




Differential-Linear Approximation Semi-Unconstrained Searching and Partition Tree: Application to LEA and Speck

Yi Chen¹ , Zhenzhen Bao^{2,4} , and Hongbo Yu^{3,4} 

¹ Institute for Advanced Study, Tsinghua University, Beijing, China

² Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China

³ Department of Computer Science and Technology, Tsinghua University, Beijing, China

⁴ Zhongguancun Laboratory, Beijing, China

{chenyi2023, zzbao, yuhongbo}@mail.tsinghua.edu.cn

Abstract. The differential-linear attack is one of the most effective attacks against ARX ciphers. However, two technical problems are preventing it from being more effective and having more applications: (1) there is no efficient method to search for good differential-linear approximations. Existing methods either have many constraints or are currently inefficient. (2) partitioning technique has great potential to reduce the time complexity of the key-recovery attack, but there is no general tool to construct partitions for ARX ciphers. In this work, we step forward in solving the two problems. First, we propose a novel idea for generating new good differential-linear approximations from known ones, based on which new searching algorithms are designed. Second, we propose a general tool named partition tree, for constructing partitions for ARX ciphers. Based on these new techniques, we present better attacks for two ISO/IEC standards, i.e., LEA and Speck. For LEA, we present the first 17-round distinguisher which is 1 round longer than the previous best distinguisher. Furthermore, we present the first key recovery attacks on 17-round LEA-128, 18-round LEA-192, and 18-round LEA-256, which attack 3, 4, and 3 rounds more than the previous best attacks. For Speck, we find better differential-linear distinguishers for Speck48 and Speck64. The first differential-linear distinguishers for Speck96 and Speck128 are also presented.

Keywords: Differential-Linear Attack · Partition · LEA · Speck.

1 Introduction

Differential cryptanalysis [11] and linear cryptanalysis [24] are the two best-known cryptanalysis techniques in symmetric cryptography. The preliminary step of differential cryptanalysis (respectively, linear cryptanalysis) is to search for high probability (resp. correlation) differential distinguishers (resp. linear

distinguishers). In the last decade, a mainstream direction is developing efficient methods to automatically search for differential or linear distinguishers, such as the work in [22, 26, 27, 32]. The larger space these methods can search in a practical time, the more firm our confidence in a cipher’s resistance against differential or linear cryptanalysis is.

For some block ciphers, within a few rounds, it is easy to find high probability (resp. correlation) differential distinguishers (resp. linear distinguishers); however, with the increase in the number of rounds, differential and linear cryptanalysis simultaneously lose their power. Differential-linear attack is a technique that combines differential cryptanalysis with linear cryptanalysis such that their individual power over a small number of rounds can be joined to efficiently attack more rounds. It is first introduced by Langford and Hellman [20] in 1994 and has been applied to the security evaluation of various ciphers (e.g., IDEA [17], Serpent [16], COCUNUT98 [10], and Chaskey [21], etc.).

In differential-linear attacks, a preliminary step is to search for differential-linear distinguishers (denoted by $\Delta_{\text{in}} \rightarrow \gamma_{\text{out}}$) with high correlation. The development of automatic searching for differential-linear distinguishers has almost been at a virtual standstill for the past 20 years. Before 2023, researchers usually search for differential-linear distinguishers in a three-stage way: (1) experimentally verify short differential-linear distinguishers denoted by $\Delta_m \rightarrow \gamma_m$ (2) search short differential distinguishers (denoted by $\Delta_{\text{in}} \rightarrow \Delta_m$) and linear distinguishers (denoted by $\gamma_m \rightarrow \gamma_{\text{out}}$), (3) concatenate three short distinguishers into a long differential-linear distinguisher $\Delta_{\text{in}} \rightarrow \Delta_m \rightarrow \gamma_m \rightarrow \gamma_{\text{out}}$. Although there have been many improvements [2, 5, 10, 21] to key recovery attacks based on differential-linear distinguishers, there is no efficient method to search for short differential-linear distinguishers $\Delta_m \rightarrow \gamma_m$. Thus, in practice, for a difference Δ_m , the search space is severely limited to the case of a linear mask γ_m of Hamming weight 1 or 2 [5, 16, 28], which is a long-term pain point since the extremely limited search space will weaken our confidence in a cipher’s resistance to differential-linear attacks. In March 2023, using Mixed-Integer Quadratic Constraint Programming (MIQCP) and Mixed-Integer Linear Programming (MILP) techniques, Bellini et al. [6] and Lv et al. [23] both propose an automatic search method respectively, which is good progress. However, as the authors admit, the efficiency of their methods is currently not high [6, 23], which is further demonstrated by the comparison (see Table 9) with our search method proposed in this paper.

Building symmetric-key primitives with modular additions, rotations, and XORs is a common practice. The resulting primitives are named ARX ciphers and their representatives can be found everywhere, including block ciphers (e.g., LEA [18] and Speck [3]), stream ciphers (e.g., Salsa20 [7]), MAC algorithms (e.g., Chaskey [25]), and so on. Differential-linear attack is one of the best attacks against ARX ciphers, such as the work in [4, 5, 21]. Recently, Beierle et al. proposed several improvements to the framework of differential-linear attacks with a special focus on ARX ciphers [4, 5]. Among these improvements, building partitions for complex encryption functions is very helpful to reduce the key bits

to be guessed. However, except for introducing the final partitions of a special example, the authors in [4, 5] give no general tools to build partitions for other encryption functions, which is also a pain point since it hinders researchers from further applying the partitioning technique to other ARX ciphers.

Contribution. First, we propose a novel idea for searching for good differential-linear approximations, based on which we design new search algorithms that have no constraints on the intermediate linear mask and are efficient. We apply the search algorithms to LEA and Speck, and have found many better distinguishers. Table 1 summarizes the comparison of distinguishers. For Speck, only the 11-round differential-linear distinguisher of Speck48 is reported in Table 1, since it is the best distinguisher so far. Actually, we have found the first differential-linear distinguishers for 13-round Speck64, 15-round Speck96, and 18-round Speck128, respectively. Refer to Tables 8 and 9 for more details.

Table 1. The comparison of distinguishers. Cor: correlation, Pr: Probability.

Cipher	Type	Round	Cor / Pr	Source
LEA	Differential	13	$\text{Pr} = 2^{-123.79}$	[30]
	Impossible Differential	10	$\text{Pr} = 0$	[18]
	Boomerang	16	$\text{Pr} = 2^{-117.1114}$	[19]
	Differential-Linear	17	$\text{Cor} = -2^{-59.04}$	This Paper
Speck48	Differential	11	$\text{Pr} = 2^{-44.31}$	[30]
	Differential-Linear	11	$\text{Cor} = -2^{-17.55}$	[23]
	Differential-Linear	11	$\text{Cor} = -2^{-17.40}$	This Paper

Second, we propose a general tool named partition tree for building partitions for ARX encryption functions. Using this tool, we build dynamic partitions for parallel modular additions, based on which best key recovery attacks on round reduced LEA are obtained. Table 2 summarizes the key recovery attacks. The designers claim that the secure number of rounds is 17 for LEA-128, 18 for LEA-192, and 19 for LEA-256 [18].

Note that the 17-round key recovery attack as shown in Table 2 is based on the attack framework introduced in [4, 5], and the 18-round key recovery attack is obtained by using the classical attack framework summarized in [12]. Due to the limitation in the differential part, we can only attack 17-round LEA using the attack framework in [4, 5].

Table 2. Key recovery attacks on round-reduced LEA. R.A.: Rounds Attacked, T.R.: Total Rounds, D.T.: Distinguisher Type, CP: Chosen-Plaintexts.

Variant	R.A. / T.R.	D.T.	Time	Data (CP)	Reference
LEA-128	14/24	Differential	$2^{124.79}$	$2^{124.79}$	[30]
	17/24	Differential-Linear	$2^{82.9}$	$2^{70.9}$	This Paper
LEA-192	14/28	Differential	$2^{124.79}$	$2^{124.79}$	[30]
	17/28	Differential-Linear	$2^{82.9}$	$2^{70.9}$	This Paper
	18/28	Differential-Linear	$2^{189.63}$	$2^{126.63}$	This Paper
LEA-256	15/32	Differential	$2^{252.79}$	$2^{124.79}$	[30]
	17/32	Differential-Linear	$2^{82.9}$	$2^{70.9}$	This Paper
	18/32	Differential-Linear	$2^{189.63}$	$2^{126.63}$	This Paper

2 Preliminaries

Given a set $\mathcal{S} \subseteq \mathbb{F}_2^n$ and a Boolean function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$, we define the correlation

$$\mathbf{Cor}_{x \in \mathcal{S}}[f(x)] := \frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} (-1)^{f(x)}. \quad (1)$$

We denote the XOR operation by \oplus , the j -th unit vector of a binary vector space by $[j]$, and the sum of unit vectors $[j_1] \oplus [j_2] \oplus \dots \oplus [j_t]$ by $[j_1, j_2, \dots, j_t]$. Given a vector $x \in \mathbb{F}_2^n$, $x[j]$ denotes the j -th bit of x . For $x, \lambda \in \mathbb{F}_2^n$, we define the inner product by $\langle \lambda, x \rangle = \bigoplus_{j=0}^{n-1} \lambda[j]x[j]$.

2.1 Differential-Linear Distinguisher

Fig.1 shows the latest structure of differential-linear distinguishers [2]. The cipher E is divided into three sub-ciphers E_1 , E_m , and E_2 , such that $E = E_2 \circ E_m \circ E_1$. A differential distinguisher and a linear distinguisher are applied to E_1 and E_2 successively.

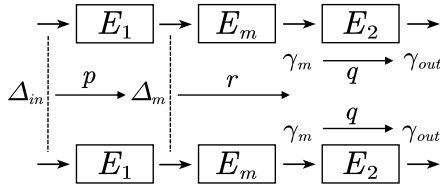


Fig. 1. The latest structure of differential-linear distinguishers.

