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

Mingjiang Huang\textsuperscript{1,2}, Liming Wang\textsuperscript{1}

\textsuperscript{1} State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
\textsuperscript{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 preferred 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 characteristics 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 $(v, w) \rightarrow u$ 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 re-
duction 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 cal-
culation 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 calcula-
tion of modular addition, and introduce an algorithm to construct the input-out
masks of specific correlation weight. Then, we propose a novel concept of combi-
national 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 characteris-
tics 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 ob-
tained. 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 character-
cistics 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 preliminar-
ies 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 construct-
ing 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 addition modulo $2^n$, i.e. $x \oplus y = z$, we use the symbols $\ll$, $\gg$ to indicate
rotation to the left and right, and $\lll$, $\ggg$ to indicate the left and right shift
operation, respectively. The binary operator symbols $\oplus$, $\lor$, $\land$, $\|$, $\neg$ 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.

<table>
<thead>
<tr>
<th>Variants</th>
<th>Round</th>
<th>Cor</th>
<th>Tadd</th>
<th>ALP</th>
<th>TALP</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECK32</td>
<td>9</td>
<td>2−14</td>
<td>N/A</td>
<td>2−29.1</td>
<td>N/A</td>
<td>[14]</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>2−14</td>
<td>N/A</td>
<td>2−28</td>
<td>N/A</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>2−14</td>
<td>9s</td>
<td>2−27.78</td>
<td>25a</td>
<td>This paper.</td>
</tr>
<tr>
<td>SPECK48</td>
<td>10</td>
<td>2−22</td>
<td>N/A</td>
<td>2−24</td>
<td>N/A</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2−22</td>
<td>N/A</td>
<td>2−24</td>
<td>N/A</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2−22</td>
<td>3.2h</td>
<td>2−23.64</td>
<td>157.3h</td>
<td>This paper.</td>
</tr>
<tr>
<td>SPECK64</td>
<td>13</td>
<td>2−30</td>
<td>N/A</td>
<td>2−60</td>
<td>N/A</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>2−30</td>
<td>8.6h</td>
<td>2−55.29</td>
<td>7.3h</td>
<td>This paper.</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>2−33</td>
<td>25.6h</td>
<td>2−61.24</td>
<td>5.8h</td>
<td>This paper.</td>
</tr>
<tr>
<td>SPARX-64</td>
<td>10</td>
<td>2−22</td>
<td>3d</td>
<td>2−29</td>
<td>1h</td>
<td>This paper.</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>2−28</td>
<td>5m</td>
<td>2−26</td>
<td>-</td>
<td>This paper.</td>
</tr>
<tr>
<td>SPARX-128</td>
<td>9</td>
<td>2−18</td>
<td>27m</td>
<td>2−37.22</td>
<td>6h</td>
<td>This paper.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2−23</td>
<td>4.4d</td>
<td>2−46</td>
<td>-</td>
<td>This paper.</td>
</tr>
<tr>
<td>Chaskey</td>
<td>3</td>
<td>2−30</td>
<td>N/A</td>
<td>2−60</td>
<td>N/A</td>
<td>[14]</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2−29</td>
<td>15.7m</td>
<td>2−58</td>
<td>-</td>
<td>This paper.</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2−31</td>
<td>6.6h</td>
<td>2−122</td>
<td>-</td>
<td>This paper.</td>
</tr>
<tr>
<td>CHAM-64</td>
<td>34</td>
<td>2−31</td>
<td>N/A</td>
<td>2−62</td>
<td>N/A</td>
<td>[11]</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>2−31</td>
<td>1.1d</td>
<td>2−62</td>
<td>-</td>
<td>This paper.</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>2−33</td>
<td>4.8d</td>
<td>-</td>
<td>-</td>
<td>This paper.</td>
</tr>
</tbody>
</table>

2.2 Linear Correlation Calculation for Modular Addition

Let \( F_2^n \) be the \( n \) dimensional vector space over binary field \( F_2 = \{0, 1\} \), for boolean function \( f : F_2^n \rightarrow F_2 \) and \( h : F_2^n \rightarrow F_2 \), \( x \in F_2^n \), 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 addition \( x \oplus y = z \), 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 \((v \cdot x \oplus w \cdot y) \oplus u \cdot z = 0\), the linear approximation probability is defined as

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

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

\[
Cor(u, v, w) = 2^{-3n} \times \sum_{x, y, z \in F_2^n} \mu((v \cdot x \oplus w \cdot y) \oplus u \cdot z). \tag{3}
\]

Let \( Pr(u, v, w) = \frac{1}{2} + \varepsilon \), where \( \varepsilon \) is the bias. When \((v \cdot x \oplus w \cdot y) \oplus u \cdot z = 1\), the linear approximation probability is \( Pr(u, v, w) = \frac{1}{2} - \varepsilon \). 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 $Cw(u, v, w) = -\log_2 \text{Cor}(u, v, w)$ as the correlation weight, the linear square correlation can be denoted by

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

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

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

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 $d_i = u_i \oplus w_i \in \mathbb{F}_2$ is an octal word, $0 \leq i \leq n - 1$. Then, the linear correlation can be denoted by

$$\text{Cor}(u, v, w) = LA_{d_{n-1}}A_{d_{n-2}} \cdots A_{d_1}A_{d_0}C.$$ Where the row vector $L = (1, 0)$, the column vector $C = (1, 1)^T$, and each $2 \times 2$ matrix $A_{d_i}$ is defined by

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

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

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

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

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

where $1_{G}$ 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 $0 \leq i \leq n - 1$. 

