in \mathbb{F}_2^n$, the linear correlation between f and h can be denoted by

$$Cor(f, h) = 2 \times \frac{\#\{x|f(x) \oplus h(x) = 0\}}{2^n} - 1. \tag{1}$$

For modular additon $x \boxplus y = z$, let (v, w) be the input masks, u be the output mask, and $\cdot$ be the standard inner product. According to the definition of linear correlation, when $(v \cdot x \boxplus w \cdot y) \oplus u \cdot z = 0$, the linear approximation probability is defined as

$$\Pr(u, v, w) = 2^{-3n} \times \#\{(x, y, z)|(v \cdot x \boxplus w \cdot y) \oplus u \cdot z = 0\}. \tag{2}$$

Let $\mu(t) = (-1)^t$, then, the linear correlation of modular addition can be denoted by the Walsh transformation, there have

$$Cor(u, v, w) = 2^{-3n} \times \sum_{x,y,z \in \mathbb{F}_2^n} \mu((v \cdot x \boxplus w \cdot y) \oplus u \cdot z). \tag{3}$$

Let $\Pr(u, v, w) = \frac{1}{2} + \varepsilon$, where ε is the bias. When $(v \cdot x \boxplus w \cdot y) \oplus u \cdot z = 1$, the linear approximation probability is $\overline{\Pr}(u, v, w) = \frac{1}{2} - \varepsilon$. The linear correlation can be denoted by

$$Cor(u, v, w) = \Pr(u, v, w) - \overline{\Pr}(u, v, w) = 2\Pr(u, v, w) - 1. \tag{4}$$

We call $Cw(u,v,w) = -\log_2 Cor(u,v,w)$ as the *correlation weight*, the linear square correlation can be denoted by

$$LSC(u,v,w) = Cor(u,v,w)^2 = 2^{-2\times Cw(u,v,w)}. \tag{5}$$

For addition $x \boxplus y$ modulo 2^n, it can be rewritten as $x \boxplus y = x \oplus y \oplus carry(x,y)$, in which $carry(x,y)_{i+1} = carry(x,y)_i \oplus x_i \oplus y_i$, and $carry(x,y)_0 = 0$ for $0 \leq i \leq n-1$. The first order approximation is $carry(x,y) = (x \wedge y) \ll 1$. If all $carry(x,y)_j = 0$ for $j \leq i$, and $0 \leq i \leq n-1$, the high order approximation is

$$carry(x,y)_{i+1} = \frac{1}{2}|(-1)^{x_i} + (-1)^{y_i} + carry(x,y)_i - (-1)^{x_i+y_i} carry(x,y)_i|$$

In [26], Wallén introduced the theorem to calculate the linear correlation by analyzing the carry high order approximation function recursively. In [23], based on the bit state transformation, the formula to calculate the correlation was given by the following theorem.

Theorem 1 ([23]). *For addition modulo 2^n, let v, w be the input masks and u be the output mask. Define an auxiliary vector $d = d_{n-1} \cdots d_0$, each $d_i = u_i || v_i || w_i \in \mathbb{F}_2^3$ is an octal word, $0 \leq i \leq n-1$. Then, the linear correlation can be denoted by*

$$Cor(u,v,w) = LA_{d_{n-1}} A_{d_{n-2}} \cdots A_{d_1} A_{d_0} C.$$

Where the row vector $L = (1\ 0)$, the column vector $C = (1\ 1)^T$, and each 2×2 matrice A_{d_i} is defined by

$$A_0 = \frac{1}{2}\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, A_1 = A_2 = -A_4 = \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix},$$

$$A_7 = \frac{1}{2}\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}, -A_3 = A_5 = A_6 = \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

In [24], Schulte-Geers extended Theorem 1 and derived a fully explicit formula for the linear correlation calculation, given by Theorem 2.

Theorem 2 ([24]). *For addition modulo 2^n with input-output mask tuple (u,v,w), a vectorial boolean function $M : \mathbb{F}_2^n \to \mathbb{F}_2^n$ denotes the partial sums mapping,*

$$\boldsymbol{x} = (x_0, x_1, \cdots, x_{n-1}) \to M(\boldsymbol{x}) = (0, x_0, x_0 \oplus x_1, \cdots, x_0 \oplus x_1 \oplus \cdots x_{n-2}).$$

Let $\boldsymbol{z} := M^T(u \oplus v \oplus w) = (0, x_{n-1}, x_{n-1} \oplus x_{n-2}, \cdots, x_{n-1} \oplus x_{n-2} \oplus \cdots x_1)$, then, the linear correlation can be denoted by

$$Cor(u,v,w) = 1_{\{u \oplus v \preceq \boldsymbol{z}\}} 1_{\{u \oplus w \preceq \boldsymbol{z}\}} (-1)^{v \cdot w} 2^{-wt(\boldsymbol{z})},$$

where 1_{G_f} is an indicator function for graph $G_f := \{(x, f(x)) | x \in \mathbb{F}_2^n\}$, for n-bit vectors a and b, $a \preceq b$ represents $a_i \leq b_i$ for $0 \leq i \leq n-1$.

In iterative ciphers, the correlation of a single r-round linear trail is the product of the correlations of each round [5]. Assuming that there are N_A additions modulo 2^n with two inputs in i^{th} round, Γ_{in}, Γ_{out} are the input and output mask of the r-round linear trail, the correlation of it can be denoted by

$$Cor(\Gamma_{in}, \Gamma_{out}) = \prod_{i=1}^{r} \prod_{j=1}^{N_A} Cor(u_{i,j}, v_{i,j}, w_{i,j}). \tag{6}$$

The linear approximation of a linear hull represents the potential of all linear trails with same input-output masks [22]. The averaged linear potential (ALP) can be counted by the following formula (7).

$$ALP(\Gamma_{in}, \Gamma_{out}) = \frac{1}{|K|} \sum_{k \in K} Cor(\Gamma_{in}, \Gamma_{out})^2. \tag{7}$$

Assuming that the key k is selected uniformly from the key space K, the statistics of ALP can be formulated as (8), where $T[Cw]$ is the number of trails with *correlation weight* of Cw. Let C_{min} be the *correlation weight* of the linear trail whose input-output masks are choosen as the fixed input-output masks of the linear hull. C_{max} is the upper bound be searched, which should be choosen by the trade-off between the search time and the accuracy of ALP.

$$ALP(\Gamma_{in}, \Gamma_{out}) = \sum_{Cw=C_{min}}^{C_{max}} 2^{-2Cw} \times T[Cw]. \tag{8}$$

2.3 Linear Properties of SPECK, SPARX, Chaskey and CHAM

The SPECK family ciphers were designed by NSA in 2013 [1]. The SPARX family ciphers were introduced by Dinu et al. at ASIACRYPT'16 [6]. In SPARX, the non-linear ARX-box (SPECKEY) is obtained by modifying the round function of SPECK32. The linear mask propagation properties of the round function in SPECK and SPECKEY are shown in Fig. 1. The rotation parameters $(r_a, r_b) = (7, 2)$ for SPECK32, while $(r_a, r_b) = (8, 3)$ for other variants.

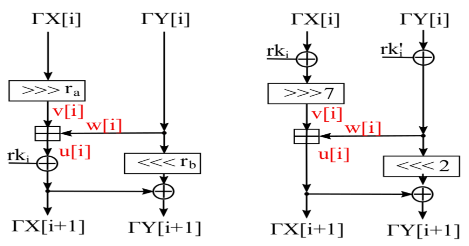

Fig. 1. The linear masks propagation properties of SPECK and SPECKEY.

If the input-output masks $(u[i], v[i], w[i])$ and $(u[i+1], v[i+1], w[i+1])$ of the modular additions in the two consecutive rounds of SPECK are known, the input and output masks of these two rounds can be denoted by *Property 1*.

Property 1. If $(u[i], v[i], w[i])$ and $(u[i+1], v[i+1], w[i+1])$ are given, then there have $\Gamma X[i] = v[i] \lll r_a$, $\Gamma X[i+1] = v[i+1] \lll r_a$, $\Gamma Y[i+1] = (v[i+1] \lll r_a) \oplus u[i]$, $\Gamma Y[i] = (\Gamma Y[i+1] \ggg r_b) \oplus w[i]$, $\Gamma Y[i+2] = (\Gamma Y[i+1] \oplus w[i+1]) \lll r_b$, and $\Gamma X[i+2] = \Gamma Y[i+1] \oplus u[i+1]$.