Assume that the differential $\Delta_{\text{in}} \xrightarrow{E_1} \Delta_m$ holds with probability p , the linear approximation $\gamma_m \xrightarrow{E_2} \gamma_{\text{out}}$ has correlation q , and the experimental correlation of the middle part $\Delta_m \xrightarrow{E_m} \gamma_m$ is r , i.e.,

$$\begin{aligned} \Pr_{x \in \mathbb{F}_2^n} [E_1(x) \oplus E_1(x \oplus \Delta_{\text{in}}) = \Delta_m] &= p, \\ \mathbf{Cor}_{x \in \mathbb{F}_2^n} [\langle \gamma_m, x \rangle \oplus \langle \gamma_{\text{out}}, E_2(x) \rangle] &= q, \\ \mathbf{Cor}_{x \in \mathcal{S}} [\langle \gamma_m, E_m(x) \rangle \oplus \langle \gamma_m, E_m(x \oplus \Delta_m) \rangle] &= r, \end{aligned} \quad (2)$$

where \mathcal{S} denotes the set of samples over which the correlation is computed¹. Then the total correlation of the differential-linear distinguisher $\Delta_{\text{in}} \xrightarrow{E} \gamma_{\text{out}}$ is estimated as

$$\mathbf{Cor}_{x \in \mathbb{F}_2^n} [\langle \gamma_{\text{out}}, E(x) \rangle \oplus \langle \gamma_{\text{out}}, E(x \oplus \Delta_{\text{in}}) \rangle] = prq^2. \quad (3)$$

Therefore, by preparing $\epsilon p^{-2} r^{-2} q^{-4}$ pairs of chosen plaintexts $(x, x \oplus \Delta_{\text{in}})$, where ϵ is a small non-negative constant, one can distinguish the cipher from a pseudorandom permutation.

Recently, Beierle et al. proposed a technique to reduce the complexity [5]. The high-level idea of the technique is as follows. Denote the set of all right pairs for the differential $\Delta_{\text{in}} \xrightarrow{E_1} \Delta_m$ by \mathcal{X} . To amplify the correlation of the distinguisher $\Delta_{\text{in}} \xrightarrow{E_1} \gamma_{\text{out}}$, we choose $\epsilon r^{-2} q^{-4}$ right pairs in the set \mathcal{X} to observe its correlation. To efficiently get the right pairs, we exploit the structure of the set \mathcal{X} . Concretely, the set \mathcal{X} might have a special structure, such that for any $x \in \mathcal{X}$, one can obtain a set $X = \{(x \oplus u, x \oplus u \oplus \Delta_{\text{in}}) | u \in \mathcal{U}\}$, where \mathcal{U} is a subspace, such that all elements in X are the right pairs for the differential $\Delta_{\text{in}} \rightarrow \Delta_m$. For a differential whose set of right pairs has such a special structure, once one right pair is obtained, one can generate a set of $2^{\dim \mathcal{U}}$ right pairs for free. To find such subspace \mathcal{U} for a differential, one can use the concept of the differential's neutral bits [9]. In particular, we require $2^{\dim \mathcal{U}} \geq \epsilon r^{-2} q^{-4}$. For some differentials for which obtaining a large enough \mathcal{U} is difficult, one might use a probabilistic approach related to the concept of probabilistic neutral bits [1]. Assume that the probability that a randomly generated input x belongs to \mathcal{X} is \bar{p} . Then the complexity of the distinguisher is $\epsilon \bar{p}^{-1} r^{-2} q^{-4}$.

2.2 Partitioning Technique for Key Recovery

The partitioning technique is first proposed by Biham and Carmeli [8] to amplify the bias of a linear approximation of addition. It is then improved and used by Gaëtan in differential-linear attacks and helps improve data complexity significantly [21]. Recently, Beierle et al. not only further propose improved

¹ When E_m involves round keys, the correlation r is estimated using N samples and M random keys, i.e., computing an empirical value using a random key and repeating for M times. The final value of r is set to the median (or mean) of the obtained M values [2, 12, 16].

partitioning techniques for a modular addition, but also introduce a partition technique for complex ARX encryption function that contains two consecutive modular additions [4, 5]. The basic idea of these partitioning techniques is to partition the data into several subsets according to some ciphertext bits. In each subset, one can observe a high correlation of a specific approximation. Besides, with partitioning technique, partial key guessing is more feasible.

The partition is done according to the property of modular addition. Specifically, consider two n -bit words x and z , and a modular addition operation

$$\mathbb{F}_2^{2n} \rightarrow \mathbb{F}_2^n, \quad (x, z) \mapsto y = x \boxplus z,$$

as depicted in Fig. 2. In [4], Beierle et al. introduce Lemma 1 to compute the value of $z[i]$ and $z[i] \oplus z[i-1]$.

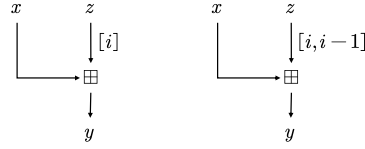


Fig. 2. One modular addition. We are interested in the value of $z[i]$ or $z[i] \oplus z[i-1]$.

Lemma 1. [4] *Let $s = y \oplus x$ and let $i \geq 3$. Let $\mathcal{S}_{b_0 b_1} := \{(x, y) \in \mathbb{F}_2^{2n} | s[i-1] = b_0 \text{ and } s[i-2] = b_1\}$. We have*

$$z[i] = \begin{cases} x[i] \oplus y[i] \oplus y[i-1] \oplus 1, & \text{with corr. 1, if } (x, y) \in \mathcal{S}_{1*} \\ x[i] \oplus y[i] \oplus y[i-2] \oplus 1, & \text{with corr. 1, if } (x, y) \in \mathcal{S}_{01} \\ x[i] \oplus y[i] \oplus y[i-3] \oplus 1, & \text{with corr. 0.5, if } (x, y) \in \mathcal{S}_{00}, \end{cases} \quad (4)$$

$$z[i] \oplus z[i-1] = \begin{cases} x[i] \oplus y[i], & \text{with corr. 1, if } (x, y) \in \mathcal{S}_{0*} \\ x[i] \oplus y[i] \oplus y[i-1] \oplus y[i-2] \oplus 1, & \text{with corr. 1, if } (x, y) \in \mathcal{S}_{11} \\ x[i] \oplus y[i] \oplus y[i-1] \oplus y[i-3] \oplus 1, & \text{with corr. 0.5, if } (x, y) \in \mathcal{S}_{10}, \end{cases}$$

where $\mathcal{S}_{1*} = \mathcal{S}_{10} \cup \mathcal{S}_{11}$, and $\mathcal{S}_{0*} = \mathcal{S}_{00} \cup \mathcal{S}_{01}$.

In the attacks presented later, we also adopt another way to compute the value of $z[i]$, see Lemma 2.

Lemma 2. *Let $s = y \oplus x$ and let $i \geq 3$. Let $\mathcal{S}_{b_0 b_1} := \{(x, y) \in \mathbb{F}_2^{2n} | s[i-1] = b_0 \text{ and } s[i-2] = b_1\}$. We have*

$$z[i] = \begin{cases} y[i] \oplus x[i] \oplus x[i-1], & \text{with corr. 1, if } (x, y) \in \mathcal{S}_{1*} \\ y[i] \oplus x[i] \oplus x[i-2], & \text{with corr. 1, if } (x, y) \in \mathcal{S}_{01} \\ y[i] \oplus x[i] \oplus x[i-3], & \text{with corr. 0.5, if } (x, y) \in \mathcal{S}_{00} \end{cases}$$

where $\mathcal{S}_{1*} = \mathcal{S}_{10} \cup \mathcal{S}_{11}$.

For the convenience of applying Lemmas 1 and 2 in the rest of this paper, we denote by $v \langle i_1, \dots, i_m \rangle$ the value $\langle \gamma, v[i_1] \parallel \dots \parallel v[i_m] \rangle$ in the case of an indeterministic linear mask γ . For example, formula 4 is rewritten as $z[i] = x[i] \oplus y \langle i, i-1, i-2, i-3 \rangle \oplus 1$.

3 Differential-Linear Approximations Searching

Consider the middle part E_m which is verified experimentally. Given a difference Δ_m , we will first introduce a new method to search for differential-linear approximations $\Delta_m \rightarrow \gamma_m$ with a correlation higher than a threshold. Then it is extended to search for differential-linear approximations $\Delta_m \rightarrow \gamma_{\text{out}}$ of $E_2 \circ E_m$.

3.1 Core Idea and Motivation

Core idea. Fig. 3 depicts the core idea of our search methods. Assume that the correlations of two differential-linear approximations are known. We can generate a new differential-linear approximation by the XOR operation, e.g., $\gamma_3 = \gamma_1 \oplus \gamma_2$ or $\gamma_6 = \gamma_4 \oplus \gamma_5$. Assume that the absolute correlation of $\Delta \rightarrow \gamma_i$ for $i \in \{1, 2\}$ exceeds a threshold, and the absolute correlation of $\Delta \rightarrow \gamma_j$ for $j \in \{4, 5\}$ does not exceed the threshold. Then we preferentially test $\Delta \rightarrow \gamma_3$ due to the motivation introduced later.

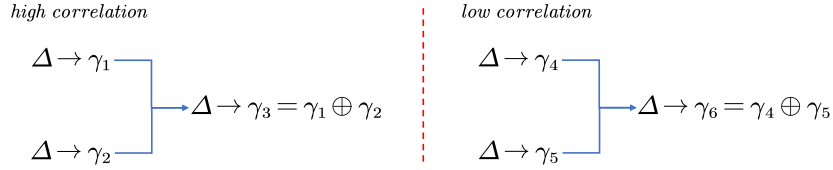


Fig. 3. The core idea of our differential-linear approximation searching methods. We preferentially verify the correlation of $\Delta \rightarrow \gamma_3$ which is generated from two differential-linear approximations with high correlation.

Motivation. The motivation behind our idea is relatively intuitive. Given a differential-linear approximation $\Delta \rightarrow \gamma$ of an encryption function E , we define a variable

$$z_\gamma = \langle E(P) \oplus E(P \oplus \Delta), \gamma \rangle \quad (5)$$

where $P \in \mathbb{F}_2^n$ is the plaintext and $z_\gamma \in \mathbb{F}_2$. Assume that $z_\gamma = 0$ holds with probability p , i.e., $\Pr_{P \in \mathbb{F}_2^n} [z_\gamma = 0] = p$. As a result, the correlation of $\Delta \rightarrow \gamma$ is:

$$\text{Cor}_{P \in \mathbb{F}_2^n} [z_\gamma] = \frac{1}{|2^n|} \sum_{P \in \mathbb{F}_2^n} (-1)^{z_\gamma} = 2p - 1. \quad (6)$$

For convenience, denote by $G(z_\gamma)$ the correlation, i.e., $G(z_\gamma) = 2p - 1$.