Automatic Search for the Linear (hull) Characteristics of ARX Ciphers 5
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, $\Gamma_{in}$, $\Gamma_{out}$ are the input and output mask of the $r$-round linear trail, the correlation of it can be denoted by

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

(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).

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

(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 chosen as the fixed input-output masks of the linear hull. $C_{max}$ is the upper bound be searched, which should be chosen by the trade-off between the search time and the accuracy of $ALP$.

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

(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 \textit{Property 1}.

\textbf{Property 1.} If \((u[i], v[i], w[i])\) and \((u[i+1], v[i+1], w[i+1])\) are given, then there have \(\Gamma X[i] = v[i] \ll r_a, \Gamma X[i+1] = v[i+1] \ll r_a, \Gamma Y[i] = (\Gamma Y[i+1] \gg r_b) \oplus w[i], \Gamma Y[i+2] = (\Gamma Y[i+1] \oplus w[i+1]) \ll r_b, \) and \(\Gamma X[i+2] = \Gamma Y[i+1] \oplus u[i+1].\)

The linear layer functions \(L/L'\) [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.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2.png}
\caption{(A) and (B) represent the linear layer of SPARX-64, (C) and (D) represent the linear layer of SPARX-128.}
\end{figure}

\textbf{Property 2.} For SPARX-64, if the masks are transformed by the linear layer function \(L\), let \(c = \Gamma X_2 \oplus \Gamma X_3, d = c \gg 8\), 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 \(\Gamma X_3 = \Gamma X_1 \oplus d \oplus \Gamma X_3.\)

\textbf{Property 3.} For SPARX-128, if the masks are transformed by the linear layer function \(L'\), let \(e = \Gamma X_4 \oplus \Gamma X_6 \oplus \Gamma X_8 \oplus \Gamma X_7, f = e \gg 8\), there have \(\Gamma X'_0 = \Gamma X_4, \Gamma X'_1 = \Gamma X_5, \Gamma X'_2 = \Gamma X_6, \Gamma X'_3 = \Gamma X_7, \Gamma X'_4 = \Gamma X_0 \oplus f \oplus \Gamma X_6, \Gamma X'_5 = \Gamma X_1 \oplus f \oplus \Gamma X_5, \Gamma X'_6 = \Gamma X_0 \oplus f \oplus \Gamma X_4, \) 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 \textit{Property 4}.

\textbf{Property 4.} For the permutation of Chaskey, if the input-output masks of each modular addition in the first round are \((u[i], v[i], w[i]), 0 \leq i \leq 3,\) 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 \(a = v[0], b = w[0] \oplus ((u[0] \oplus v[3]) \gg 8)\).
Huang et al.

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

\[c = v[1], d = w[1] \oplus (u[1] \oplus v[2]) \gg 8, a' = u[3] \oplus (u[1] \oplus v[2] \oplus w[3]) \ll 13, b' = (u[0] \oplus w[2] \oplus w[3] \ll 16) \ll 7, c' = u[2] \oplus (w[2] \oplus (u[0] \oplus v[3]) \gg 16) \ll 7, \text{ and } d' = (u[1] \oplus v[2] \oplus w[3]) \ll 13.\]

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.

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 \(\leq i \leq 4\), the input and output masks of the first 4 rounds can be deduced by follows.

\[
\begin{align*}
\Gamma X_0^0 &= v[1], \\
\Gamma X_0^1 &= v[2], \\
\Gamma X_0^2 &= (w[2] \ll 8) \oplus v[3], \\
\Gamma X_0^3 &= (w[3] \ll 1) \oplus v[4]; \\
\Gamma X_1^0 &= (w[1] \ll 1) \oplus v[2], \\
\Gamma X_1^1 &= \Gamma X_0^2, \\
\Gamma X_1^2 &= \Gamma X_0^3, \\
\Gamma X_1^3 &= u[1] \ll 8; \\
\Gamma X_2^0 &= \Gamma X_1^2, \\
\Gamma X_2^1 &= \Gamma X_3^2, \\
\Gamma X_2^2 &= \Gamma X_3^3, \\
\Gamma X_2^3 &= u[2] \ll 1; \\
\Gamma X_3^0 &= (w[4] \ll 8) \oplus \Gamma X_1^3, \\
\Gamma X_3^1 &= \Gamma X_2^3, \\
\Gamma X_3^2 &= \Gamma X_2^4, \\
\Gamma X_3^3 &= u[4] \ll 1.
\end{align*}
\]
3 Automatic Search for the linear Characteristics on ARX Ciphers

3.1 Input-Output Masks of Specific Correlation Weight

The number of input-output masks tuple 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 C_w from 0 to n − 1, and construct the input-output masks subspace S_{C_w} corresponding to correlation weight C_w, 0 ≤ C_w ≤ n − 1. Therefore, the total space S can be divided into n subspaces, i.e. S = \bigcup_{C_w=0}^{n-1} S_{C_w}.