The linear layer functions $\mathcal{L}/\mathcal{L}'$ [6] for SPARX-64 and SPARX-128 are shown in Fig. 2. Due to the existence of the three-forked branches, the masks of the linear transformation layer have the following properties.

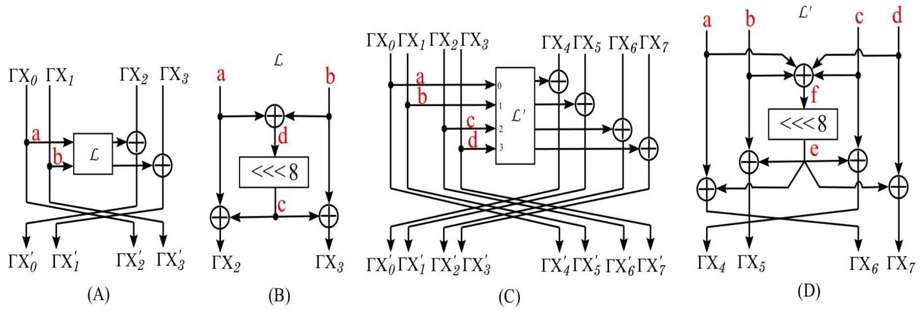

Fig. 2. (A) and (B) represent the linear layer of SPARX-64, (C) and (D) represent the linear layer of SPARX-128.

Property 2. For SPARX-64, if the masks are transformed by the linear layer function $\mathcal{L}$, let $c = \Gamma X_2 \oplus \Gamma X_3$, $d = c \ggg 8$, there have $\Gamma X'_0 = \Gamma X_2$, $\Gamma X'_1 = \Gamma X_3$, $\Gamma X'_2 = \Gamma X_0 \oplus d \oplus \Gamma X_2$, and $\Gamma X'_3 = \Gamma X_1 \oplus d \oplus \Gamma X_3$.

Property 3. For SPARX-128, if the masks are transformed by the linear layer function $\mathcal{L}'$, let $e = \Gamma X_4 \oplus \Gamma X_5 \oplus \Gamma X_6 \oplus \Gamma X_7$, $f = e \ggg 8$, there have $\Gamma X'_0 = \Gamma X_4$, $\Gamma X'_1 = \Gamma X_5$, $\Gamma X'_2 = \Gamma X_6$, $\Gamma X'_3 = \Gamma X_7$, $\Gamma X'_4 = \Gamma X_0 \oplus f \oplus \Gamma X_6$, $\Gamma X'_5 = \Gamma X_1 \oplus f \oplus \Gamma X_5$, $\Gamma X'_6 = \Gamma X_2 \oplus f \oplus \Gamma X_4$, and $\Gamma X'_7 = \Gamma X_3 \oplus f \oplus \Gamma X_7$.

Chaskey is a MAC algorithm introduced by Mouha et al at SAC'14 [21], and an enhanced variant was proposed in 2015 [20], which increases the number of permutation rounds from 8 to 12. The round function of the permutation $(v0', v1', v2', v3') = \pi(v0, v1, v2, v3)$ is shown in Fig.3. The 4 modular additions are labeled by $A0, A1, A2, A3$ respectively. The input mask (a, b, c, d) and the output mask (a', b', c', d') of the first round can be denoted by *Property 4*.

Property 4. For the permutation of Chaskey, if the input-output masks of each modular addition in the first round are $(u[i], v[i], w[i])$, $0 \le i \le 3$, and the corresponding correlation weight of each modular addition are c_0, c_1, c_2, c_3 respectively. Hence, in the first round, there have $a = v[0]$, $b = w[0] \oplus ((u[0] \oplus (v[3] \ggg$

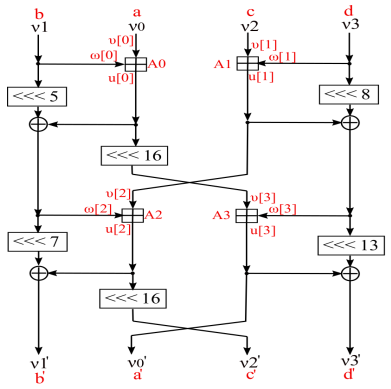

Fig. 3. The linear masks in the first round of Chaskey.

16))) $\ggg 5$, $c = v[1]$, $d = w[1] \oplus (u[1] \oplus v[2]) \ggg 8$, $a' = u[3] \oplus (u[1] \oplus v[2] \oplus w[3]) \lll 13$, $b' = (u[0] \oplus w[2] \oplus w[3] \ggg 16) \lll 7$, $c' = u[2] \oplus (w[2] \oplus (u[0] \oplus v[3]) \ggg 16) \lll 7$, and $d' = (u[1] \oplus v[2] \oplus w[3]) \lll 13$. The corresponding correlation weight of the round function is $Cw = \sum_{i=0}^{3} c_i$.

CHAM is a family of lightweight block ciphers that proposed by Koo et al. at ICISC'17, which blends the good designs of SIMON and SPECK [11]. The 3 variants of CHAM have two kinds of block size, i.e. CHAM-64 and CHAM-128. The linear mask propagation for the 4 consecutive rounds of CHAM is shown in Fig.4. If the input-output mask tuples of each modular addition of the first 4 rounds are given, the input and output masks of the first 4 rounds can be deduced by *Property 5*.

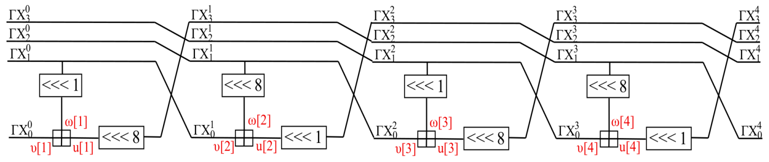

Fig. 4. The linear masks in the first 4 rounds of CHAM.

Property 5. For CHAM, if the input-output mask tuples $(u[i], v[i], w[i])$ of each modular addition of the first 4 rounds are given, $1 \le i \le 4$, the input and output masks of the first 4 rounds can be deduced by follows. $\Gamma X_0^0 = v[1]$, $\Gamma X_1^0 = (w[1] \lll 1) \oplus v[2]$, $\Gamma X_2^0 = (w[2] \lll 8) \oplus v[3]$, $\Gamma X_3^0 = (w[3] \lll 1) \oplus v[4]$; $\Gamma X_0^1 = v[2]$, $\Gamma X_1^1 = \Gamma X_2^0$, $\Gamma X_2^1 = \Gamma X_3^0$, $\Gamma X_3^1 = u[1] \lll 8$; $\Gamma X_0^2 = v[3]$, $\Gamma X_1^2 = \Gamma X_2^1$, $\Gamma X_2^2 = \Gamma X_3^1$, $\Gamma X_3^2 = u[2] \lll 1$; $\Gamma X_0^3 = v[4]$, $\Gamma X_1^3 = \Gamma X_2^2$, $\Gamma X_2^3 = \Gamma X_3^2$, $\Gamma X_3^3 = u[3] \lll 8$; $\Gamma X_0^4 = (w[4] \lll 8) \oplus \Gamma X_1^3$, $\Gamma X_1^4 = \Gamma X_2^3$, $\Gamma X_2^4 = \Gamma X_3^3$, $\Gamma X_3^4 = u[4] \lll 1$.

3 Automatic Search for the linear Characteristics on ARX Ciphers

3.1 Input-Output Masks of Specific Correlation Weight

The number of input-output mask tuples in the first round is closely related to the complexity of the branch-and-bound search algorithm, but traversing all possible input masks of the first round will result in high complexity. An alternative approach is to consider the possible correlation weight corresponding to the input-output masks, and exclude those tuples that have a large correlation weight. However, for a fixed correlation weight, it may correspond to multiple input-output mask tuples, although the correlation can be calculated by Theorem 2 when the input-output masks are fixed for a modular addition.