Suppose that $G(z_{\gamma_i})$ is the correlation of $\Delta \rightarrow \gamma_i$ (see Fig. 3). Under the assumption that the two variables z_{γ_1} and z_{γ_2} (resp. z_{γ_4} and z_{γ_5}) are independent, we have

$$G(z_{\gamma_3}) = G(z_{\gamma_1}) \times G(z_{\gamma_2}), \quad G(z_{\gamma_6}) = G(z_{\gamma_4}) \times G(z_{\gamma_5}), \quad (7)$$

due to the piling-up lemma [24]. If $|G(z_{\gamma_i})| > c, i \in \{1, 2\}$ and $|G(z_{\gamma_j})| < c, i \in \{4, 5\}$ hold simultaneously where c is a threshold, then $|G(z_{\gamma_3})| > |G(z_{\gamma_6})|$ will hold too.

Thus, there is a heuristic conclusion: compared with two differential-linear approximations with a low absolute correlation, two ones with a high absolute correlation would be more likely to generate another relatively good differential-linear approximation. Although the assumption that the two variables are independent may not hold in a real scenario, this conclusion still shows a surprisingly positive influence in the support experiment introduced in Section 3.2.

3.2 Iterative Search

Based on the heuristic conclusion, an iterative search algorithm is designed to search for differential-linear approximations $\Delta_m \xrightarrow{E_m} \gamma_m$ (see Fig. 1).

The iterative search algorithm contains two phases:

1. **Initialization phase:** Preset a difference Δ_m and a threshold c . Then select t differential-linear approximations $\Delta_m \rightarrow \gamma_i$ with an absolute correlation higher than c , and add the linear masks $\gamma_i, i \in \{1, \dots, t\}$ to a pool \mathcal{P} .
2. **iterative phase:** at the beginning of each iteration, traverse each possible tuple (γ_i, γ_j) where γ_i, γ_j are two different linear masks picked from the pool. If $\gamma_i \oplus \gamma_j \notin \mathcal{P}$ and the absolute correlation of $\Delta_m \rightarrow \gamma_i \oplus \gamma_j$ exceeds the threshold, add $\gamma_i \oplus \gamma_j$ to the pool.

Since the XOR operation (i.e., $\gamma_i \oplus \gamma_j$) is a linear operation, the search space of the iterative algorithm is decided by the t linear masks added to the pool in the initialization phase. Denote by $\vec{\gamma}_i$ the row vector transformed from the linear mask γ_i , i.e., $\vec{\gamma}_i[j] = \gamma_i[n - 1 - j]$. Let $\boldsymbol{\gamma} = [\vec{\gamma}_1, \dots, \vec{\gamma}_t]^\top$ denote the $t \times n$ matrix composed of the t vectors. Apparently, the search space is decided by the rank of $\boldsymbol{\gamma}$. If the rank is t , i.e., the t row vectors are linearly independent, the size of the search space of the iterative algorithm is 2^t .

To ensure the vectors corresponding to the t linear masks are linearly independent, we focus on linear masks of Hamming weight 1 in the initialization phase. For the convenience of further introducing the iterative algorithm, two concepts named *strong unbalanced bit* and *weak unbalanced bit* are adopted, see **Definition 1**.

Definition 1 For a preset difference Δ and a threshold c , suppose that the linear mask is $\gamma = [i], i \in \{0, \dots, n - 1\}$ where n is the blocksize, if the absolute correlation of the differential-linear approximation $\Delta \rightarrow \gamma$ exceeds c , the i -th ciphertext bit is called a **strong unbalanced bit**. Otherwise, it is called a **weak unbalanced bit**.

Suppose that \mathcal{B}_S is the strong unbalanced bit set and \mathcal{B}_W is the weak unbalanced bit set for the given difference Δ . In the initialization phase, all the linear masks $[i]$ for $i \in \mathcal{B}_S$ are added to the pool. Then the size of the search space is $2^{|\mathcal{B}_S|}$. Algorithm 1 summarizes the iterative search algorithm.

Algorithm 1 Search $\Delta_m \xrightarrow{E_m} \gamma_m$ in an iterative way

Require: A difference, Δ_m ; A threshold, c ; Number of iterations, Ni ; Sample size, N .

Ensure: A list of linear masks $\gamma_m \in \mathbb{F}_2^n$.

- 1: $\mathcal{P} \leftarrow []$;
- 2: Randomly generate N plaintexts $P_i, i \in \{1, \dots, N\}$;
- 3: Collect N ciphertext pairs $(E_m(P_i), E_m(P_i \oplus \Delta_m))$ for $i \in \{1, \dots, N\}$.
- 4: **Initialization phase:**
- 5: **for** $i \in \{0, \dots, n-1\}$ **do**
- 6: $\gamma_m \leftarrow [i]$;
- 7: Compute the correlation **Cor** of $\Delta_m \rightarrow \gamma_m$ over the N ciphertext pairs.
- 8: **if** $|\mathbf{Cor}| \geq c$ **then**
- 9: Add $[i]$ to \mathcal{P} ;
- 10: **end if**
- 11: **end for**
- 12: **Iterative phase:**
- 13: **for** $i \in \{1, \dots, Ni\}$ **do**
- 14: $\mathcal{Q} \leftarrow []$;
- 15: Traverse each possible tuple (γ_1, γ_2) where $\gamma_1, \gamma_2 \in \mathcal{P}$. If $\gamma_1 \oplus \gamma_2 \notin \mathcal{P} \cup \mathcal{Q}$, compute the correlation of $\Delta_m \rightarrow \gamma_1 \oplus \gamma_2$ over the N ciphertext pairs. If the absolute correlation exceeds c , add $\gamma_1 \oplus \gamma_2$ to \mathcal{Q} ;
- 16: $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{Q}$; /*Merge two sets \mathcal{P} and \mathcal{Q} */
- 17: **end for**

A support experiment. We have performed an experiment on two ciphers (i.e., 8-round LEA and 5-round Speck32) to support the previous heuristic conclusion and verify the iterative search, i.e., Algorithm 1.

Consider an encryption function $E_m : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ and an input difference Δ_m . Set an absolute correlation threshold c to identify the strong (weak) unbalanced bit set \mathcal{B}_S (\mathcal{B}_W). We focus on differential-linear approximations $\Delta_m \xrightarrow{E_m} \gamma_m$ with an absolute correlation higher than the threshold c , and are interested in their distributions in two spaces \mathcal{X}_1 and \mathcal{X}_2 , where \mathcal{X}_1 and \mathcal{X}_2 are defined as:

$$\mathcal{X}_1 = \{\Delta_m \xrightarrow{E_m} \gamma_m \mid 0 < HW(\gamma_m) \leq d\},$$

$$\mathcal{X}_2 = \{\Delta_m \xrightarrow{E_m} \gamma_m \mid 0 < HW(\gamma_m) \leq d; \gamma_m[i] = 0 \text{ for } i \notin \mathcal{B}_S\},$$

where $HW(\gamma_m)$ denotes the Hamming weight of γ_m , and d is the Hamming weight threshold. We traverse the space \mathcal{X}_1 and denote by \mathcal{G} all the found differential-linear approximations with an absolute correlation higher than c .

Table 3. Comparison of differential-linear approximations in two spaces.

E_m	n	Δ_m	c	$ \mathcal{B}_S $	d	$ \mathcal{X}_1 $	$ \mathcal{X}_2 $	$ \mathcal{G} $	$ \mathcal{G} \cap \mathcal{X}_2 $	$\frac{ \mathcal{G} }{ \mathcal{X}_1 }$	$\frac{ \mathcal{G} \cap \mathcal{X}_2 }{ \mathcal{X}_2 }$
8-round LEA	128	[31]	2^{-4}	14	2	8256	105	72	43	0.0087	0.4095
5-round Speck32	32	[22]	2^{-4}	10	4	41448	385	785	311	0.0189	0.8078
					3	5488	175	250	146	0.0456	0.8343

¹ $|X|$: the size of set X ; $X \cap Y$: the intersection of sets X and Y .

$$^2 |\mathcal{X}_1| = \sum_{i \in \{1, \dots, d\}} \binom{n}{i}; |\mathcal{X}_2| = \sum_{i \in \{1, \dots, d\}} \binom{|\mathcal{B}_S|}{i}.$$

At first, we let $c = 2^{-4}$, set $d = 2$ for 8-round LEA and $d = 4$ for 5-round Speck32. If we set a larger d , the experiment takes too long. Table 3 summarizes the settings and experiment results. For 8-round LEA (respectively, 5-round Speck32/64), we find a total of 72 (resp. 785) differential-linear approximations with a correlation higher than 2^{-4} , and 43 (resp. 311) out of which belong to the space \mathcal{X}_2 . Thus, if we randomly pick a sample $x \in \mathcal{X}_1$, the probability that $x \in \mathcal{G}$ is 0.0087 (resp. 0.0189). However, if we randomly pick a sample $x \in \mathcal{X}_2$, the corresponding probability is $0.4095 \approx 47 \times 0.0087$ (resp. $0.8078 \approx 42 \times 0.0189$). The advantage is extremely significant ². We wonder whether increasing d will weaken the advantage. Thus, we set $d = 3$ for 5-round Speck32 and perform the experiment again. Interestingly, the advantage becomes even more significant when d increases from 3 to 4, due to the size (i.e., $|\mathcal{X}_1|$) of the first space \mathcal{X}_1 increasing too sharply (see the second and third row of Table 3).

Next, we run Algorithm 1 by setting the same difference Δ_m (i.e., [31] for 8-round LEA and [22] for 5-round Speck32/64) and threshold $c = 2^{-4}$. Finally, for 8-round LEA (respectively, 5-round Speck32/64), all the 43 (resp. 311) differential-linear approximations are found within $Ni = 1$ (resp. 2) iterations.

3.3 Meet-in-the-Middle Search

Consider $E_2 \circ E_m$ and a given difference Δ_m . Based on each linear mask γ_m returned by Algorithm 1, one can search linear approximations $\gamma_m \xrightarrow{E_2} \gamma_{\text{out}}$ using an automatic search tool (e.g., [31]), and obtain a differential-linear approximation $\Delta_m \xrightarrow{E_2 \circ E_m} \gamma_{\text{out}}$ by connecting $\Delta_m \xrightarrow{E_m} \gamma_m$ and $\gamma_m \xrightarrow{E_2} \gamma_{\text{out}}$.

However, Algorithm 1 is time-consuming. Moreover, in distinguishing attacks or key recovery attacks, there are usually some requirements about the linear approximation $\gamma_m \rightarrow \gamma_{\text{out}}$, such as the upper bound of the correlation or the Hamming weight of γ_{out} . If the value of γ_m is fixed to a concrete value in advance, it is likely to be inefficient to find linear approximations $\gamma_m \rightarrow \gamma_{\text{out}}$ satisfying the requirements.