Definition 1. Let ((v, w) → 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 ≤ i ≤ n − 1.

Definition 2. Let’s define three sets that \( \xi_i \) 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 \( z \) and the 3 consecutive words in \( \Phi \) have following relationships, shown in Observation 1.

Observation 1. Let \( x = u \oplus v \oplus w \) and \( z_i = \bigoplus_{j=i+1}^{n-1} x_j \) for 0 ≤ i ≤ n − 2, \( z_{n-1} = 0 \), hence, \( z_i = z_{i+1} \oplus x_{i+1} \). For \( z \in \mathbb{F}_2^n \), assuming when \( x_j = 0 \) for \( n − 1 ≥ j > i + 1 \), or \( z_{i+2} = 0 \), there should have \( u_i \oplus v_i \leq z_i \), \( u_i \oplus w_i \leq z_i \) on bit level, it’s equivalent to \( u_{i-1} \oplus v_{i-1} \leq x_{i+1} \oplus x_{i} \) and \( u_{i-1} \oplus w_{i-1} \leq x_{i+1} \oplus x_{i} \)

Since \( x_i = 1 \) when \( \xi_i \in U_0 \), and \( x_i = 0 \) when \( \xi_i \in U_1 \), there have,

if \( (z_{i+1}, z_i, z_{i-1}) = (0, 0, 0) \), then \( \xi_{i+1} = 0, \xi_i = 0, \xi_{i-1} = 0 \);

if \( (z_{i+1}, z_i, z_{i-1}) = (0, 0, 1) \), then \( \xi_{i+1} = 0, \xi_i = 0, \xi_{i-1} = 7 \);

if \( (z_{i+1}, z_i, z_{i-1}) = (0, 1, 0) \), then \( \xi_{i+1} = 0, \xi_i = 7, \xi_{i-1} \in U_0 \);

if \( (z_{i+1}, z_i, z_{i-1}) = (0, 1, 1) \), then \( \xi_{i+1} = 0, \xi_i = 7, \xi_{i-1} \in U_1 \);

if \( (z_{i+1}, z_i, z_{i-1}) = (1, 0, 0) \), then \( \xi_{i+1} = 7, \xi_i \in U_0, \xi_{i-1} = 0 \);

if \( (z_{i+1}, z_i, z_{i-1}) = (1, 0, 1) \), then \( \xi_{i+1} = 7, \xi_i \in U_0, \xi_{i-1} = 7 \);

if \( (z_{i+1}, z_i, z_{i-1}) = (1, 1, 0) \), then \( \xi_{i+1} = 7, \xi_i \in U_1, \xi_{i-1} \in U_0 \);

if \( (z_{i+1}, z_i, z_{i-1}) = (1, 1, 1) \), 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 \( z_{i+1}, z_i \) are active. The last significant bits \( (u_0, v_0, w_0) \) of the input-output masks construct the value of \( z_0 \), which is only related to the Hamming weight of \( z_0 = \bigoplus_{j=1}^{n-1} x_j \), i.e. \( u_0 \oplus v_0 \leq wt(z_0) \) and \( u_0 \oplus w_0 \leq wt(z_0) \). Therefore, if we get
the Hamming weight distribution of \( z \), from the LSB to MSB direction, as \( z_0 \) is determined, \( \xi_0 \) can be obtained. Next, \( \mathbf{x}_1 = z_1 \oplus z_0 \) is determined, and \( u_1 \oplus v_1 \preceq wt(\mathbf{x}_1) \), \( u_1 \oplus w_1 \preceq wt(\mathbf{z}_1) \) should be satisfied, hence, the possible values of \( \xi_1 \) can be obtained. Recursively, all \( \xi_i \) 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 \( \Phi \). The process to construct the subspace \( S_{Cw} \) is shown in Algorithm 1, marked as \textbf{Const}(\( S_{Cw} \)).