For addition modulo 2^n, its maximum correlation weight is $n-1$, and the size of the total space S of all input-output mask tuples is 2^{3n}. We can rank the correlation weights Cw from 0 to $n-1$, and construct the input-output masks subspace S_{Cw} corresponding to correlation weight Cw, $0 \le Cw \le n-1$. Therefore, the total space S can be divided into n subspaces, i.e $S = \bigcup_{Cw=0}^{n-1} S_{Cw}$.

Definition 1. *Let $((v,w) \rightarrow u)$ be the input-output masks for a modular addition with non-zero correlation. Let's define an octal word sequence $\Phi := \{\xi_{n-1} \cdots \xi_0\}$, where $\xi_i = u_i||v_i||w_i \in \mathbb{F}_2^3$, for $0 \le i \le n-1$.*

Definition 2. *Let's define three sets that ξ_i may belongs to, i.e. $U_0 = \{1,2,4,7\}$, $U_1 = \{0,3,5,6\}$, $U_2 = \{0,7\}$.*

In Theorem 2, when the the correlation of a modular additon is non-zero, the value distribution of the 3 consecutive bits in $\boldsymbol{z}$ and the 3 consecutive words in Φ have following relationships, shown in *Observation 1.*

Observation 1. Let $\boldsymbol{x} = u \oplus v \oplus w$ and $\boldsymbol{z}_i = \bigoplus_{j=i+1}^{n-1} \boldsymbol{x}_j$ for $0 \le i \le n-2$, $\boldsymbol{z}_{n-1} = 0$, hence, $\boldsymbol{z}_i = \boldsymbol{z}_{i+1} \oplus x_{i+1}$. For $\boldsymbol{z} \in \mathbb{F}_2^n$, assuming when $\boldsymbol{z}_j = 0$ for $n-1 \ge j > i+1$, or $\boldsymbol{z}_{i+2} = 0$, there should have $u_i \oplus v_i \preceq \boldsymbol{z}_i$, $u_i \oplus w_i \preceq \boldsymbol{z}_i$ on bit level, it's equivalent to $u_{i-1} \oplus v_{i-1} \preceq x_{i+1} \oplus x_i$ and $u_{i-1} \oplus w_{i-1} \preceq x_{i+1} \oplus x_i$. Since $x_i = 1$ when $\xi_i \in U_0$, and $x_i = 0$ when $\xi_i \in U_1$, there have,
if $(\boldsymbol{z}_{i+1}, \boldsymbol{z}_i, \boldsymbol{z}_{i-1}) = (0,0,0)$, then $\xi_{i+1} = 0$, $\xi_i = 0$, $\xi_{i-1} = 0$;
if $(\boldsymbol{z}_{i+1}, \boldsymbol{z}_i, \boldsymbol{z}_{i-1}) = (0,0,1)$, then $\xi_{i+1} = 0$, $\xi_i = 0$, $\xi_{i-1} = 7$;
if $(\boldsymbol{z}_{i+1}, \boldsymbol{z}_i, \boldsymbol{z}_{i-1}) = (0,1,0)$, then $\xi_{i+1} = 0$, $\xi_i = 7$, $\xi_{i-1} \in U_0$;
if $(\boldsymbol{z}_{i+1}, \boldsymbol{z}_i, \boldsymbol{z}_{i-1}) = (0,1,1)$, then $\xi_{i+1} = 0$, $\xi_i = 7$, $\xi_{i-1} \in U_1$;
if $(\boldsymbol{z}_{i+1}, \boldsymbol{z}_i, \boldsymbol{z}_{i-1}) = (1,0,0)$, then $\xi_{i+1} = 7$, $\xi_i \in U_0$, $\xi_{i-1} = 0$;
if $(\boldsymbol{z}_{i+1}, \boldsymbol{z}_i, \boldsymbol{z}_{i-1}) = (1,0,1)$, then $\xi_{i+1} = 7$, $\xi_i \in U_0$, $\xi_{i-1} = 7$;
if $(\boldsymbol{z}_{i+1}, \boldsymbol{z}_i, \boldsymbol{z}_{i-1}) = (1,1,0)$, then $\xi_{i+1} = 7$, $\xi_i \in U_1$, $\xi_{i-1} \in U_0$;
if $(\boldsymbol{z}_{i+1}, \boldsymbol{z}_i, \boldsymbol{z}_{i-1}) = (1,1,1)$, then $\xi_{i+1} = 7$, $\xi_i \in U_1$, $\xi_{i-1} \in U_1$.

Hence, the value of $\xi_i = u_i||v_i||w_i$ depends on whether the bit positions of $\boldsymbol{z}_{i+1}$, $\boldsymbol{z}_i$ are active. The last significant bits (u_0, v_0, w_0) of the input-output masks construct the value of ξ_0, which is only related to the Hamming weight of $\boldsymbol{z}_0 = \bigoplus_{j=1}^{n-1} \boldsymbol{x}_j$, i.e. $u_0 \oplus v_0 \preceq wt(\boldsymbol{z}_0)$ and $u_0 \oplus w_0 \preceq wt(\boldsymbol{z}_0)$. Therefore, if we get the Hamming weight distribution of $\boldsymbol{z}$, from the LSB to MSB direction, as $\boldsymbol{z}_0$ is determined, ξ_0 can be obtained. Next, $\boldsymbol{x}_1 = \boldsymbol{z}_1 \oplus \boldsymbol{z}_0$ is determined, and $u_1 \oplus v_1 \preceq wt(\boldsymbol{z}_1)$, $u_1 \oplus w_1 \preceq wt(\boldsymbol{z}_1)$ should be satisfied, hence, the possible values of ξ_1 can be obtained. Recursively, all ξ_i can be constructed as an octal word sequence from the LSB to MSB direction to subject to the above observation. Hence, the tuples of (u, v, w) can be generated from the elements in Φ. The process to construt the subspace S_{Cw} is shown in Algorithm 1, marked as **Const**(S_{Cw}).

Algorithm 1 Const(S_{Cw}): Constructing the input-output mask truples with linear correlation weight of Cw for modular addition, $0 \le Cw \le n-1$.

```
Input: $Cw$ and $\Lambda = \{\lambda_{Cw}, ..., \lambda_1\}$. Each pattern of the Hamming weight distribution of $\boldsymbol{z}$ can be
    calculated by the combinations algorithm in [7], which is the combinations pattern of $\binom{n-1}{Cw}$,
    where $\xi_i = u_i||v_i||w_i$ for $0 \le i \le n-1$.
1: Func_LSB: $i = 0$.  //Constructing the LSBs of $u, v, w$.
2: if $Cw = 0$ then
3:     Output the tuple of $(u, v, w)$ with (1,1,1) or (0,0,0);
4: end if
5: if $\lambda_1 \ne 0$ then
6:     For each $\xi_i \in U_2$, $c = 1$, $Fw = 0$, call Func_Middle$(i+1, c, Fw)$;
7: else
8:     For each $\xi_i \in \mathbb{F}_2^3$, $c = 2$, $Fw = 1$, call Func_Middle$(i+1, c, Fw)$;
9: end if
10: Func_Middle$(i, c, Fw)$:  //Constructing the middle bits of $u, v, w$.
11: if $c = Cw$ then
12:     call Func_MSB$(i, c, Fw)$;
13: end if
14: if $\lambda_c \ne i$ then
15:     if $Fw = 0$ then  //$Fw$ recorded whether the value of $\lambda_{i-1}$ is 1 or not.
16:         For each $\xi_i = 0$, $Fw' = 0$, call Func_Middle$(i+1, c, Fw')$;
17:     else
18:         For each $\xi_i = 7$, $Fw' = 0$, call Func_Middle$(i+1, c, Fw')$;
19:     end if
20: else //$\lambda_c = i$. The value of $Fw$ determines whether $\xi_i$ belongs to $U_0$ or $U_1$.
21:     if $Fw = 0$ then
22:         For each $\xi_i \in U_0$, $Fw' = 1$, call Func_Middle$(i+1, c+1, Fw')$;
23:     else
24:         For each $\xi_i \in U_1$, $Fw' = 1$, call Func_Middle$(i+1, c+1, Fw')$;
25:     end if
26: end if
27: Func_MSB$(i, c, Fw)$:  //Constructing the bits of $u, v, w$ with position higher than $\lambda_{Cw}$.
28: if $\lambda_c \ne i$ then  //The value of $Fw$ determines whether $\xi_i$ equals to 0 or 7.
29:     if $Fw = 0$ then
30:         Let $\xi_i = 0$, $Fw' = 0$, call Func_MSB$(i+1, c, Fw')$;
31:     else
32:         Let $\xi_i = 7$, $Fw' = 0$, call Func_MSB$(i+1, c, Fw')$;
33:     end if
34: else //$\lambda_c = i$.
35:     if $Fw = 0$ then
36:         For each $\xi_i \in U_0$, $\xi_{i+1} = 7$, output each tuple of $(u, v, w)$;
37:     else
38:         For each $\xi_i \in U_1$, $\xi_{i+1} = 7$, output each tuple of $(u, v, w)$;
39:     end if
40: end if
```

3.2 The Combinational Linear Approximation Table

For addition modulo 2^n, the full LAT requires a storage size of 2^{3n}, when n is too large, it will be very difficult to store. To facilitate the storage, an intuitive approach is to store only a part of the full LAT. For a n-bit vector, we can split it into t sub-vectors of m bits, where $n = mt$. When each m-bit sub-vector is determined, the n-bit vector can be obtained by concatenating. This idea give birth to the concept of combinatorial LAT (cLAT).