² The support experiment is also performed on 5-round PRESENT and 4-round DES, and the advantage is significant too. More details are available at https://github.com/AI-Lab-Y/DLA_search_and_partition_tree

Hence, a meet-in-the-middle search algorithm is designed to search differential-linear approximations $\Delta_m \xrightarrow{E_2 \circ E_m} \gamma_{\text{out}}$. At a high-level view, we first search the linear approximations $\gamma_m \xrightarrow{E_2} \gamma_{\text{out}}$ without setting the value of γ_m . For each returned γ_m , check whether it is in the list returned by Algorithm 1.

Two tricks are applied to reduce the time consumption of the meet-in-the-middle search. First, in order to accelerate the matching of linear mask γ_m , we add the following necessary conditions when searching $\gamma_m \rightarrow \gamma_{\text{out}}$:

$$\gamma_m[i] = 0, \quad i \in \mathcal{B}_{\mathcal{W}} \quad (8)$$

where $\mathcal{B}_{\mathcal{W}}$ is the *weak unbalanced bit* set. Second, compute the experimental correlation of $\Delta_m \rightarrow \gamma_m$ after a linear approximation $\gamma_m \rightarrow \gamma_{\text{out}}$ is returned, instead of running Algorithm 1 to get the list of γ_m at the beginning. Algorithm 2 summarizes the meet-in-the-middle search algorithm.

Algorithm 2 Search $\Delta_m \xrightarrow{E_m} \gamma_m \xrightarrow{E_2} \gamma_{\text{out}}$ in a meet-in-the-middle way

Require: A difference, Δ_m ; A threshold, c ; Sample size, N .

Ensure: A list of linear mask tuples $(\gamma_m, \gamma_{\text{out}})$ where $\gamma_m, \gamma_{\text{out}} \in \mathbb{F}_2^n$.

```

1:  $\mathcal{P} \leftarrow []$ ;  $\mathcal{B}_{\mathcal{S}} \leftarrow []$ ;
2: Randomly generate  $N$  plaintexts  $P_i, i \in \{1, \dots, N\}$ ;
3: Collect  $N$  pseudo-ciphertext pairs  $(E_m(P_i), E_m(P_i \oplus \Delta_m))$  for  $i \in \{1, \dots, N\}$ .
4: Stage 1 (initialization phase in Algorithm 1):
5: for  $i \in \{0, \dots, n-1\}$  do
6:    $\gamma_m \leftarrow [i]$ ;
7:   Compute the correlation Cor of  $\Delta_m \rightarrow \gamma_m$  over the  $N$  pseudo-ciphertext pairs.
8:   if  $|\mathbf{Cor}| \geq c$  then
9:     Add  $i$  to  $\mathcal{B}_{\mathcal{S}}$ ;
10:  end if
11: end for
12: Stage 2 (search  $\gamma_m \rightarrow \gamma_{\text{out}}$ ):
13: for  $i \in \{0, \dots, n-1\}$  and  $i \notin \mathcal{B}_{\mathcal{S}}$  do
14:   Add a condition  $\gamma_m[i] = 0$  to Model() ; /*Model() is the automatic search
      model of linear approximations  $\gamma_m \rightarrow \gamma_{\text{out}}$  */
15: end for
16: Collect linear mask tuples  $(\gamma_m, \gamma_{\text{out}})$  by running Model(); /* filter tuples with
      duplicate mask  $\gamma_m$  before Stage 3 */
17: Stage 3 (compute the correlation of  $\Delta_m \rightarrow \gamma_m$ ):
18: for each returned tuple  $(\gamma_m, \gamma_{\text{out}})$  do
19:   Compute the correlation Cor of  $\Delta_m \rightarrow \gamma_m$  over the  $N$  pseudo-ciphertext pairs.
20:   if  $|\mathbf{Cor}| \geq c$  then
21:     Add  $(\gamma_m, \gamma_{\text{out}})$  to  $\mathcal{P}$ ;
22:   end if
23: end for

```

Remark 1. In order to apply Algorithm 2, one needs to determine the number of rounds (denoted by r_m) covered by the middle part E_m . If r_m is too large, the strong unbalanced bit set \mathcal{B}_S corresponding to a given difference Δ_m may be an empty set, which will reduce the search space of Algorithm 2.

To avoid this problem and take full advantage of Algorithm 2, we provide an empirical rule on setting r_m as follows. Consider a pre-defined set (denoted by \mathcal{D}_m) of differences Δ_m to be searched. For most (for example, three-fourths) elements $\Delta_m = v \in \mathcal{D}_m$, the setting r_m should make sure that the corresponding strong unbalanced bit sets \mathcal{B}_{S_v} are non-empty sets. This is based on our experience from searching for differential-linear approximations of LEA and Speck.

4 Partition Tree

In this section, we present a generic tool named *partition tree* to generate partitions for ARX encryption functions, starting from basic concepts.

4.1 Basic Concepts

A partition tree is a tree that describes the partition conditions and approximations simultaneously. In a partition tree, each non-leaf node stands for one object whose value is unknown, and each leaf node stands for one object whose value is known.

To build such a tree, some necessary concepts are proposed.

- *Partition Edge:* Denote by $A \dashrightarrow B$ (with a red dashed arrow) a partition edge, which means that we need to divide all the data into multiple partitions according to the value of B , for computing the value of A .
- *Approximation Edge:* Denote by $A \xrightarrow{X} B$ (with a black arrow) an approximation edge, which means that $A = X \oplus B$ where the value of X is known and the value of B is unknown. Similarly, the approximation edge $A \rightarrow B$ stands for $A = B$.

4.2 Building Process and Usage

In this subsection, we introduce the process of building a partition tree, and show how to derive the final partitions from the partition tree. For a more intuitive and deeper understanding, partition trees related to two encryption functions are first presented directly.

Partition trees related to two encryption functions. The first encryption function is the single modular addition as shown in Fig. 2, i.e., $y = z \boxplus x$. Again, we are interested in the value of $z[i]$ or $z[i, i-1]$. Three partition trees are shown in Fig. 4.

The second encryption function is the two consecutive modular additions as shown in the left picture in Fig. 5. The right picture shows a partition tree related

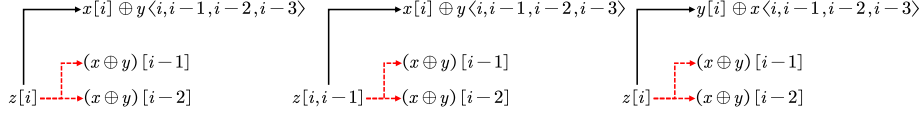


Fig. 4. Partition trees related to one modular addition. The left and middle partition trees correspond to Lemma 1. The right one corresponds to Lemma 2.

to it. In [4,5], the authors have presented the final partitions and approximations of $z_1[i]$. Later, we will explain how to obtain the results presented in [4,5] using the partition tree.

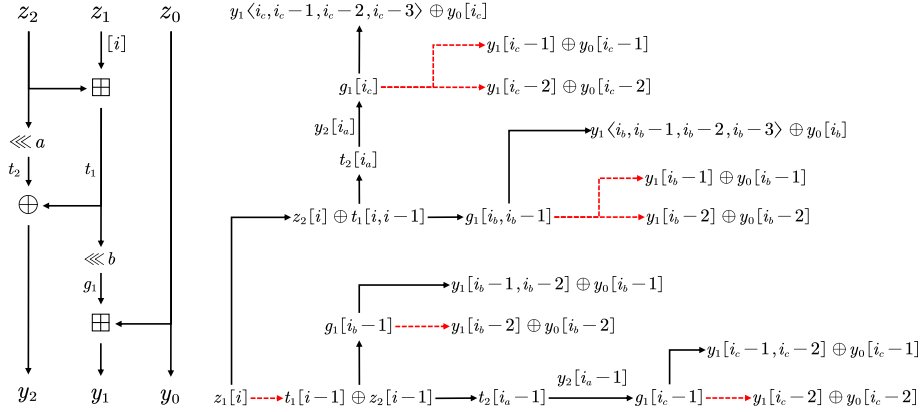


Fig. 5. Partition tree for two consecutive modular additions. The left picture shows the encryption function. The right picture shows the partition tree. Notice that $i_a = (i + a) \bmod n$, $i_b = (i + b) \bmod n$, $i_c = (i + a + b) \bmod n$, and n is the word size.

Building process. At a high level, we build a partition tree by simulating the propagation of linear approximation. First, create a root node standing for the target bit (e.g. $z[i], z[i, i-1]$ in Fig. 4). Second, propagate the approximation. When meeting a modular addition, we choose partition conditions (eg. $(x \oplus y)[i-1]$ and $(x \oplus y)[i-2]$ in Fig. 4) for creating partition edges, and choose an approximation for creating an approximation edge. The available partition conditions and approximations refer to Lemma 1 and Lemma 2. When meeting other linear operations, we just need to create approximation edges. In an iterative way, we expand the tree until each leaf node is only related to objects whose values are known.

When meeting a modular addition, one can flexibly choose the partition conditions and approximations. For example, we always choose two partition conditions and an approximation with an indeterministic linear mask in Fig. 4. In Fig. 5, for computing $z_1[i]$, we choose only one partition condition and a deterministic approximation. The choice depends on the concrete encryption function.

Usage. Once a partition tree is built, we can obtain the final partitions and approximations. Take the partition tree as shown in Fig. 5 as an example.

In the partition tree, there are 7 partition edges that contain five different partition conditions. The related five values are v_i for $i \in \{1, 2, 3, 4, 5\}$:

$$\begin{aligned}
v_1 &= t_1[i-1] \oplus z_2[i-1] \\
&= (y_2[i_a-1] \oplus y_1[i_b-2] \oplus y_1[i_c-2]) \oplus \\
&\quad (y_1[i_b-1] \oplus y_0[i_b-1]) \oplus (y_1[i_c-1] \oplus y_0[i_c-1]); \quad (9) \\
v_2 &= y_1[i_b-1] \oplus y_0[i_b-1]; \quad v_3 = y_1[i_b-2] \oplus y_0[i_b-2]; \\
v_4 &= y_1[i_c-1] \oplus y_0[i_c-1]; \quad v_5 = y_1[i_c-2] \oplus y_0[i_c-2].
\end{aligned}$$

Since the first value is related to v_2 and v_4 , the authors in [4, 5] transformed it into a simplified form, i.e., $v_1 = y_2[i_a-1] \oplus y_1[i_b-2] \oplus y_1[i_c-2]$. As a result, the data is split into 2^5 partitions according to the value $v_1||v_2||v_3||v_4||v_5$.