\begin{algorithm}
\caption{\textbf{Const}(\( S_{Cw} \)): Constructing the input-output mask tuples with linear correlation weight of \( Cw \) for modular addition, \( 0 \leq Cw \leq n - 1 \).}
\begin{algorithmic}[1]
\Input \( Cw \) and \( A = \{\lambda_{Cw}, ..., \lambda_1\} \). Each pattern of the Hamming weight distribution of \( z \) can be calculated by the combinations algorithm in [7], which is the combinations pattern of \( \binom{(\mathbb{C}w)_{-1}}{Cw} \).
\State where \( \xi_i = u_i \| v_i \| w_i \) for \( 0 \leq i \leq n - 1 \).
\State \textbf{Func}_{LSB}: \( i = 0 \). \hspace{1cm} //Constructing the LSBs of \( u, v, w \).
\If {\( Cw = 0 \)}
\State Output the tuple of \((u, v, w)\) with \((1,1,1)\) or \((0,0,0)\);
\EndIf
\If {\( \lambda_i \neq 0 \)}
\For {each \( \xi_i \in U_2, c = 1, Fw = 0 \)}
\Call{Func\_Middle}{$i + 1, c, Fw$};
\EndFor
\Else
\For {each \( \xi_i \in U_2, c = 2, Fw = 1 \)}
\Call{Func\_Middle}{$i + 1, c, Fw$};
\EndFor
\EndIf
\For {each \( \xi_i \in U_1, c = 1, Fw' = 1 \)}
\Call{Func\_Middle}{$i + 1, c + 1, Fw'$};
\EndFor
\Else
\For {each \( \xi_i \in U_1, c = 1, Fw' = 0 \)}
\Call{Func\_Middle}{$i + 1, c + 1, Fw'$};
\EndFor
\EndIf
\EndFor
\EndIf
\EndFor
\State \Call{Func\_MSB}{$i, c, Fw$}: \hspace{1cm} //Constructing the bits of \( u, v, w \) with position higher than \( \lambda_{Cw} \).
\If {\( \lambda_i \neq i \)}
\If {\( Fw = 0 \)}
\State \hspace{1cm} //The value of \( Fw \) determines whether \( \xi_i \) belongs to \( U_0 \) or \( U_1 \).
\EndIf
\EndIf
\EndFor
\EndIf
\EndFor
\EndIf
\EndFor
\EndFor
\EndFor
\EndFor
\EndFor
\EndIf
\EndFor
\EndIf
\State \Call{Func\_Middle}{$i, c, Fw$}: \hspace{1cm} //Constructing the middle bits of \( u, v, w \).
\If {\( \lambda_i = i \)}
\If {\( Fw' = 0 \)}
\State \hspace{1cm} //The value of \( Fw \) determines whether \( \xi_i \) equals to 0 or 7.
\EndIf
\EndIf
\EndFor
\EndFor
\EndFor
\EndIf
\EndFor
\EndFor
\EndIf
\EndFor
\EndFor
\EndFor
\EndAlgorithm

\subsection{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 an \( n \)-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 gives birth to the concept of combinatorial LAT (cLAT).

**Property 6.** When \( u \oplus v \preceq z \) and \( u \oplus w \preceq z \), which are equivalent to \((u \oplus v) \land (-(u \oplus v) \land z)) = 0\) and \((u \oplus w) \land (-(u \oplus w) \land z)) = 0\).

**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 \( A = u \oplus v \), \( B = u \oplus w \), \( C = u \oplus v \oplus w \), \( z = M^T(C) \). Splitting the vectors \( A = A^{t-1} || \cdots || A^0 \), \( B = B^{t-1} || \cdots || B^0 \), \( C = C^{t-1} || \cdots || C^0 \), \( z = z^{t-1} || \cdots || z^0 \) into \( t \) sub-vectors respectively, \( n = mt \), \( A^k, B^k, C^k \in \mathbb{F}_2^m \), \( 0 \leq k \leq t - 1 \). Then the correlation weight of the modular addition can be denoted by

\[
- \log_2 \text{Cor}(u, v, w) = \sum_{k=0}^{t-1} \sum_{j=0}^{m-1} C_{mk+j+1} \oplus z_{mk+j+1},
\]

when

\[
A^k \land (-(A^k \land z^k)) = 0,
\]

\[
B^k \land (-(B^k \land z^k)) = 0.
\]

**Proof.** \( wt(z) \) is the sum of the Hamming weight of each subvector \( z^k \), so

\[
- \log_2 \text{Cor}(u, v, w) = \sum_{k=0}^{t-1} \sum_{j=0}^{m-1} C_{mk+j+1} \oplus z_{mk+j+1},
\]

when \( A^k \land (-(A^k \land z^k)) = 0 \) and \( B^k \land (-(B^k \land z^k)) = 0 \) for \( 0 \leq k \leq t - 1 \).