Property 6. When $u \oplus v \preceq \boldsymbol{z}$ and $u \oplus w \preceq \boldsymbol{z}$, whcih are equivalent to $(u \oplus v) \wedge (\neg((u \oplus v) \wedge \boldsymbol{z})) = \mathbf{0}$ and $(u \oplus w) \wedge (\neg((u \oplus w) \wedge \boldsymbol{z})) = \mathbf{0}$.

Corollary 1. *Let $u, v, w \in \mathbb{F}_2^n$ be the input-output masks of the modular addition with non-zero correlation, let $A = u \oplus v$, $B = u \oplus w$, $C = u \oplus v \oplus w$, $\boldsymbol{z} = M^T(C)$. Splitting the vectors $A = A^{t-1}||\cdots||A^0$, $B = B^{t-1}||\cdots||B^0$, $C = C^{t-1}||\cdots||C^0$, $\boldsymbol{z} = \boldsymbol{z}^{t-1}||\cdots||\boldsymbol{z}^0$ into t sub-vectors respectively, $n = mt$, $A^k, B^k, C^k \in \mathbb{F}_2^m$, $0 \le k \le t-1$. Then the correlation weight of the modular addition can be denoted by*

$$-\log_2 Cor(u, v, w) = \sum_{k=0}^{t-1} wt(\boldsymbol{z}^k) = \sum_{k=0}^{t-1}\sum_{j=0}^{m-1} C_{mk+j+1} \oplus \boldsymbol{z}_{mk+j+1},$$

when

$$A^k \wedge (\neg(A^k \wedge \boldsymbol{z}^k)) = \mathbf{0},$$
$$B^k \wedge (\neg(B^k \wedge \boldsymbol{z}^k)) = \mathbf{0}.$$

Proof. $wt(\boldsymbol{z})$ is the sum of the Hamming weight of each subvector $\boldsymbol{z}^k$, so $-\log_2 Cor(u, v, w) = \sum_{k=0}^{t-1} wt(\boldsymbol{z}^k)$. For $C = u \oplus v \oplus w$, the i^{th} bit in $\boldsymbol{z}$ can be denoted by $\boldsymbol{z}_i = \bigoplus_{j=i+1}^{n-1} C_j = \boldsymbol{z}_{i+1} \oplus C_{i+1}$. Let $0 \le j < i$, $0 \le i \le n-1$, the j^{th} bit in $\boldsymbol{z}^k$ should be $\boldsymbol{z}_j^k = C_{mk+j+1} \oplus \boldsymbol{z}_{mk+j+1}$. Hence, $wt(\boldsymbol{z}^k) = \sum_{j=0}^{m-1} C_{mk+j+1} \oplus \boldsymbol{z}_{mk+j+1}$, when $u_i \oplus v_i \preceq \bigoplus_{l=i+1}^{n-1} C_l$ and $u_i \oplus w_i \preceq \bigoplus_{l=i+1}^{n-1} C_l$ are satisfied, i.e. $A^k \wedge (\neg(A^k \wedge \boldsymbol{z}^k)) = \mathbf{0}$ and $B^k \wedge (\neg(B^k \wedge \boldsymbol{z}^k)) = \mathbf{0}$ for $0 \le k \le t-1$. □

If the m-bit sub-vector $\boldsymbol{z}^{k+1}$ adjacent to $\boldsymbol{z}^k$ is known, $\boldsymbol{z}^k$ can be calculated by sub-vector tuple (u^k, v^k, w^k) and the lowest bit of $\boldsymbol{z}^{k+1}$. We call (u^k, v^k, w^k) as a *sub-block*, and we call the bit $\boldsymbol{z}_{(k+1)m} \in \{0, 1\}$ as the *connection status* when used in the calculation of $\boldsymbol{z}^k$. Spliting the n-bit vector $\boldsymbol{z}$ into t sub-vectors, there should have $t-1$ *connection status* $\boldsymbol{z}_j$, $j \in \{(t-1)m, \cdots, 2m, m\}$, and for the highest sub-vector, its *connection status* $b = 0$. Hence for the hightest sub-block (u^{t-1}, v^{t-1}, w^{t-1}), the Hamming weight of $\boldsymbol{z}^{t-1}$ and the bit $\boldsymbol{z}_{(t-1)m}$ can be obtained, recursively, the Hamming weight of the remaining sub-vectors can also be obtained. Therefore, as *connection status* $b \in \{0, 1\}$, and $u^k, v^k, w^k \in \mathbb{F}_2^m$, we can construct a m-bit lookup table for modular addition in advance, and query the tables by indexing input-output masks and the *connection status*. In additon, the *connection status* for the next sub-block can also be generated.

In the top-down search techniques for the ARX ciphers, for the modular additions in the middle rounds, in most case, only one input mask is fixed (assuming it's v), another input mask w and the output mask u are unknown. In the lookup tables, we need to lookup all valid sub-vectors of (u, w) that correspond to non-zero correlation based on v. The lookup table (called as cLAT) is constructed by Algorithm 2, it takes about 4 seconds on a 2.5 GHz CPU to generate the table with storage size about 1.2GByte when $m = 8$.

Algorithm 2 Constructing the m-bit cLAT for modular addition.

```
1: for each $b \in \{0,1\}$ and input mask $v \in \mathbb{F}_2^m$ do
2:     $cLAT_{min}[v][b] = \mathrm{m}$, let $M^T[k] = 0$ and $cLAT_N[v][b][k] = 0$,for $0 \le k \le m-1$;
3:     for each input mask $w \in \mathbb{F}_2^m$ and output mask $u \in \mathbb{F}_2^m$ do
4:         $A = u \oplus v$, $B = u \oplus w$, $C = u \oplus v \oplus w$, $Cw = 0$;
5:         for $j = 0$ to $m-1$ do
6:             $C_b[j] = (C \gg (m-1-j)) \wedge 1$;
7:         end for
8:         if $b = 1$ then //Determining the connection status generated by the upper sub-block.
9:             $Cw{+}{+}$, $M^T[0] = 1$, $Z = 1 \ll (m-1)$;
10:        else
11:            $M^T[0] = 0$, $Z = 0$;
12:        end if
13:        for $i = 1$ to $m-1$ do //Determining the correlation weight.
14:            $M^T[i] = (C_b[i-1] + M^T[i-1]) \wedge 1$;
15:            if $M^T[i] = 1$ then
16:                $Cw{+}{+}$, $Z = Z \vee (1 \ll (m-1-i))$;
17:            end if
18:        end for
19:        $F_1 = A \wedge (\neg(A \wedge Z))$, $F_2 = B \wedge (\neg(B \wedge Z))$;
20:        if $F_1 = \mathbf{0}$ and $F_2 = \mathbf{0}$ then //Judgment conditions $u \oplus v \preceq \boldsymbol{z}$ and $u \oplus w \preceq \boldsymbol{z}$.
21:            $cLAT_w[v][b][cLAT_N[v][b][Cw]] = w$;
22:            $cLAT_u[v][b][cLAT_N[v][b][Cw]] = u$;
23:            $cLAT_N[v][b][Cw]{+}{+}$; //The number of tuples correspond to $v$ and $b$.
24:            $cLAT_b[u][v][w][b] = (M_T[m-1] + C_b[m-1]) \wedge 1$; //Connection status.
25:            if $cLAT_{min}[v][b] > Cw$ then
26:                $cLAT_{min}[v][b] = Cw$; //The minimum correlation weight correspond to $v$, $b$.
27:            end if
28:        end if
29:    end for
30: end for.
```

3.3 *Spliting-Lookup-Recombination*

Algorithm 2 constructs a m-bit cLAT for the addition modulo 2^n, this section describes how to use it. When one input mask is fixed, we can get another input mask, the output mask and the corresponding correlation weight by the *Splitting-Lookup-Recombination* approach, which contains three steps.