Traverse the partition tree from the root node along with approximation edges, and we have

$$\begin{aligned}
z_1[i] \approx \langle \gamma, & y_2[i_a]||y_0[i_b]||y_1[i_b]||y_1[i_b-1]||y_1[i_b-2]||y_1[i_b-3]|| \\
& y_0[i_c]||y_1[i_c]||y_1[i_c-1]||y_1[i_c-2]||y_1[i_c-3] \rangle.
\end{aligned}$$

Next, we determine γ for each partition with the help of the partition tree and the value $v_1||v_2||v_3||v_4||v_5$. Finally, the corresponding correlation is estimated experimentally. Readers can verify the above analysis by referring to the work in [5] and [4].

4.3 Dynamic Partitioning Technique

To make the key recovery attack achieve a good performance, for a given encryption function, the correlation corresponding to each partition should be non-zero [4]. When the encryption function is relatively complex, we may need to dynamically choose the partition conditions for each data to achieve the target.

In this section, we introduce the dynamic partitions for parallel modular additions as depicted in Fig. 6, which are used in attacks on LEA.

Partition tree for three parallel modular additions. At first, we are interested in the value $z_3[i]$. Fig. 7 shows a partition tree.

The data is split into 64 partitions in the following way

$$\mathcal{T}_{b_0b_1b_2b_3b_4b_5} = \{(z_0, y_0, y_1, y_2, k_0, k_1, k_2) \in (\mathbb{F}_2^n)^7 \mid b_0b_1b_2b_3b_4b_5 = v_1||v_2||v_3||v_4||v_5||v_6\},$$

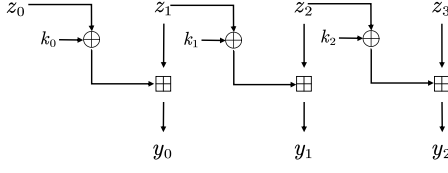


Fig. 6. Parallel modular additions. We assume that the values of the three keys k_0, k_1, k_2 , and the input word z_0 are known. The values of the other three input words z_1, z_2, z_3 are unknown.

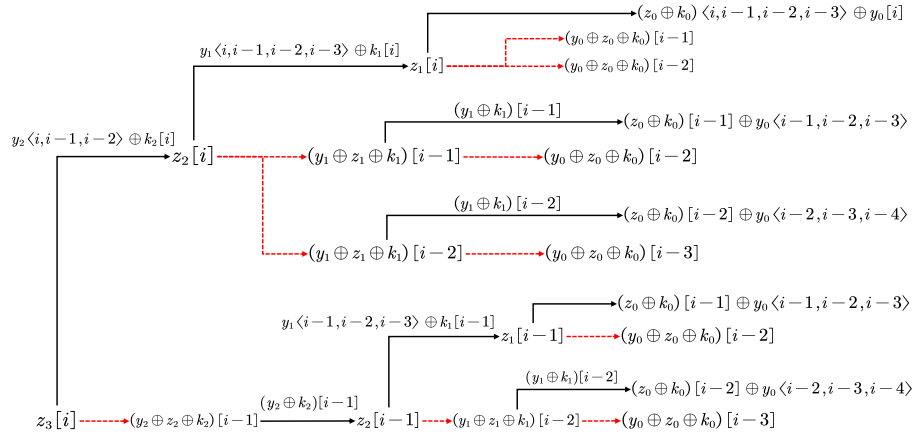


Fig. 7. A partition tree for three parallel modular additions.

where the six partition conditions v_i for $i \in \{1, \dots, 6\}$ are:

$$\begin{aligned}
v_1 &= (y_0 \oplus z_0 \oplus k_0)[i-1]; \\
v_2 &= (y_0 \oplus z_0 \oplus k_0)[i-2]; \quad v_3 = (y_0 \oplus z_0 \oplus k_0)[i-3]; \\
v_4 &= \begin{cases} (y_1 \oplus k_1)[i-1] \oplus y_0[i-2], & \text{if } v_2 = 1; \\ (y_1 \oplus k_1)[i-1] \oplus y_0[i-3], & \text{if } v_2 = 0. \end{cases} \\
v_5 &= \begin{cases} (y_1 \oplus k_1)[i-2] \oplus y_0[i-3], & \text{if } v_3 = 1; \\ (y_1 \oplus k_1)[i-2] \oplus y_0[i-4], & \text{if } v_3 = 0; \end{cases} \\
v_6 &= \begin{cases} (y_2 \oplus k_2)[i-1] \oplus y_1[i-2], & \text{if } v_2 \oplus v_5 = 0; \\ (y_2 \oplus k_2)[i-1] \oplus y_1[i-3], & \text{if } v_2 \oplus v_5 = 1; \end{cases}
\end{aligned} \tag{10}$$

The approximation of $z_3[i]$ is

$$\begin{aligned}
z_3[i] \approx & \langle \gamma, y_2[i] \| y_2[i-1] \| y_2[i-2] \| y_1[i] \| y_1[i-1] \| y_1[i-2] \| y_1[i-3] \| \\
& y_0[i] \| z_0[i] \| z_0[i-1] \| z_0[i-2] \| z_0[i-3] \| k_2[i] \| k_1[i] \| k_0[i] \| \\
& k_0[i-1] \| k_0[i-2] \| k_0[i-3] \rangle,
\end{aligned}$$

where γ and corresponding correlation **Cor** are summarized in Table 10. The average absolute correlation is $2^{-0.868}$.

To identify the belonging partition for data, one needs to dynamically choose the approximations of v_4, v_5 , and v_6 . If any one of v_4, v_5, v_6 is computed using a fixed approximation, the correlation corresponding to some partitions will be 0.

Partition trees for two parallel modular additions. In the key recovery attacks introduced later, we are interested in the value $z_2[i]$ too. We directly adopt the subtree (see Fig. 7) that takes $z_2[i]$ as the root node.

The data is split into 32 partitions in the following way:

$$\mathcal{T}_{b_0b_1b_2b_3b_4} = \{(z_0, y_0, y_1, k_0, k_1) \in (\mathbb{F}_2^n)^5 \mid b_0b_1b_2b_3b_4 = v_1 \parallel v_2 \parallel v_3 \parallel v_4 \parallel v_5\},$$

where the five values v_i for $i \in \{1, \dots, 5\}$ are shown in formula 10.

The approximation of $z_2[i]$ is

$$z_2[i] \approx \langle \gamma, y_1[i] \parallel y_1[i-1] \parallel y_1[i-2] \parallel y_1[i-3] \parallel y_0[i] \parallel z_0[i] \parallel z_0[i-1] \parallel z_0[i-2] \parallel z_0[i-3] \parallel k_1[i] \parallel k_0[i] \parallel k_0[i-1] \parallel k_0[i-2] \parallel k_0[i-3] \rangle,$$

where γ and corresponding correlation **Cor** are summarized in Table 11. The average absolute correlation is $2^{-0.452}$.

Partition tree makes the process of generating partitions and approximations more efficient and easier to understand, which breaks the barrier to applying the partitioning idea to ARX ciphers.

5 Overview of Our Attack Framework

This paper mainly focuses on the attack framework which is first proposed in [5] at CRYPTO 2020 and further improved in [4]. The framework has a constraint on how the secret key to be recovered is operated in the encryption process, which is removed in this section. Fig. 8 (the left picture) shows our attack framework. Here the encryption function F uses the secret key k to encrypt its input. In [4,5], the secret key is used as the last whitening key (see the right picture), which is the core difference. Our aim is to recover parts of the secret key k by using a distinguisher $\Delta_{\text{in}} \rightarrow \gamma_{\text{out}}$.

In the following, we assume that the ciphertext-key (i.e., the concatenation $c \parallel k$ of the ciphertext and key) space \mathbb{F}_2^n is split into a direct sum $\mathcal{P} \oplus \mathcal{R}$ with $n_{\mathcal{P}} := \dim \mathcal{P}$ and $n_{\mathcal{R}} := n - n_{\mathcal{P}}$, so that the partitions will be given by the cosets $\mathcal{T}_{p_i} = p_i \oplus \mathcal{R}$ for any $p_i \in \mathcal{P}$ (i.e., p_i represents a set of the partition). Therefore, we can uniquely express $c \parallel k$ as $(c \parallel k)_{\mathcal{P}} \oplus (c \parallel k)_{\mathcal{R}}$, where $(c \parallel k)_{\mathcal{P}} \in \mathcal{P}$ and $(c \parallel k)_{\mathcal{R}} \in \mathcal{R}$. Then, for any $p_i \in \mathcal{P}$, the partition $\mathcal{T}_{p_i} \subset \mathbb{F}_2^n$ is defined as $\mathcal{T}_{p_i} = \{(c \parallel k) \in \mathbb{F}_2^n \mid (c \parallel k)_{\mathcal{P}} = p_i\}$.

The idea is to identify several tuples $(\mathcal{T}_{p_i}, \lambda^{(p_i)})$, $i \in \{1, \dots, s\}$, for which we can observe a high absolute correlation

$$\varepsilon_i := \mathbf{Cor}_{(c \parallel k) \in \mathcal{T}_{p_i}} \left[\langle \gamma_{\text{out}}, z \rangle \oplus \langle \lambda^{(p_i)}, c \parallel k \rangle \right],$$

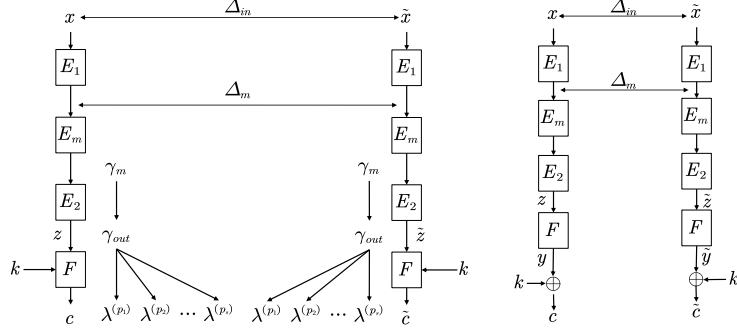


Fig. 8. The schematic diagram of the key recovery attack framework.

where $\lambda^{(p_i)}$ is the linear mask of $c||k$ when $(c||k)_{\mathcal{P}} = p_i$. We finally observe the following correlation for (c, \tilde{c}) by guessing the secret key k , and the final correlation is defined as

$$\rho_{i,j} := \mathbf{Cor}_{(c||k, \tilde{c}||k) \in \mathcal{T}_{p_i} \times \mathcal{T}_{p_j}} \left[\langle \lambda^{(p_i)}, c||k \rangle \oplus \langle \lambda^{(p_j)}, \tilde{c}||k \rangle \right]. \quad (11)$$

According to the analysis in [4], the correlation $\rho_{i,j}$ can be estimated as

$$\rho_{i,j} = \varepsilon_i \varepsilon_j \mathbf{Cor}_{(x, \tilde{x}) \in (\mathbb{F}_2^n)^2} [\langle \gamma_{out}, z \oplus \tilde{z} \rangle]. \quad (12)$$

Next, let us consider the differential-linear attack using N pairs. Let (c, \tilde{c}) be the l th pair and assume that c and \tilde{c} belong to the i th and j th partitions, respectively, i.e., $(c||k)_{\mathcal{P}} = p_i$ and $(\tilde{c}||k)_{\mathcal{P}} = p_j$ (for ease of notation, we do not make the dependency of c, \tilde{c}, i, j on l explicit). Then we get the 1-bit representation

$$w_l = \langle \lambda^{(p_i)}, c||k \rangle \oplus \langle \lambda^{(p_j)}, \tilde{c}||k \rangle$$

and let $C_l = \rho_{i,j}$ where $\rho_{i,j}$ is the correlation for the l th pair when the i th and j th partitions are used for this pair.