If the \( m \)-bit sub-vector \( z^{k+1} \) adjacent to \( z^k \) is known, \( z^k \) can be calculated by sub-vector tuple \( (u^k, v^k, w^k) \) and the lowest bit of \( z^{k+1} \). We call \( (u^k, v^k, w^k) \) as a sub-block, and we call the bit \( z_{(k+1)m} \in \{0, 1\} \) as the connection status when used in the calculation of \( z^k \). Splitting the \( n \)-bit vector \( z \) into \( t \) sub-vectors, there should have \( t - 1 \) connection status \( z_j \), \( j \in \{(t - 1)m, \cdots, 2m, m\} \), and for the highest sub-vector, its connection status \( b = 0 \). Hence for the highest sub-block \( (u^{t-1}, v^{t-1}, w^{t-1}) \), the Hamming weight of \( z^{t-1} \) and the bit \( z_{(t-1)m} \) 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 \( u^k, v^k, w^k \in \mathbb{F}_2^n \), 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 addition, 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 $v \in \mathbb{F}_2^n$ do
2: \hspace{1em} $cLAT_{MT}[v][b] = m$, let $MT^T[k] = 0$ and $cLAT_N[v][b][k] = 0$, for $0 \leq k \leq m - 1$;
3: for each input mask $w \in \mathbb{F}_2^n$ and output mask $u \in \mathbb{F}_2^n$ do
4: \hspace{1em} $A = u \oplus v$, $B = u \oplus w$, $C = u \oplus v \oplus w$, $Cw = 0$;
5: \hspace{1em} for $j = 0$ to $m - 1$ do
6: \hspace{2em} $C_b[j] = (C \gg (m - 1 - j)) \land 1$;
7: \hspace{1em} end for
8: \hspace{1em} if $b = 1$ then //Determining the connection status generated by the upper sub-block.
9: \hspace{2em} $Cw++$, $MT^T[0] = 1$, $Z = 1 \ll (m - 1)$;
10: \hspace{1em} else
11: \hspace{2em} $MT^T[0] = 0$, $Z = 0$;
12: \hspace{1em} end if
13: \hspace{1em} for $i = 1$ to $m - 1$ do //Determining the correlation weight.
14: \hspace{2em} $Cw++$, $Z = Z \lor (1 \ll (m - 1 - i))$;
15: \hspace{1em} end if
16: \hspace{1em} end for
17: \hspace{1em} end for
18: \hspace{1em} $F_1 = A \land (\sim(A \land Z))$, $F_2 = B \land (\sim(B \land Z))$;
19: \hspace{1em} if $F_1 = 0$ and $F_2 = 0$ then //Judgment conditions $u \oplus v \leq z$ and $u \oplus v \leq z$.
20: \hspace{2em} $cLAT_{MT}[v][b][cLAT_N[v][b][Cw]] = w$;
21: \hspace{2em} $cLAT_{MT}[v][b][cLAT_N[v][b][Cw]] = u$;
22: \hspace{2em} $cLAT_N[v][b][Cw] = ++$; //The number of tuples correspond to $v$ and $b$.
23: \hspace{2em} $cLAT_{MT}[v][b][b] = (MT^T[m - 1] + C_b[m - 1]) \land 1$; //Connection status.
24: \hspace{2em} if $cLAT_{MT}[v][b] > Cw$ then
25: \hspace{3em} $cLAT_{MT}[v][b] = Cw$; //The minimum correlation weight correspond to $v$, $b$.
26: \hspace{3em} end if
27: \hspace{2em} end if
28: \hspace{1em} end for
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 Spliting-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 concatenation 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. $Cw = \sum_{k=1}^{t-1} c[k]$.

When there are multiple modular additions in the round function, i.e. $N_A > 1$, for each modular addition, 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,

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

(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 addition 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 $0 \leq k \leq t - 1$. For any $u, w \in \mathbb{F}_2^m$ of non-zero correlation, the correlation boundary should have,

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

*Proof.* The correlation of modular addition is the product of the correlation of each sub-block after splitting, i.e. $\text{Cor}(u, v, w) = \prod_{k=0}^{t-1} \text{Cor}(u^k, v^k, w^k)$. Let $-\log_2 \text{Cor}(u^{t-1} \cdots || u^k, v^{t-1} \cdots || v^k, w^{t-1} \cdots || w^k) = \sum_{j=k}^{t-1} c[l]$ 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. $\sum_{j=0}^{k-1} c[j]_{\text{min}} \leq -\log_2 \text{Cor}(u, v, w) - \sum_{l=k}^{t-1} c[l]$. 

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 Splitting-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 $Bc_{r-i}$, $1 \leq i \leq r - 1$, and let the expected $r$-round correlation weight be $Bc_r$. The correlation weight of the first two rounds should subject to Matsui’s pruning condition, i.e. $Cw_1 + Bc_{r-1} \leq Bc_r$ and $Cw_1 + Cw_2 + Bc_{r-2} \leq Bc_r$.

Let $\sum_{i=1}^{N_A'} Cw_i = Bc_r - Bc_{r-1}$, $N_A'$ 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 addition, the total input-output masks of all $N_A'$ modular additions in
the first two round is \( S' = 2^{3n \times N_A} \). Therefore, when the value of \( Bc_r - Bc_{r-1} \) is small, the search space \( S \) will be much smaller than \( S' \) intuitively.

\[
S = \prod_{i=1}^{N_A} Cw_i \sum_{c=0} \# \{(u, v, w) | -\log_2 Cor(u, v, w) = c\}.
\]

(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 \( Cw_i = \sum_{j=1}^{N_A} Cw^{j}_i \), \( Cw^{j}_i \) 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 \( Bc_r \) 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 \( Bc_r \), fixed the input mask of a certain round, and limiting 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 \( Bc_r \) 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 \( n = m t \).