Spliting. For addition modulo 2^n, $n = mt$, if one of the two input masks v is fixed, then split v into t m-bit sub-vectors. The larger m, the fewer times to lookup cLAT and the fewer number of bit concatination operation, but the more space the memory takes up, and after the trade-offs, we choose $m = 8$.

Lookup. From the MSB to the LSB direction, querying the sub-vectors of (u, w) that correspond to each sub-vector of v, and the corresponding correlation weights. For the highest m-bit sub-vector v^{t-1}, its *connection status* $b = 0$, looking up cLAT to get w^{t-1}, u^{t-1}, the corresponding correlation weight $c[t-1]$, and the *connection status* for the sub-vector v^{t-2}. Similarly, other sub-vectors of u, w and the corresponding correlation weights can be obtained.

Recombination. All sub-vectors of u and w can be obtained by lookup tables, and the n-bit u and w can be obtained by bit concatenation. The correlation weight of the modular addition is the sum of the weight of each sub-block, i.e. $Cw = \sum_{k=0}^{t-1} c[k]$.

When there are multiple modular additions in the round function, i.e. $N_A > 1$, for each modular additon, its undetermined input mask and output mask need to be obtained by the *Splitting-Lookup-Recombination* approach respectively. In the lookup phase, a total of tN_A lookup operations are required. And the correlation weight of the round function is $Cw = \sum_{j=1}^{N_A} \sum_{k=0}^{t-1} c_j[k]$.

For each sub-vector v^k, the possible minimum linear correlation weight corresponding to it can be calculated in advance by Algorithm 2, that is,

$$c[k]_{min} = min(cLAT_{min}[v^k][0], cLAT_{min}[v^k][1]) \tag{9}$$

During the *Recombination* phase, the correlation boundary can be constructed by the associated weights that have been obtained and the possible minimum correlation weights, shown in Corollary 2.

Corollary 2. *For additon modulo 2^n, one of the input mask $v = v^{t-1}|| \cdots ||v^0$ is fixed, $n = mt$, $v^k \in \mathbb{F}_2^m$, and $0 \le k \le t-1$. For any $u, w \in \mathbb{F}_2^n$ of non-zero correlation, the correlation boundary should have,*

$$Cor(u, v, w) \le Cor(u^{t-1}|| \cdots ||u^k, v^{t-1}|| \cdots ||v^k, w^{t-1}|| \cdots ||w^k) + 2^{-\sum_{j=0}^{k-1} c[j]_{min}}.$$

Proof. The correlation of modular addition is the product of the correlation of each sub-block after splitting, i.e. $Cor(u, v, w) = \prod_{k=0}^{t-1} Cor(u^k, v^k, w^k)$. Let $-\log_2 Cor(u^{t-1}|| \cdots ||u^k, v^{t-1}|| \cdots ||v^k, w^{t-1}|| \cdots ||w^k) = \sum_{l=k}^{t-1} c[l]$ be the correlation weight of the sub-vector tuples that are obtained by lookup tables. The sum of the correlation weights of the sub-vector tuples have not been looked up yet, which should s.t. $\sum_{j=0}^{k-1} c[j]_{min} \le -\log_2 Cor(u, v, w) - \sum_{l=k}^{t-1} c[l]$. □

Assuming the number of (u^k, w^k) corresponding to each sub-vector v^k is X_k, hence, the number of mask branches corresponding to the modular addition is $\prod_{k=0}^{t-1} X_k$. Corollary 2 can be used to filter out (u, w) of large correlation weight.

3.4 Improved Automatic Search Algorithm

In this section, we adopt a top-down technique[3,15,16], taking the first round as the starting point of the search process. In the first/second rounds, the input-output mask tuples of each modular addition with correlation weight increase monotonically can be obtained by Algorithm 1. In the middle rounds, for each modular addition, u and w can be obtained by the *Spliting-Lookup-Recombination* approach. Algorithm 3 takes SPECK as an example.

Let the optimal correlation weight of the $(r-i)^{th}$ round that has been obtained be Bc_{r-i}, $1 \le i \le r-1$, and let the expected r-round correlation weight be $\overline{Bc_r}$. The correlation weight of the first two rounds should subject to Matsui's pruning condition, i.e. $Cw_1 + Bc_{r-1} \le \overline{Bc_r}$ and $Cw_1 + Cw_2 + Bc_{r-2} \le \overline{Bc_r}$.

Let $\sum_{i=1}^{N'_A} Cw_i = \overline{Bc_r} - Bc_{r-1}$, N'_A is the number of additions in the first two rounds, then the search space need to be constructed is no more than (10). When the block size of a ARX cipher is large, let n be the word size of the modular additon, the total input-output masks of all N'_A modular additons in the first two round is $S' = 2^{3n \times N'_A}$. Therefore, when the value of $\overline{Bc_r} - Bc_{r-1}$ is small, the search space S will be much smaller than S' intuitively.

$$S = \prod_{i=1}^{N'_A} \sum_{c=0}^{Cw_i} \#\{(u,v,w)| - \log_2 Cor(u,v,w) = c\}. \tag{10}$$

Flexible search scenario settings. Combining Algorithm 1, cLAT, pruning conditions and the properties of the target ciphers in Section 2.3, Algorithm 3 can be adapted to the following search scenarios with appropriate modifications.

Scenario 1: For some ARX ciphers, such as SPARX and Chaskey, the number of modular additions in the round function is more than 1, $N_A > 1$. Hence the correlation weight of each round is $Cw_i = \sum_{j=1}^{N_A} Cw_i^j$, Cw_i^j is the correlation weight of the j^{th} addition in the i^{th} round. In the first/second round, the input-output masks of each modular addition should be generated by Algorithm 1. In the middle rounds, the tuple (u, w) of each modular addition should be obtained by calling $LR(v)$ multiple times.

Scenario 2: The linear hulls can also be searched by simply modifying Algorithm 3. For an obtained optimal linear trail, fixed the input mask of the first round and the output mask of the last round, calling Round-$r(i, x, y)$ directly and modifying $\overline{Bc_r}$ to the expected maximum statistical correlation weight C_{max}. Therefore, for ALP, all linear trails with linear correlation weight between the optimal correlation weight C_{min} and C_{max} can be counted.

Scenario 3: When the number of rounds of a linear trail or the block size is large, the linear correlation tend to be very small, and the search process will be very time-consuming. Hence, good linear characteristics results under certain conditions can be explored by the heuristic search settings. We can exclude a large number of search branches and reduce the search complexity by these methods, such as starting the search from a desired large correlation weight $\overline{Bc_r}$, fixed the input mask of a certain round, and limitting the correlation weight of a round or a certain modular addition.

4 Applied to SPECK, SPARX, Chaskey and CHAM-64

4.1 The Linear Hulls for SPECK32/48/64

Applying Algorithm 3, the optimal linear trails for SPECK32/48/64 with correlation close to $2^{-\frac{n}{2}}$ can be obtained, shown in Table 2. Fixed the input and output masks, the ALP of the linear hulls obtained by the cluster experiment are given in Table 3. For SPECK64, a new 14-round linear hull with average linear potential of $2^{-61.24}$ have been found.

When search for the linear hulls, we need modify Algorithm 3 to adapt to *Scenario 2*. We use formula (8) to count ALP, where $C_{min} \leq Cw \leq C_{max}$, C_{min} is the correlation weight of the linear trail we choose to use to pin the input and output masks, and C_{max} is the maximum correlation weight we limit our search. In the middle rounds, we adopt C_{max} to instead $\overline{Bc_r}$ for filtering out those trails that contribute less to the ALP.