Compute the following log-likelihood ratio (LLR) statistic:

$$\text{LLR} = \frac{1}{2} \sum_{l=1}^N \ln(1 - C_l^2) + \frac{1}{2} \sum_{l=1}^N \ln \left(\frac{1 + C_l}{1 - C_l} \right) (-1)^{w_l}. \quad (13)$$

According to the analysis in [4], the LLR statistic follows normal distribution $\mathcal{N}(\mu_0, \sigma_0^2)$ and $\mathcal{N}(\mu_1, \sigma_1^2)$ when the correct and wrong keys are guessed, respectively. The corresponding distribution parameters are

$$\mu_0 = -\mu_1 = \frac{1}{2} \sum_{l=1}^N C_l^2 = \frac{N}{2} \bar{C}, \quad \sigma_0^2 = \sigma_1^2 = \sum_{l=1}^N C_l^2 = N \bar{C}.$$

Thus, one can recover the correct key by distinguishing between $\mathcal{N}(\mu_0, \sigma_0^2)$ and $\mathcal{N}(\mu_1, \sigma_1^2)$.

To compute the LLR statistic, we need to calculate C_l and w_l . At first, the key denoted by $k_{\mathcal{P}}$ is guessed to identify the partition. After guessing $k_{\mathcal{P}}$, the first term of the LLR statistic, i.e., $\alpha := \frac{1}{2} \sum_{l=1}^N \ln(1 - C_l^2)$, is constant and independent of w_l . Then, we may need to guess some key bits denoted by $k_{\mathcal{L}}$ to compute w_l , and further compute the second term of the LLR statistic.

Divide the linear mask $\lambda^{(p_i)}$ and $\lambda^{(p_j)}$ into two parts, i.e., $\lambda^{(p_i)} = g^{(p_i)} \oplus \Gamma^{(p_i)}$ and $\lambda^{(p_j)} = g^{(p_j)} \oplus \Gamma^{(p_j)}$, such that

$$\langle \lambda^{(p_i)}, c \oplus k \rangle = \langle g^{(p_i)}, c \rangle \oplus \langle \Gamma^{(p_i)}, k \rangle, \quad \langle \lambda^{(p_j)}, \tilde{c} \oplus k \rangle = \langle g^{(p_j)}, \tilde{c} \rangle \oplus \langle \Gamma^{(p_j)}, k \rangle, \quad (14)$$

then $k_{\mathcal{L}}$ is determined by $\Gamma^{(p_i)} \oplus \Gamma^{(p_j)}$. The size of $k_{\mathcal{L}}$ is $\dim W$ where a linear subspace W is defined by $W := \text{Span}\{\Gamma^{(p_i)} \oplus \Gamma^{(p_j)} \mid i, j \in \{1, \dots, s\}\}$.

The computation of the second term of the LLR statistic is further accelerated by using the Fast Walsh-Hadamard transform (FWHT) [13], like the work in [4]. In some cases, due to the XOR operation $k \oplus \chi$ in F where k is the secret key and χ is an intermediate state, the linear masks $g^{(p_i)}$ and $g^{(p_j)}$ can be further divided into two parts, i.e., $g^{(p_i)} = \phi^{(p_i)} \oplus \Gamma^{(p_i)}$, and $g^{(p_j)} = \phi^{(p_j)} \oplus \Gamma^{(p_j)}$.

Next, we use an array β , whose element $\beta(\Gamma)$ is defined as

$$\beta(\Gamma) := \frac{1}{2} \sum_{l=1: \Gamma = \Gamma^{(p_i)} \oplus \Gamma^{(p_j)}}^N \ln \left(\frac{1 + C_l}{1 - C_l} \right) (-1)^{\langle \phi^{(p_i)} \oplus \Gamma^{(p_i)}, c \rangle \oplus \langle \phi^{(p_j)} \oplus \Gamma^{(p_j)}, \tilde{c} \rangle}.$$

Then, the second term of the LLR statistic is computed as

$$\text{LLR}' = \frac{1}{2} \sum_{l=1}^N \ln \left(\frac{1 + C_l}{1 - C_l} \right) (-1)^{w_l} = \sum_{\Gamma \in W} \beta(\Gamma) \times (-1)^{\langle \Gamma, k \rangle}.$$

Using the FWHT, one can evaluate LLR' for each $\Gamma, k \in \mathbb{F}_2^{\dim W}$ with a time complexity $\dim W \cdot 2^{\dim W}$.

Algorithm 3 summarizes the attack procedure using the FWHT. We first collect N ciphertext pairs, guess $k_{\mathcal{P}}$, and prepare a real number α and the array of real numbers β to compute the LLR statistic. For every ciphertext pair, we identify partitions, get corresponding correlation $\rho_{i,j}$ and linear mask, and update α and β accordingly. We finally apply the FWHT to β and the LLR statistic is computed as $\alpha + \tilde{\beta}(k_{\mathcal{L}})$. The overall running time is estimated as $2^{n_{\mathcal{P}}} (2N + \dim W \cdot 2^{\dim W})$, where $n_{\mathcal{P}}$ is the bit length of $k_{\mathcal{P}}$. A total of $n_{\mathcal{P}} + \dim W$ key bits are recovered. Besides, one can deduce the following proposition.

Proposition 1. [4] *After running Algorithm 3 for p^{-1} times where p is the probability of the prepended differential $\Delta_{\text{in}} \rightarrow \Delta_{\text{m}}$, the probability that the correct key is among the key candidate is*

$$p_{\text{success}} \geq \frac{1}{2} \Pr[C(k_{\mathcal{P}}, k_{\mathcal{L}}) \geq \Theta] = \frac{1}{2} \left(1 - \Phi \left(\frac{\Theta - \frac{N\bar{C}}{2}}{\sqrt{N\bar{C}}} \right) \right),$$

where Φ is the cumulative distribution of the standard normal distribution $\mathcal{N}(0, 1)$.

The expected number of wrong keys is $\frac{2^{n_{\mathcal{P}} + \dim W}}{p} \times \left(1 - \Phi \left(\frac{\Theta + \frac{N\bar{C}}{2}}{\sqrt{N\bar{C}}} \right) \right)$.

Algorithm 3 Key recovery

Require: Cipher E , sample size N , threshold Θ .

Ensure: List of key candidates $(k_{\mathcal{P}}, k_{\mathcal{L}})$ for $n_{\mathcal{P}} + \dim W$ bit of information on k .

```
1: for  $l \in \{1, \dots, N\}$  do
2:    $x \xleftarrow{\$} \mathcal{U} \oplus a$ ;
3:    $(c(l), \tilde{c}(l)) \leftarrow (E(x), E(x \oplus \Delta_{in}))$ 
4: end for
5: for each possible  $k'_{\mathcal{P}}$  do
6:    $\alpha \leftarrow 0$ 
7:   for  $\Gamma \in W$  do
8:      $\beta(\Gamma) \leftarrow 0$ 
9:   end for
10:  for  $l \in \{1, \dots, N\}$  do
11:     $(c, \tilde{c}) \leftarrow (c(l), \tilde{c}(l))$ 
12:    Compute  $(c||k)_{\mathcal{P}}$  and  $(\tilde{c}||k)_{\mathcal{P}}$  to identify partitions.
13:    Identify  $\mathcal{T}_i \times \mathcal{T}_j$  for  $((c||k)_{\mathcal{P}}, (\tilde{c}||k)_{\mathcal{P}})$  and get corresponding correlation  $\rho_{i,j}$ .
14:     $\Gamma \leftarrow \Gamma^{(c||k)_{\mathcal{P}}} \oplus \Gamma^{(\tilde{c}||k)_{\mathcal{P}}}$ 
15:     $\alpha \leftarrow \alpha + \frac{1}{2} \ln(1 - \rho_{i,j}^2)$ 
16:     $\beta(\Gamma) \leftarrow \beta(\Gamma) + \frac{1}{2} \ln \left( \frac{1 + \rho_{i,j}}{1 - \rho_{i,j}} \right) (-1)^{\langle g^{(c||k)_{\mathcal{P}}}, c \rangle \oplus \langle g^{(\tilde{c}||k)_{\mathcal{P}}}, \tilde{c} \rangle}$ 
17:  end for
18:  Compute  $\hat{\beta}$  by using the FAST Walsh-Hadamard Transform.
19:   $C(k'_{\mathcal{P}}, k'_{\mathcal{L}}) \leftarrow \alpha + \hat{\beta}(k'_{\mathcal{L}'})$ 
20:  if  $C(k'_{\mathcal{P}}, k'_{\mathcal{L}}) > \Theta$  then
21:    Save  $(k'_{\mathcal{P}}, k'_{\mathcal{L}})$  as a key candidate.
22:  end if
23: end for
```

6 Application to LEA

In 2013, the LEA family of block ciphers is published at WISA [18], expected to provide confidentiality in both high-speed and lightweight environments. In 2016, it is established as the national standard of Republic of Korea (KS X 3246). After six years of public evaluation, it is included in the ISO/IEC 29192-2:2019 standard (Information security - Lightweight cryptography - Part 2: Block ciphers). The LEA family has three members with a common block size of 128 bits and three different key sizes of 128, 192, and 256 bits, denoted by LEA-128, LEA-192, and LEA-256, respectively.

Round function. The encryption of LEA maps a plaintext of four 32-bit words $(x_0^0, x_1^0, x_2^0, x_3^0)$ into a ciphertext $(x_0^r, x_1^r, x_2^r, x_3^r)$ using a sequence of r rounds, where $r = 24$ for LEA-128, $r = 28$ for LEA-192 and $r = 32$ for LEA-256. Fig. 9 provides a schematic view of the round function of LEA.