**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 \( Bc_r = Bc_{r-1} - 1 \), and \( Bc_r = \text{null} \) //\( Bw_1 \) can be derived manually for most ARX ciphers.
3. while \( Bc_r \neq Bc_r \) do
4. \( Bc_r + 1 \) //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 \( Bc_r - Bc_{r-1} \).
9. for \( Cw_1 = 0 \) to \( n - 1 \) do //\( Cw_1 \) increases monotonously.
10. if \( Cw_1 + Bc_{r-1} > Bc_r \) then
11. Return to the upper procedure with FALSE state;
12. else
13. Call Algorithm 1 \( \text{Const}(\mathit{Sc}_{c_{w_1}}) \), 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 \( Bc_r - Bc_{r-1} - Cw_1 \).
21. for \( Cw_2 = 0 \) to \( n - 1 \) do //\( Cw_2 \) increases monotonously.
22. if \( Cw_2 + Bc_{r-1} + Cw_1 > Bc_r \) then
23. Return to the upper procedure with FALSE state;
24. else
25. Call Algorithm 1 \( \text{Const}(\mathit{Sc}_{c_{w_2}}) \), and traverse each output tuple \((u_2, v_2, w_2)\);
26. \( y = (u_1 \oplus v_1, w_1, u_1) \in \mathcal{R}_r \); //\( r \)-round optimal linear trail of expected correlation weight
27. if call Round-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-3(\( x, y) \):** //Middle rounds, \( 3 \leq i \leq r \).
34. \( x = x_{i-1} \), and let \( v = x_{i-1} \oplus \cdots \oplus x_{r-1} \in \mathbb{F}_2^n \), \( 0 \leq i \leq t - 1 \); //Spliting \( v \).
35. Call LR\( (v) \), traversing each \( u \) and \( w \); //Where \( Cw_i = \text{Cor}(v, w, u) \).
36. if \( i = r \) and \( Cw_1 + \cdots + Cw_{r-1} + Cw_1 > Bc_r \) then //The last round.
37. Let \( Bc_r = Bc_r \), break from LR\( (v) \) and return TRUE;
38. end if
39. if \( i = r \) and \( Cw_1 + \cdots + Cw_{r-1} + Cw_i = Bc_r \) then //The last round.
40. Break from LR\( (v) \) and return TRUE; //Record the masks of each round and return.
41. end if
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 \( c_i[k] = \min(c_i \cdot \text{LAT}_{\nu_i}^k[0], \text{LAT}_{\nu_i}^k[1]) \), 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 \( c_i[k] = \text{LAT}_{\nu_i}^k[0]b[k] \) to \( c_i[k] \) do //\( c_i[k] = m - 1 \) and \( c_i[k] = m \) for \( 0 \leq k \leq t - 2 \).
48. if \( \sum_i=1^t c_i[k] + \sum_{g=0}^{i-1} c_g[l] + Bc_{r-1} \leq Bc_r \) then
49. for \( X_k = 0 \) to \( \text{LAT}_{\nu_i}^k[2] \in \text{LAT}_{\nu_i}^{k+1}[0] \), \( k = 0 \) do //Querying \( u^* \) and \( w^* \).
50. \( u^* = \text{LAT}_{\nu_i}^k[2]{\text{LAT}}_{\nu_i}^k[0]X_k \); //Recombining \( u^* \) and \( w^* \).
51. \( w^* = \text{LAT}_{\nu_i}^k[2]{\text{LAT}}_{\nu_i}^k[0]X_k \); \( k = 0 \) then //Recombining \( u \) and \( w \).
52. Output each \( u = u^* \), where \( w = w^* \), \( Cw_i = \sum_{k=0}^{t-1} c_i[k] \); //Record the next connection status.
53. end if
54. end if
55. end for
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.

<table>
<thead>
<tr>
<th>r</th>
<th>SPECK32</th>
<th>SPECK48</th>
<th>SPECK64</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_{in}$</td>
<td>$C_{w}$</td>
<td>$T_{in}$</td>
</tr>
<tr>
<td>9</td>
<td>0000062F</td>
<td>1</td>
<td>000180880001</td>
</tr>
<tr>
<td>1</td>
<td>78881B89</td>
<td>4</td>
<td>000000000001</td>
</tr>
<tr>
<td>2</td>
<td>00906021</td>
<td>1</td>
<td>00000000000E</td>
</tr>
<tr>
<td>6</td>
<td>06804081</td>
<td>1</td>
<td>7A0070007078</td>
</tr>
<tr>
<td>4</td>
<td>08000000</td>
<td>0</td>
<td>C2080829380</td>
</tr>
<tr>
<td>5</td>
<td>00001000</td>
<td>1</td>
<td>000000000780</td>
</tr>
<tr>
<td>8</td>
<td>00009000</td>
<td>3</td>
<td>000000000000</td>
</tr>
<tr>
<td>7</td>
<td>00000000</td>
<td>0</td>
<td>000004000000</td>
</tr>
<tr>
<td>8</td>
<td>00380803</td>
<td>1</td>
<td>200002060020</td>
</tr>
<tr>
<td>9</td>
<td>1708133A</td>
<td>1</td>
<td>013000310100</td>
</tr>
<tr>
<td>10</td>
<td>880018880009</td>
<td>1</td>
<td>0018000000000000</td>
</tr>
<tr>
<td>11</td>
<td>0000099000000000</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>5000040400000404</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>2n</th>
<th>$r$</th>
<th>$T_{in}$</th>
<th>$T_{out}$</th>
<th>$C_{min}$</th>
<th>$C_{max}$</th>
<th>$ALP$</th>
<th># trails</th>
<th>Time</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>9</td>
<td>0010,1400</td>
<td>0800,0800</td>
<td>015</td>
<td>25</td>
<td>2^{-28.4}</td>
<td>69737</td>
<td>N/A</td>
<td>[14]</td>
</tr>
<tr>
<td>9</td>
<td>0380,0224</td>
<td>066A,0608</td>
<td>14</td>
<td>14</td>
<td>2^{-28}</td>
<td>1</td>
<td>N/A</td>
<td>[8]</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>03A0,062F</td>
<td>170B,1304</td>
<td>14</td>
<td>20</td>
<td>2^{-37.76}</td>
<td>14</td>
<td>25s</td>
<td>This paper.</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>10</td>
<td>000131,00021</td>
<td>2484F2,2480F6</td>
<td>22</td>
<td>N/A</td>
<td>2^{-24}</td>
<td>1</td>
<td>N/A</td>
<td>[8]</td>
</tr>
<tr>
<td>10</td>
<td>0080121,159021</td>
<td>08800C,08640C</td>
<td>22</td>
<td>5</td>
<td>N/A</td>
<td>2^{-44}</td>
<td>1</td>
<td>N/A</td>
<td>[14]</td>
</tr>
<tr>
<td>10</td>
<td>000180,880001</td>
<td>880048,880109</td>
<td>22</td>
<td>28</td>
<td>2^{-3.6}</td>
<td>50</td>
<td>157.3h</td>
<td>This paper.</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>13</td>
<td>18600015,10724000,0024982,00248020</td>
<td>30</td>
<td>30</td>
<td>2^{-59}</td>
<td>1</td>
<td>N/A</td>
<td>[8]</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>00101800,00001812</td>
<td>00000605,00000608</td>
<td>30</td>
<td>30</td>
<td>2^{-60}</td>
<td>1</td>
<td>N/A</td>
<td>[14]</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>01000120,1400021</td>
<td>50000040,80000040</td>
<td>30</td>
<td>32</td>
<td>2^{-55.29}</td>
<td>178</td>
<td>7.3h</td>
<td>This paper.</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>01000120,1401021</td>
<td>26900000,20800006</td>
<td>33</td>
<td>35</td>
<td>2^{-61.24}</td>
<td>194</td>
<td>5.8h</td>
<td>This paper.</td>
<td></td>
</tr>
</tbody>
</table>