Algorithm 3 Automatic search for the optimal linear trails of ARX ciphers, and take the application to SPECK as an example, where $n = mt$.

```
Input: The cLAT is pre-computed and stored by Algorithm 2. $Bw_1, \cdots, Bw_{r-1}$ have been recorded
 1: Program entry:
 2: Let $\overline{Bc_r} = Bc_{r-1} - 1$, and $Bc_r$ = null  //$Bw_1$ can be derived manually for most ARX ciphers..
 3: while $\overline{Bc_r} \neq Bc_r$ do
 4:     $\overline{Bc_r}++$;  //The expected $r$-round correlation weight increases monotonously.
 5:     Call Procedure Round-1;
 6: end while
 7: Exit the program.
 8: Round-1:  //Exclude the search space with correlation weights larger than $\overline{Bc_r} - Bc_{r-1}$.
 9: for $Cw_1 = 0$ to $n-1$ do  //$Cw_1$ increases monotonously.
10:     if $Cw_1 + Bc_{r-1} > \overline{Bc_r}$ then
11:         Return to the upper procedure with FALSE state;
12:     else
13:         Call Algorithm 1 Const($S_{Cw_1}$), and traverse each output tuple $(u_1, v_1, w_1)$;
14:         if call Round-2$(u_1, v_1, w_1)$ and the return value is TRUE, then
15:             Stop Algorithm 1 and return TRUE;  //Record the optimal linear trail be found.
16:         end if
17:     end if
18: end for
19: Return to the upper procedure with FALSE state;
20: Round-2$(u_1, v_1, w_1)$:  //Exclude the correlation weights larger than $\overline{Bc_r} - Bc_{r-1} - Cw_1$.
21: for $Cw_2 = 0$ to $n-1$ do  //$Cw_2$ increases monotonously.
22:     if $Cw_1 + Cw_2 + Bc_{r-2} > \overline{Bc_r}$ then
23:         Return to the upper procedure with FALSE state;
24:     else
25:         Call Algorithm 1 Const($S_{Cw_2}$), and traverse each output tuple $(u_2, v_2, w_2)$;
26:         $y = (u_1 \oplus (v_2 \lll r_a) \oplus w_2) \lll r_b$, $x = u_2 \oplus y$;  //$(r_a, r_b)$: rotation parameters.
27:         if call Round-r$(3, x, y)$ and the return value is TRUE, then
28:             Stop Algorithm 1, compute the masks of the first/second round and return TRUE;
29:         end if
30:     end if
31: end for
32: Return to the upper procedure with FALSE state;
33: Round-$r(i, x, y)$:  //Middle rounds, $3 \leq i \leq r$.
34: $v = x \ggg r_a$, and let $v = v^{t-1}||\cdots||v^0$ and $v^k \in \mathbb{F}_2^m$, $0 \leq k \leq t-1$;  //Spliting $v$.
35: Call $LR(v)$, traversing each $u$ and $w$;  //Where $Cw_i = Cor(v, w, u)$.
36: if $i = r$ and $Cw_1 + \ldots + Cw_{i-1} + Cw_i = \overline{Bc_r}$ then  //The last round.
37:     Let $Bc_r = \overline{Bc_r}$, break from $LR(v)$ and return TRUE;
38: end if  //$r$-round optimal linear trail of expected correlation weight $\overline{Bc_r}$ have been found.
39: $y' = (y \oplus w) \lll r_b$, $x' = y' \oplus u$;
40: if call Round-$r(i+1, x', y')$ and the return value is TRUE, then
41:     Break from $LR(v)$ and return TRUE;  //Record the masks of each round and return.
42: end if
43: Return to the upper procedure with FALSE state;
44: LR$(v)$:  //Looking up cLAT and recombining another input mask $w$ and the output mask $u$.
45: Let $c[k]_{min} = min(cLAT_{min}[v^k][0], cLAT_{min}[v^k][1])$, and $b[k] = 0$, for $0 \leq k \leq t-1$;
46: for $k = t-1$ to 0 do  //From MSB to LSB direction.
47:     for $c_i[k] = cLAT_{min}[v^k][b[k]]$ to $\overline{c_i^k}$ do  //$\overline{c_i^{t-1}} = m-1$ and $\overline{c_i^k} = m$ for $0 \leq k \leq t-2$.
48:         if $\sum_{g=1}^{i-1} Cw_g + \sum_{j=0}^{k-1} c[j]_{min} + \sum_{l=k}^{t-1} c_i[l] + Bc_{r-i} \leq \overline{Bc_r}$ then
49:             for $X_k = 0$ to $cLAT_N[v^{t-1}][b[k]][c_i[k]] - 1$ do
50:                 $u^k = cLAT_u[v^k][b[k]][X_k]$;  //Querying $u^k$ and $w^k$.
51:                 $w^k = cLAT_w[v^k][b[k]][X_k]$;
52:                 $b[k-1] = cLAT_b[u^k][v^k][w^k][b[k]]$; //Record the next connection status.
53:                 if $k = 0$ then  //Recombining $u$ and $w$.
54:                     Output each $u = u^{t-1}||\cdots||u^0$, $w = w^{t-1}||\cdots||w^0$, and $Cw_i = \sum_{k=0}^{t-1} c_i[k]$;
55:                 end if
56:             end for
57:         end if
58:     end for
59: end for
```

Table 2. The 9/10/13 round optimal linear trails for SPECK32/48/64.

r	SPECK32		SPECK48		SPECK64	
	ΓX_r	Cw_r	ΓX_r	Cw_r	ΓX_r	Cw_r
0	00A0062F	1	000180B80001	1	0100012014010021	2
1	78B818B9	4	000000C00001	0	0001810020000101	1
2	00906021	1	00000E00000E	2	0000010000000001	0
3	60804081	1	7A0070700070	5	0000000100000000	1
4	00800001	0	C2C080829380	6	0D0000000C000000	2
5	00010000	1	D00000108300	2	60610000606C0000	3
6	0B000800	3	800000809800	1	00024D0300620C03	6
7	20402050	2	00000400C004	1	181070141B107358	6
8	008380C3	1	200020260020	2	0013001818031840	3
9	170B130A	-	013100310100	2	1818000000181200	2
10			8800A8880109	-	0018000000001000	1
11					0000100000000000	1
12					0000009800000080	2
13					5000040480000404	-

Table 3. The linear hulls for SPECK32/48/64.

2n	r	Γin	Γout	C_{min}	C_{max}	ALP	$\#trails$	Time	Reference
32	9	0010,1400	0B00,0800	15	25	$2^{-29.1}$	69737	N/A	[14]
	9	0380,5224	066A,0608	14	14	2^{-28}	1	N/A	[8]
	9	00A0,062F	170B,130A	14	20	$2^{-27.78}$	14	25s	This paper.
48	10	000131,050021	2484F2,2480F6	22	N/A	2^{-44}	1	N/A	[8]
	10	800121,158021	DE84DC,C684DC	22	N/A	2^{-44}	1	N/A	[14]
	10	000180,B80001	8800A8,880109	22	28	$2^{-43.64}$	50	157.3h	This paper.
64	13	18600010,10724800	00024982,00420802	30	30	2^{-60}	1	N/A	[8]
	13	00101800,00001812	00006065,00006068	30	30	2^{-60}	1	N/A	[14]
	13	01000120,14010021	50000404,80000404	30	32	$2^{-55.29}$	178	7.3h	This paper.
	14	01000120,14010021	26902000,20802006	33	35	$2^{-61.24}$	194	5.8h	This paper.

All experiment code in this work are run on a single high performance server with Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz. All masks are represented in hexadecimal.