In [18], the designers claim that the actual differential-linear characteristics should be much shorter than 14 rounds or have a bias (equal to half of the correlation) whose absolute value is significantly smaller than 2^{-57} . Moreover,

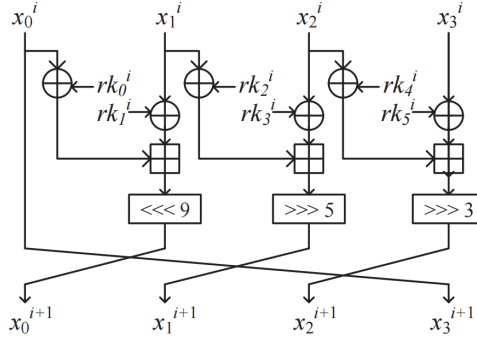


Fig. 9. The round function of LEA.

the designers claim that the secure number of rounds is 17 for LEA-128, 18 for LEA-192, and 19 for LEA-256.

6.1 Differential-Linear Approximations

We first report differential-linear approximations that are searched using the methods introduced in Section 3.

Let r_1, r_m, r_2 denote the number of rounds covered by E_1, E_m, E_2 respectively. The differential-linear approximation $\Delta_m \xrightarrow{E_m} \gamma_m$ and linear approximation $\gamma_m \xrightarrow{E_2} \gamma_{out}$ are first searched together using Algorithm 2. Then a prepend differential $\Delta_{in} \xrightarrow{E_1} \Delta_m$ is added. Since it is practically infeasible to test all the differences Δ_m , we restricted ourselves to the case of a difference Δ_m of the form $[i]$ or $[i, i + 1]$, i.e., 1-bit or consecutive 2-bit differences. We set $c = 2^{-8}$ (the correlation threshold) for Algorithm 2. Table 4 summarizes three differential-linear approximations searched by the above setting, which will be used in key recovery attacks.

Table 5 shows one of the optimal 4-round differential characteristics whose output difference is $\Delta_m = [31]$. By placing it before the 12-round or 13-round differential-linear approximation as shown in Table 4, we obtain a 16-round or 17-round distinguisher for LEA. The 17-round distinguisher with a correlation $2^{-33} \times (-2^{-6.04}) \times 2^{-10 \times 2} = -2^{-59.04}$ is the best distinguisher so far. The previous best distinguisher is the 16-round Boomerang distinguisher with a probability $2^{-117.2}$ [19].

6.2 Detecting Subspaces

For the 4-round differential characteristic as shown in Table 5, we need to detect a subspace \mathcal{U} of the input space such that $E_1(P \oplus u) \oplus E_1(P \oplus u \oplus \Delta_{in}) = \Delta_m$ for all $u \in \mathcal{U}$ if $E_1(P) \oplus E_1(P \oplus \Delta_{in}) = \Delta_m$. To detect such a subspace, we first

Table 4. Differential-Linear approximations of round reduced LEA.

(r_m, r_2)	$\Delta_m \rightarrow \gamma_m \rightarrow \gamma_{out}$	Cor of $\Delta_m \rightarrow \gamma_{out}$
(8, 3)	[26] $\rightarrow \Sigma \rightarrow [0, 9, 61, 91, 105]$	$2^{-4.679}$
(8, 4)	[31] $\rightarrow \Sigma \rightarrow [0, 9, 61, 91, 105]$	$-2^{-10.970}$
(8, 5)	[31] $\rightarrow [8, 41, 42, 73, 74] \rightarrow [0, 29, 37, 38, 61, 68, 88, 91, 101, 102, 105, 114]$	$-2^{-6.04} \times 2^{-10 \times 2}$

¹ Σ : all the possible linear masks γ_m are considered. Since many distinguishers returned by Algorithm 2 share the same linear mask γ_{out} , we directly estimate the experimental correlation of $\Delta_m \rightarrow \gamma_{out}$.

² The three correlations $2^{-4.679}$, $-2^{-10.970}$, $-2^{-6.04}$ are estimated again using N plaintext pairs and 100 keys. The three values are the median of 100 experimental correlations. For $2^{-4.679}$ and $-2^{-6.04}$, $N = 2^{24}$. For $-2^{-10.970}$, $N = 2^{32}$. The number 2^{-10} in the third row is the correlation of the linear approximation $\gamma_m \rightarrow \gamma_{out}$.

Table 5. An 4-round differential characteristic with an output difference $\Delta_m = [31]$.

r	Δ	Pr
0	(0x8a000080, 0x80402080, 0x80402210, 0xc0402234)	
1	(0x80400014, 0x80000014, 0x88000004, 0x8a000080)	2^{-17}
2	(0x80000000, 0x80400000, 0x80400010, 0x80400014)	2^{-10}
3	(0x80000000, 0x80000000, 0x80000000, 0x80000000)	2^{-6}
4	[31]	1

collect 2^{10} plaintext pairs conforming to the differential characteristic using the fast method introduced in [14].

The subspace is searched as follows: (1) Traverse $i \in \{0, \dots, 127\}$, flip the i th bit of each pair (P, \bar{P}) and compute the probability \mathbf{Pr} that the new pair $(P \oplus [i], \bar{P} \oplus [i])$ still conforms to the 4-round differential characteristic. If $\mathbf{Pr} \geq 0.7$, save i as a basis element. (2) Traverse each possible tuple (i_1, \dots, i_k) for $k \leq 3$ where $i_j, j \in \{1, \dots, k\}$ is not a basis element, and check whether $[i_1, \dots, i_k]$ can be a basis element by flipping the k bits of each pair simultaneously.

Table 6 summarizes the result of the search. Given any plaintext pair $(P, P \oplus \Delta_{in})$ conforming to the 4-round differential characteristic as shown in Table 5, using the 34 basis elements, one can create from the plaintext pair a plaintext structure consisting of 2^{34} plaintext pairs. These 2^{34} plaintext pairs are expected to pass the differential characteristic together, with a theoretical probability $2^{-3.7}$ under the assumption that the effects of the 34 basis elements are independent. For verifying the theoretical probability $2^{-3.7}$, we generate 2^{10} plaintext pairs conforming to the 4-round differential characteristic, and find that the empirical probability is $2^{-3.18}$ (resp. $2^{-3.17}$, $2^{-3.33}$) for LEA-128 (resp.

Table 6. Probability that adding one basis element doesn't affect the output difference.

Probability	Basis	Number
$\Pr = 1$	[2], [4, 36], [22, 54, 86], [31]	4
$0.9 \leq \Pr < 1$	[14], [15], [16], [17], [18], [19], [20], [21], [50, 82], [51], [52] [63, 95], [110], [111], [112], [113], [114], [124], [125], [126], [127]	21
$0.8 \leq \Pr < 0.9$	[23], [46, 78], [53], [83], [104], [105], [115]	7
$0.7 \leq \Pr < 0.8$	[24], [47, 79]	2

LEA-192, LEA-256)³. Thus, we obtain Lemma 3. The 17-round key recovery attack introduced later uses this linear subspace.

Lemma 3. *There is a set $\mathcal{X} \subseteq \mathbb{F}_2^{128}$ of size $2^{128-33-3.7}$ and a 34-dimensional linear subspace \mathcal{U} , such that for any element $x \in \mathcal{X}$ and any $u \in \mathcal{U}$ it holds that $E_1(x \oplus u) \oplus E_1(x \oplus u \oplus \Delta_{in}) = \Delta_m$ where E_1 denotes 4 rounds of LEA.*

6.3 The 17-Round Key Recovery Attack

The 16-round differential-linear approximation introduced in Section 6.1 is used to attack 17-round LEA by guessing the last round key. Look at the LEA round function as shown in Fig. 9. For the convenience of introducing the 17-round key recovery attack, we make the following transformation:

$$\begin{aligned}
 (z_0, z_1, z_2, z_3) &:= (x_0^{16}, x_1^{16} \oplus rk_1^{16}, x_2^{16} \oplus rk_3^{16}, x_3^{16} \oplus rk_5^{16}); \\
 (k_0, k_1, k_2) &:= (rk_0^{16}, rk_1^{16} \oplus rk_2^{16}, rk_3^{16} \oplus rk_4^{16}); \\
 (y_0, y_1, y_2, y_3) &:= (x_0^{17} \ggg 9, x_1^{17} \lll 5, x_2^{17} \lll 3, x_3^{17}).
 \end{aligned} \tag{15}$$

Now, the notations are consistent with those used in Fig. 6.

Consider the linear mask is [0, 9, 61, 91, 105]. The corresponding five bits to be computed are $z_3[0]$, $z_3[9]$, $z_2[29]$, $z_1[27]$, and $z_0[9]$. For $z_3[0]$ and $z_0[9]$, there are two deterministic relation:

$$z_3[0] = (z_0 \oplus y_0 \oplus y_1 \oplus y_2)[0] \oplus (k_0 \oplus k_1 \oplus k_2)[0]; \quad z_0[9] = y_3[9]. \tag{16}$$

Notice that the key bit $(k_0 \oplus k_1 \oplus k_2)[0]$ will be canceled in the key recovery stage. Thus, only the remaining three bits need to be computed by guessing key bits and using the partitioning technique.

Table 7 summarizes the approximations and partition conditions related to $z_3[9]$, $z_2[29]$, $z_1[27]$. To identify the partition, we need to know

$$\begin{aligned}
 &s[26], s[25], s[28], s[27], g[28] \oplus y_0 \langle 27, 26 \rangle, g[27] \oplus y_0 \langle 26, 25 \rangle, \\
 &s[8], s[7], s[6], g[8] \oplus y_0 \langle 7, 6 \rangle, g[7] \oplus y_0 \langle 6, 5 \rangle, h[8] \oplus y_1 \langle 7, 6 \rangle
 \end{aligned}$$

³ The code for verifying the theoretical probability $2^{-3.7}$ is available at https://github.com/AI-Lab-Y/DLA_search_and_partition_tree

Table 7. Approximations and partition conditions related to $z_3[9]$, $z_2[29]$, $z_1[27]$.