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 SPARX64.

<table>
<thead>
<tr>
<th>$r$</th>
<th>$C_{w}$</th>
<th>$T_{trail}$</th>
<th>$T_{in}$</th>
<th>$T_{out}$</th>
<th>$C_{min}$</th>
<th>$C_{max}$</th>
<th>$ALP$</th>
<th># trails</th>
<th>$T_{ALP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0s</td>
<td>0000 0000 0080 4001</td>
<td>0000 0000 0000 0001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0s</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0s</td>
<td>0000 0000 0080 4001</td>
<td>0000 0000 0000 0004</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0s</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0s</td>
<td>0000 0000 0080 4001</td>
<td>2C10 2010 2C1C 201C</td>
<td>1</td>
<td>10</td>
<td>2^{-2}</td>
<td>1</td>
<td>0s</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0s</td>
<td>0090 6021 0000 0000</td>
<td>0000 0000 0800 0800</td>
<td>3</td>
<td>10</td>
<td>2^{-6}</td>
<td>1</td>
<td>0s</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0s</td>
<td>0040 3021 0000 0000</td>
<td>0000 0000 85C2 8442</td>
<td>5</td>
<td>10</td>
<td>2^{-9.98}</td>
<td>3</td>
<td>3s</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>0s</td>
<td>0080 5021 0000 0000</td>
<td>4285 4284 4385 4384</td>
<td>7</td>
<td>15</td>
<td>2^{-13.91}</td>
<td>11</td>
<td>22s</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>3.2m</td>
<td>0090 5021 0000 0000</td>
<td>170B 1304 170F 130F</td>
<td>11</td>
<td>17</td>
<td>2^{-21.87}</td>
<td>5</td>
<td>9s</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>4.6h</td>
<td>0880 5030 0090 6021</td>
<td>0000 0000 0861 1860</td>
<td>15</td>
<td>20</td>
<td>2^{-29.73}</td>
<td>22</td>
<td>2m</td>
</tr>
<tr>
<td>9</td>
<td>19</td>
<td>4.6h</td>
<td>0880 5021 2000 0058</td>
<td>2058 2040 3858 3840</td>
<td>19</td>
<td>25</td>
<td>2^{-35.97}</td>
<td>1215</td>
<td>28.7h</td>
</tr>
<tr>
<td>10</td>
<td>22</td>
<td>3d</td>
<td>0080 4001 0000 0000</td>
<td>0205 0204 0205 0204</td>
<td>22</td>
<td>25</td>
<td>2^{-40.92}</td>
<td>144</td>
<td>1h</td>
</tr>
<tr>
<td>11</td>
<td>28</td>
<td>5m</td>
<td>0080 4001 0000 0000</td>
<td>38AB 2080 2068 2049</td>
<td>28</td>
<td>28</td>
<td>2^{-56}</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>
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 limiting 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.

<table>
<thead>
<tr>
<th>$r$</th>
<th>$C_w$</th>
<th>$T_{round}$</th>
<th>$T_{in}$</th>
<th>$T_{out}$</th>
<th>$C_{min}$</th>
<th>$C_{max}$</th>
<th>$ALP$</th>
<th>$#trails$</th>
<th>$T_{hull}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0001</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>4</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>22</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>24</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>26</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>28</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>30</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>32</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>34</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>36</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>38</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>40</td>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

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. $C_w^j$ represents the correlation weight of the $j^{th}$ modular addition in the $r^{th}$ round, $C_w^r$ 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.