Table 4. The linear characteristics for SPARX-64.

r	Cw	T_{trail}	Γin	Γout	C_{min}	C_{max}	ALP	$\#trails$	T_{ALP}
1	0	0s	0000 0000 0080 4001	0000 0000 0000 0001	0	0	1	1	0s
2	0	0s	0000 0000 0080 4001	0000 0000 0004 0004	0	0	1	1	0s
3	1	0s	0000 0000 0080 4001	2C10 2010 2C1C 201C	1	10	2^{-2}	1	0s
4	3	0s	0090 6021 0000 0000	0000 0000 0B00 0800	3	10	2^{-6}	1	0s
5	5	0s	00A0 3021 0000 0000	0000 0000 85C2 8442	5	10	$2^{-9.98}$	3	3s
6	7	0s	0080 5021 0000 0000	4285 4284 4385 4384	7	15	$2^{-13.91}$	11	22s
7	11	3.2m	0090 6021 0000 0000	170B 130A 17CF 130F	11	17	$2^{-21.87}$	5	9s
8	15	4.6h	0880 503D 0090 6021	0000 0000 0E81 1EB0	15	20	$2^{-29.73}$	22	2m
9	19	48.6h	0080 5021 2000 0058	2058 2040 3858 3840	19	25	$2^{-35.97}$	1215	28.7h
10	22	3d	0080 4001 00C0 F001	0205 0204 0205 0204	22	25	$2^{-40.92}$	144	1h
11	≤ 28	5m	0080 4001 00C0 F001	38A8 2080 2058 2040	28	28	2^{-56}	1	-

4.2 The Linear Characteristics for SPARX

Searching for the optimal linear trails of SPARX variants, Algorithm 3 need to be modified to fit *Scenario 1*. There are multiple modular additions in each round, $N_A = 2$ for SPARX-64, and $N_A = 4$ for SPARX-128. Hence, for all $2N_A$ additions modulo 2^{16} in the first/second rounds, call Algorithm 1 for each addition to generate its input-output mask tuples, then applying *Property 3/4* to obtain the masks of the first two rounds for SPARX-64 and SPARX-128 respectively. In the middle rounds, the *Spliting-Lookup-Recombination* method is applied to obtain possible input masks and output masks of each addition, as well as correlation weight. The 10-round optimal linear trails for SPARX-64 are listed in Table 4, and the 11-round linear trail is obtained by limitting the correlation weight of each modular addition less than 3 in the first two rounds. Fixed the input and output mask of the 10-round optimal linear trail, a 10-round linear hull with ALP of $2^{-40.92}$ is obtained.

Table 5. The linear characteristics for SPARX-128.

r	Cw	T_{trail}	Γin	Γout	C_{min}	C_{max}	ALP	$\#trails$	T_{hull}
1	0	0s	0000 0000 0000 0000	0000 0000 0000 0000					
			0000 0000 0080 4001	0000 0000 0000 0001	0	0	1	1	0s
2	0	0s	0000 0000 0000 0000	0000 0000 0000 0000					
			0000 0000 0080 4001	0000 0000 0004 0004	0	0	1	1	0s
3	1	0s	0000 0000 0000 0000	0000 0000 0000 0000					
			0000 0000 0080 4001	0000 0000 2C10 2010	1	3	2^{-2}	1	0s
4	3	0s	0000 0000 0000 0000	0000 0000 0B00 0800					
			0000 0000 0090 6021	0B03 0003 0003 0803	3	10	2^{-6}	1	0s
5	5	2s	0000 0000 00A0 3021	0000 0000 0000 0000					
			0000 0000 0000 0000	0000 0000 85C2 8442	5	10	$2^{-9.98}$	3	0s
6	7	1s	0000 0000 0080 5021	0000 0000 0000 0000					
			0000 0000 0000 0000	0000 0000 4285 4284	7	12	$2^{-13.91}$	3	1s
7	9	4s	0000 0000 0090 6021	0000 0000 0000 0000					
			0000 0000 0000 0000	0000 0000 170B 130A	9	15	$2^{-17.78}$	5	13s
8	12	13d	0000 0000 0081 1381	0000 0000 170B 130A					
			0000 0000 0000 0000	160F 0104 0104 120E	12	18	$2^{-23.65}$	11	101s
9	≥ 14	-	0000 0000 0000 4020	0000 0000 0215 0012					
	≤ 18	27m	0000 0000 0000 0000	0813 0814 0A06 0806	18	22	$2^{-35.22}$	53	6h
10	≥ 18	-	0000 0000 0080 4001	0000 0000 0B81 0800					
	≤ 23	4.4d	0000 0000 0000 0000	0984 0A04 0205 0204	23	23	2^{-46}	1	-

For SPARX-128, the first 8 rounds optimal linear trails can be derived from the first 8 rounds optimal linear trails of SPECK32 when considering the minimum active ARX-box. The experiment results in Table 5 confirmed the derivation. Based on *Scenario 3*, we limit the correlation weight of each modular addition in the first/second round to less than 2, then we get the 9/10-round linear trails with correlation weight of 18/23. Although the 9/10-round linear trail cannot be guaranteed as the best, they can still be used to get the 9/10-round linear hulls with the corresponding ALP of $2^{-35.22}/2^{-46}$.

The 11/10-round linear trails for SPARX-64/SPARX-128 are given by Table 6. c_j^r represents the correlation weight of the j^{th} modular addition in the r^{th} round, Cw_r represents the correlation weight of the r^{th} round, $0 \leq j < N_A$.

Table 6. The 11/10-round linear trails for SPARX-64/SPARX-128.

11-round trail for SPARX-64					10-round trail for SPARX-128						
r	$\Gamma X_0^r\|\|\cdots\|\|\Gamma X_0^r$	c_0^r	c_1^r	Cw_r	r	$\Gamma X_0^r\|\|\Gamma X_1^r\|\|\cdots\|\|\Gamma X_6^r\|\|\Gamma X_7^r$	c_0^r	c_1^r	c_2^r	c_3^r	Cw_r
1	0080400100C0F001	0	1	1	1	00000000000402000000000000000000	0	0	0	0	0
2	000000010000C001	0	0	0	2	00000000810081000000000000000000	0	1	0	0	1
3	0004000400070007	1	2	3	3	00000000C010E0000000000000000000	0	3	0	0	3
$\mathcal{L}$	2C1020102C1C201C	-	-	-	4	00000000364834500000000000000000	0	5	0	0	5
4	2C1C201C00000000	4	0	4	$\mathcal{L}'$	00000000008A90E20000000000000000	-	-	-	-	-
5	3140013000000000	3	0	3	5	000000000000000000000000008A90E2	0	0	0	2	2
6	858D05CF00000000	4	0	4	6	0000000000000000000000000078F138E	0	0	0	4	4
$\mathcal{L}$	60117B1800000000	-	-	-	7	0000000000000000000000000000E1600	0	0	0	2	2
7	0000000060117B18	0	4	4	8	00000000000000000000000000001800	0	0	0	0	0
8	0000000000812261	0	3	3	$\mathcal{L}'$	00000000000000000000000060006000	-	-	-	-	-
9	0000000080018280	0	1	1	9	00000000600060006000000000006000	0	1	1	0	2
$\mathcal{L}$	0000000000020202	-	-	-	10	00000000828182010280020080018001	0	2	1	1	4
10	0002020200000200	1	0	1	11	0000000002150012081308140A060806	-	-	-	-	-
11	1E08180808000800	3	1	4							
12	38A8208020582040	-	-	-							

4.3 The Linear Characteristics for Chaskey

Shown in Table 7, the correlation weights of the first 3 rounds optimal linear trails of Chaskey we have found are 0/2/9. For one round optimal linear trail with correlation weight 0, whose input-output masks represented in hexadecimal are (1,1,0,0) and (0,80,800000,0). To find the linear trails with longer rounds, we use a heuristic approach in *Scenario 3*, limitting the correlation weight of each modular additions in the first round to less than 2, and setting the correlation weight to the expected values to start the heuristic search. The 4/5-round linear trails with correlation weights of 29/61 are obtained. The details of the linear trails are listed in Table 8.

Table 7. The correlation of the linear trails for Chaskey.

Round	1	2	3	4	5	Reference
Correlation	2^{-1}	2^{-2}	2^{-9}	-	-	[14]
Correlation	1	2^{-2}	2^{-9}	$\geq 2^{-29}$	$\geq 2^{-61}$	This paper.