<table>
<thead>
<tr>
<th>Round</th>
<th>11-round trail for SPARX-64</th>
<th>10-round trail for SPARX-128</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( r \Gamma_X r )</td>
<td>( r \Gamma_X r )</td>
</tr>
<tr>
<td>1</td>
<td>( 0004000100000001 )</td>
<td>( 0000000000000000 )</td>
</tr>
<tr>
<td>2</td>
<td>( 0000010000000000 )</td>
<td>( 0000000000000000 )</td>
</tr>
<tr>
<td>3</td>
<td>( 0000000000000000 )</td>
<td>( 0000000000000000 )</td>
</tr>
<tr>
<td>4</td>
<td>( 2C120102012C1201 )</td>
<td>( 05000000 )</td>
</tr>
<tr>
<td>5</td>
<td>( 3140013000000000 )</td>
<td>( 00000000 )</td>
</tr>
<tr>
<td>6</td>
<td>( 0004000400070007 )</td>
<td>( 00000000 )</td>
</tr>
<tr>
<td>7</td>
<td>( 0000000000000000 )</td>
<td>( 00000000 )</td>
</tr>
<tr>
<td>8</td>
<td>( 0000000060117B18 )</td>
<td>( 00000000 )</td>
</tr>
<tr>
<td>9</td>
<td>( 0000000000018280 )</td>
<td>( 00000000 )</td>
</tr>
<tr>
<td>10</td>
<td>( 0000000080202002 )</td>
<td>( 00000000 )</td>
</tr>
<tr>
<td>11</td>
<td>( 0000000012081308140A060806 )</td>
<td>( 00000000 )</td>
</tr>
<tr>
<td>12</td>
<td>( 384208020582040 )</td>
<td>( 00000000 )</td>
</tr>
</tbody>
</table>

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, limiting 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.

<table>
<thead>
<tr>
<th>Round</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[14]</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>$r$</th>
<th>2 round with $Cor = 2^{−4}$</th>
<th>3 round with $Cor = 2^{−9}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a, b, c, d$</td>
<td>$e'_0, e'_1, e'_2, e'_3$</td>
</tr>
<tr>
<td>0</td>
<td>2000001,8100001,1,1</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0,1,0,0</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>300004,1000,10000000,4</td>
<td>- - - -</td>
</tr>
<tr>
<td>3</td>
<td>200030,100080,81011000,240000</td>
<td>- - - -</td>
</tr>
<tr>
<td></td>
<td>240030,100080,81011000,240000</td>
<td>- - - -</td>
</tr>
<tr>
<td></td>
<td>2D224005,A83F0DA8,2DA1C86D,25004107</td>
<td>- - - -</td>
</tr>
<tr>
<td></td>
<td>50E73286,8241A0,5161469B,40D436A6</td>
<td>- - - -</td>
</tr>
<tr>
<td></td>
<td>BAAE7E16,76224512,65104022,3EA61E37</td>
<td>- - - -</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Round</th>
<th>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Cw$</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Round</th>
<th>19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Cw$</td>
<td>14 16 17 18 20 21 22 23 25 26 27 28 29 30 31 33</td>
</tr>
</tbody>
</table>

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

| $r$ | $TX_0||\cdots||TX_3$ | $Cw_r$ | $r$ | $TX_0||\cdots||TX_3$ | $Cw_r$ |
|-----|-----------------------|--------|-----|-----------------------|--------|
| 1   | 000000000000180000    | 0      | 19  | 000000000000180000    | 0      |
| 2   | 000000000000180000    | 0      | 20  | 000000000000180000    | 0      |
| 3   | 000000000000180000    | 0      | 21  | 000000000000180000    | 0      |
| 4   | 000000000000180000    | 0      | 22  | 000000000000180000    | 0      |
| 5   | 000000000000180000    | 0      | 23  | 000000000000180000    | 0      |
| 6   | 000000000000180000    | 0      | 24  | 000000000000180000    | 0      |
| 7   | 000000000000180000    | 0      | 25  | 000000000000180000    | 0      |
| 8   | 000000000000180000    | 0      | 26  | 000000000000180000    | 0      |
| 9   | 000000000000180000    | 0      | 27  | 000000000000180000    | 0      |
| 10  | 000000000000180000    | 0      | 28  | 000000000000180000    | 0      |
| 11  | 000000000000180000    | 0      | 29  | 000000000000180000    | 0      |
| 12  | 000000000000180000    | 0      | 30  | 000000000000180000    | 0      |
| 13  | 000000000000180000    | 0      | 31  | 000000000000180000    | 0      |
| 14  | 000000000000180000    | 0      | 32  | 000000000000180000    | 0      |
| 15  | 000000000000180000    | 0      | 33  | 000000000000180000    | 0      |
| 16  | 000000000000180000    | 0      | 34  | 000000000000180000    | 0      |
| 17  | 000000000000180000    | 0      | 35  | 000000000000180000    | 0      |
| 18  | 000000000000180000    | 0      | 36  | 000000000000180000    | 0      |
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 characteristics 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. 2017YFB0801900).

References