4.4 The Linear Characteristics for CHAM-64

In the first 4 rounds of CHAM, the input-output mask tuples of each modular additions can be constructed by Algorithm 1, and all input and output masks for the first 4 rounds are deduced by *Property 5*. In the forward search process, the *Spliting-Lookup-Recombination* approach is adopted to determine the possible unknown input masks and output masks for the modular addition in each round. In [11], for CHAM-64, a 34-round linear trail with bias of $\varepsilon = 2^{-31}$ was given. The correlation weights of the optimal linear trails obtained by us are shown in Table 9. The correlation of the 34-round optimal linear trail we find is 2^{-31}, and the details of the 35-round optimal linear trail with correlation of 2^{-33} is listed in Table 10.

Table 8. The linear trails for Chaskey.

	2 round with $Cor = 2^{-2}$					3 round with $Cor = 2^{-9}$				
r	a, b, c, d	c_0^r	c_1^r	c_2^r	c_3^r	a, b, c, d	c_0^r	c_1^r	c_2^r	c_3^r
0	2000001,B100001,1,1	1	0	0	0	1,8100001,201,303	0	1	0	1
1	0,1,0,0	0	0	0	1	300,1,0,0	5	0	0	0
2	300004,1000,10000000,4	-	-	-	-	0,10000,1,0	0	0	1	1
3						240030,10008080,81011000,240000	-	-	-	-
	4 round with $Cor = 2^{-29}$					5 round with $Cor = 2^{-61}$				
r	a, b, c, d	c_0^r	c_1^r	c_2^r	c_3^r	a, b, c, d	c_0^r	c_1^r	c_2^r	c_3^r
0	0,800,90001,10D0801	0	2	1	0	1,8100001,201,303	0	1	0	1
1	0,0,1,1	0	0	0	0	300,1,0,0	5	0	0	0
2	0,80,810000,0	0	2	3	1	0,10000,1,0	0	0	1	1
3	18001A20,80880040,400189,1A20	5	3	7	5	240030,10008080,81011000,240000	4	4	6	6
4	2D224005,A83F0DA8,2DA1C86D,25004107	-	-	-	-	50E73286,8241A0,5161469B,40D436A6	10	12	4	6
5						BAAE7E16,76224512,65104022,3EA61E37	-	-	-	-

Table 9. The correlation weights of the optimal linear trails for CHAM-64.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Cw	0	0	0	0	0	0	0	1	2	3	3	4	5	6	7	9	10	12
Round	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	
Cw	14	16	17	18	20	21	22	23	25	26	26	26	27	28	29	31	33	

Table 10. The 35-round optimal linear trail for CHAM-64.

r	$\Gamma X_0^r \| \cdots \| \Gamma X_3^r$	Cw_r	r	$\Gamma X_0^r \| \cdots \| \Gamma X_3^r$	Cw_r
1	0000000000018000	0	19	0007010380C08000	1
2	0000000180000000	0	20	810080C080000700	2
3	0001800000000000	0	21	0001800007000201	0
4	0000000000000100	0	22	0000070002010100	0
5	0000000001000000	0	23	0700020101000000	2
6	0000010000000000	0	24	0001010000000006	0
7	0100000000000000	1	25	0000000000060002	0
8	00C0000000000001	1	26	0000000600020000	0
9	8000000000010100	1	27	0006000200000000	1
10	4000000101000 0C0	1	28	0000000000000400	0
11	006101000 0C08000	1	29	0000000004000000	0
12	812000C080004100	4	30	0000040000000000	0
13	0041800041008201	1	31	0400000000000000	1
14	0030410082014100	3	32	0200000000000006	1
15	6500820141000060	3	33	0002000000060600	1
16	A0C1410000600047	3	34	0001000606000200	0
17	C080006000470103	2	35	0106060002000002	2
18	60000047010380C0	1	36	0682020000028401	-

5 Conclusions

In this paper, we have improved the automatic search algorithm for the linear characteristics on ARX ciphers. Combining with the optimization strategies of constructing the input-output masks correspond to specific correlation weight and the novel concept of cLAT, this search tool enables an efficient search for the linear characteristics on typical ARX ciphers. Applying this tool, we get new 9/10/14-round linear hulls for SPECK32/48/64, the ALP are $2^{-27.78}$, $2^{-43.64}$ and $2^{-61.24}$ respectively. For SPARX-64, a 10-round optimal linear trail with correlation of 2^{-22}, and a 11-round good linear trail with correlation of 2^{-28} have been obtained. For SPARX-128, a 10-round linear trail with correlation of 2^{-23} is obtained. The linear cryptanalysis results on SPARX are presented for the first time so far. For Chaskey, the linear characteristic results have been updated, which cover more rounds than the existing results. For CHAM-64, the linear characterstics we obtained are the first third-party linear cryptanalysis results. In addition, we believe that these improved optimization strategies can also be achieved to linear cryptanalysis on other ARX ciphers.

Acknowledgements. The authors will be very grateful to the anonymous referees for their helpful comments and suggestions. This work was supported by the National Key Research and Development Program of China (No. 2017YF-B0801900).

References

1. 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
2. Biryukov, A., Perrin, L.: State of the art in lightweight symmetric cryptography. IACR Cryptology ePrint Archive **2017**, 511 (2017)
3. 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)
4. Bodden, D.: Linear cryptanalysis of reduced-round speck with a heuristic approach: Automatic search for linear trails. In: Information Security - 21st International Conference, ISC 2018, Guildford, UK, September 9-12, 2018, Proceedings. pp. 132–150 (2018), https://doi.org/10.1007/978-3-319-99136-8_8
5. Daemen, J., Govaerts, R., Vandewalle, J.: Correlation matrices. In: Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings. pp. 275–285 (1994), https://doi.org/10.1007/3-540-60590-8_21
6. 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
7. 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

8. 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
9. 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)
10. 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
11. 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)
12. 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)
13. Liu, Y., Fu, K., Wang, W., Sun, L., Wang, M.: Linear cryptanalysis of reduced-round SPECK. Inf. Process. Lett. **116**(3), 259–266 (2016), https://doi.org/10.1016/j.ipl.2015.11.005
14. 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
15. Liu, Z., Li, Y., Wang, M.: Optimal differential trails in simon-like ciphers. IACR Trans. Symmetric Cryptol. **2017**(1), 358–379 (2017)
16. Liu, Z., Li, Y., Wang, M.: The security of simon-like ciphers against linear cryptanalysis. IACR Cryptology ePrint Archive **2017**, 576 (2017)
17. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. pp. 386–397 (1993), https://doi.org/10.1007/3-540-48285-7_33
18. Matsui, M.: On correlation between the order of s-boxes and the strength of DES. In: Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. pp. 366–375 (1994), https://doi.org/10.1007/BFb0053451
19. Mella, S., Daemen, J., Assche, G.V.: New techniques for trail bounds and application to differential trails in keccak. IACR Trans. Symmetric Cryptol. **2017**(1), 329–357 (2017), https://doi.org/10.13154/tosc.v2017.i1.329-357
20. Mouha, N.: Chaskey: a MAC algorithm for microcontrollers - status update and proposal of chaskey-12 -. IACR Cryptology ePrint Archive **2015**, 1182 (2015)
21. Mouha, N., Mennink, B., Herrewege, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In: Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers. pp. 306–323 (2014), https://doi.org/10.1007/978-3-319-13051-4_19
22. Nyberg, K.: Linear approximation of block ciphers. In: Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. pp. 439–444 (1994)

23. Nyberg, K., Wallén, J.: Improved linear distinguishers for SNOW 2.0. In: Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers. pp. 144–162 (2006)
24. Schulte-Geers, E.: On ccz-equivalence of addition mod 2 n. Des. Codes Cryptography **66**(1-3), 111–127 (2013), https://doi.org/10.1007/s10623-012-9668-4
25. 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)
26. Wallén, J.: Linear approximations of addition modulo 2^n. In: Fast Software Encryption, 10th International Workshop, FSE2003, Lund, Sweden, February 24-26, 2003, Revised Papers. pp. 261–273 (2003)
27. Wallén, J.: On the differential and linear properties of addition. Master's thesis, Helsinki University of Technology, Laboratory for Theoretical Computer Science (2003), http://www.tcs.hut.fi/Publications/bibdb/HUT-TCS-A84.pdf
28. Yao, Y., Zhang, B., Wu, W.: Automatic search for linear trails of the SPECK family. In: Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings. pp. 158–176 (2015)