ζ_1	Choice: $z_1[27]$ $\mathcal{P}_1 \ni p_i \cong (s[26], s[25])$ Linear: $y_0[27], z_0[27], z_0[26], z_0[25], z_0[24], k_0[27], k_0[26], k_0[25], k_0[24]$
ζ_2	Choice: $z_2[29]$ $\mathcal{P}_2 \ni p_i \cong (s[28], s[27], s[26], g[28] \oplus y_0 \langle 27, 26 \rangle, g[27] \oplus y_0 \langle 26, 25 \rangle)$ Linear: $y_1[29], y_1[28], y_1[27], y_1[26], y_0[29], z_0[29], z_0[28], z_0[27], z_0[26]$ $k_1[29], k_0[29], k_0[28], k_0[27], k_0[26]$
ζ_3	Choice: $z_3[9]$ $\mathcal{P}_3 \ni p_i \cong (s[8], s[7], s[6], g[8] \oplus y_0 \langle 7, 6 \rangle, g[7] \oplus y_0 \langle 6, 5 \rangle, h[8] \oplus y_1 \langle 7, 6 \rangle)$ Linear: $y_2[9], y_2[8], y_2[7], y_1[9], y_1[8], y_1[7], y_1[6], y_0[9], z_0[9], z_0[8],$ $z_0[7], z_0[6], k_2[9], k_1[9], k_0[9], k_0[8], k_0[7], k_0[6]$

and 12-bit key guessing is enough, where $s = z_0 \oplus k_0 \oplus y_0$ and $g = y_1 \oplus k_1$ and $h = y_2 \oplus k_2$.

After guessing the 12-bit key, we identify 3×2 partitions for a ciphertext pair, and access corresponding linear masks and correlations. Suppose that the six correlations are \mathbf{Cor}_i for $i \in \{1, \dots, 6\}$. We have

$$\rho = -2^{-10.97} \times \prod_{i=1}^6 \mathbf{Cor}_i \quad (17)$$

under the assumption that the six differential-linear approximations are independent, where ρ is used to calculate the LLR statistic for key recovery.

Experimental reports. Before estimating the complexity of the 17-round attack, we first execute a practical auxiliary experiment. Look at the two differential-linear approximations as shown in the first and second rows of Table 4. They have the same output linear mask γ_{out} . Since the correlation of $[26] \rightarrow [0, 9, 61, 91, 105]$ is very high, we use it to verify our attack procedure. The right pair and the correct key are used to observe the LLR statistic for the correct case.

The LLR statistic depends on the sum of the squared correlation $N\bar{C} = \sum_{i=1}^N c_i^2$. We estimated $\bar{C} \approx 2^{-14.052}$ and $N\bar{C} \approx 61.74$ when $N = 2^{20}$ pairs are used. Fig. 10 shows the comparison of the LLR statistics, where the theoretical distribution is drawn by the normal distribution with mean $\frac{N\bar{C}}{2}$ (for a correct case) and $-\frac{N\bar{C}}{2}$ (for a wrong case) and the standard deviation $\sqrt{N\bar{C}}$. By repeating the attack 512 times, two experimental histograms are drawn. In each trial, we ensure that the correlation of $\Delta_m \rightarrow \gamma_{\text{out}}$ approximately equals $2^{-4.679}$. Note that the experimental one is more biased than the theoretical estimation in the correct case. We expect that the reason comes from the additional auto-correlation-linear hull [4] that we do not take into account. The auxiliary experiment well verifies our attack procedure.

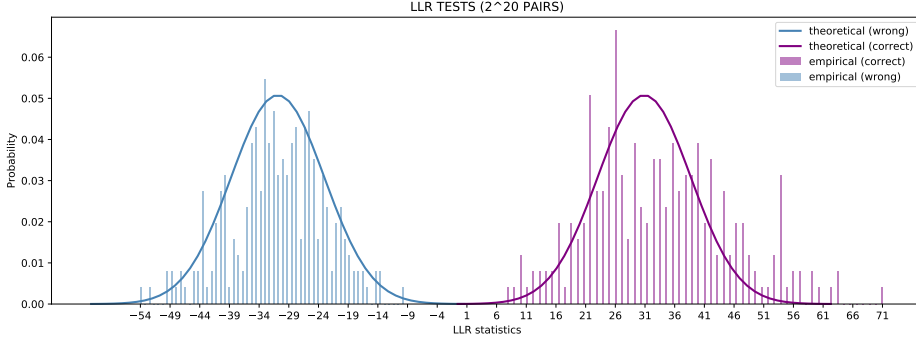


Fig. 10. Comparison with LLR statistics.

For the 17-round attack, we estimated $\bar{C} \approx 2^{-26.334}$ and $N\bar{C} \approx 50.77$ when $N = 2^{32}$ pairs are used. We finally estimate the data complexity and time complexity of this attack. To identify the partition, we need to guess the 12-bit key. We also enumerated elements of the linear subspace W and computed the basis using Gaussian elimination. As a result, the dimension of W is 9. To find a right pair, we need $2^{33+3.7}$ iterations because of Lemma 3. Thus, there are about $2^{12+9+33+3.7} = 2^{57.7}$ wrong cases. When $2^{33.2}$ pairs are used, we have $N\bar{C} \approx 116.16$. With a success probability of 90%, we can construct a 69.6-bit filter, which is enough to remove $2^{57.7}$ wrong cases. We finally estimate the time complexity by using the following formula ⁴:

$$\begin{aligned} T &= p^{-1} \times 2^{nP} \times (2N + \dim W 2^{\dim W}) \\ &= 2^{33+3.7} \times 2^{12} \times (2 \times 2^{33.2} + 9 \times 2^9) \approx 2^{82.9}. \end{aligned}$$

6.4 The 18-Round Key Recovery Attack

The 17-round differential-linear approximation introduced in Section 6.1 is used to attack 18-round LEA. Since the correlation of the differential-linear approximation of $E_2 \circ E_m$ is too low, the LLR-based key recovery technique is not applicable. Hence, the classical key recovery algorithm introduced in Appendix A is adopted. For convenience, the transformation as shown in formula 15 is adopted.

The correlation of the 17-round distinguisher is $-2^{-59.04}$. Let the advantage be $a = 50$ and the success probability be 0.99, the required data complexity is $N = 2^{124.8}$ chosen-plaintext pairs. Consider the output linear mask

$$\gamma_{\text{out}} = [0, 29, 37, 38, 61, 68, 88, 91, 101, 102, 105, 114].$$

If no partition techniques are applied, in order to obtain the above bits, we need to guess $30 \times 3 = 90$ key bits, i.e., the least significant 30 bits of k_0 and k_1 and k_2 . This will make the time complexity $2^{124.8+1+90} = 2^{215.8}$.

⁴ Notice that the required time complexity will be lower when the absolute correlation of $\Delta_m \rightarrow \gamma_{\text{out}}$ exceeds $2^{-10.97}$.

To make the 18-round attack apply to LEA-192, we make an improvement to this attack. Concretely, we compute $z_3[29]$ (i.e., bit 29) using the first two partitions (with correlation 1) as shown in Lemma 1 or Lemma 2. As a result, for the key k_2 , we just need to guess three bits $k_2[29]$ and $k_2[28]$ and $k_2[27]$, instead of 30 bits. In other words, the improved attack guesses $30 \times 2 + 3 = 63$ key bits. Notice that only $\frac{3}{4} \times \frac{3}{4} \times N \approx 2^{-0.83}N$ chosen-plaintext pairs are useful now. Thus, the new data complexity should be $2^{124.8} \times 2^{0.83} = 2^{125.63}$ chosen plaintext pairs. The improved time complexity is $2^{125.63+1+63} = 2^{189.63} < 2^{192}$.

7 Application to Speck

Speck is a family of lightweight block ciphers designed by researchers from the U.S. National Security Agency (NSA) [3]. The Speck family has been a part of the RFID air interface standard (ISO/IEC 29167-22).

The Speck family contains ten members, each of which is characterized by its block size $2n$ and key size mn , thus is named Speck $2n/mn$. All the members with the same block size are also named Speck $2n$, i.e., Speck32, Speck48, Speck64, Speck96 and Speck128 respectively. The round function of Speck is shown in Fig. 11. For Speck32, two parameters are $R = 7$ and $L = 2$. For the remaining members, two parameters are $R = 8$ and $L = 3$.

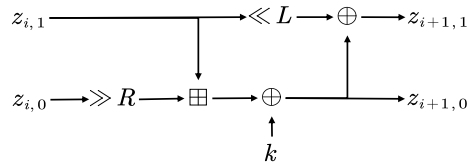


Fig. 11. The Speck round function.

Although the Speck family has been proposed for about ten years, there are few public papers on its resistance against differential-linear attacks. To our knowledge, there are no reports on differential-linear distinguishers of Speck96 and Speck128.

If we set aside the distinguishers given by fully automatic differential-linear approximation searching methods [6, 23], the previous best distinguishers are: (1) 10-round differential-linear distinguisher (with a correlation $2^{-13.90}$) [28] for Speck32; (2) 11-round (with a probability $2^{-44.31}$), 15-round, 17-round, and 20-round differential distinguishers [30] for Speck48, Speck64, Speck96, Speck128 respectively. The results in [6, 23] will be introduced and compared later.

Differential-linear approximations of Speck. Using Algorithm 2, we have found many differential-linear distinguishers for the Speck family.

Again, let r_1, r_m, r_2 denote the number of rounds covered by E_1, E_m, E_2 respectively. For all the members, the absolute correlation threshold used in Algorithm 2 is 2^{-8} . Moreover, we restricted ourselves to the case of a difference Δ_m of the form $[i]$ or $[i, i + 1]$. Table 8 summarizes some differential-linear approximations that we have found. For convenience, we put the three numbers (i.e., r_1, r_m, r_2) on the arrows.

Table 8. Differential-Linear approximations of round-reduced Speck.

Member	$\Delta_{\text{in}} \xrightarrow{r_1} \Delta_m \xrightarrow{r_m} \gamma_m \xrightarrow{r_2} \gamma_{\text{out}}$	Cor of $\Delta_{\text{in}} \rightarrow \gamma_{\text{out}}$
Speck32	$(0x2800, 0x10) \xrightarrow{1} [22] \xrightarrow{6}$ $\sum \xrightarrow{3} (0x2300, 0x4380)$	$2^{-2} \times (-2^{-10.2})$
Speck48	$(0x20082, 0x120200) \xrightarrow{2} [39] \xrightarrow{6}$ $[2] \xrightarrow{3} (0xc2801, 0xd0800)^*$	$2^{-5} \times (-2^{-6.40}) \times 2^{-3 \times 2}$
	$(0x20082, 0x120200) \xrightarrow{2} [39] \xrightarrow{6}$ $[1, 2, 5, 37] \xrightarrow{3} (0xc00c0d, 0xc0c)$	$2^{-5} \times (-2^{-10.51}) \times 2^{-2 \times 2}$
Speck64	$(0x102490, 0x10801004)^* \xrightarrow{3} [53] \xrightarrow{6}$ $[0, 29, 40] \xrightarrow{4} (0x420c0200, 0x2400200)^*$	$2^{-11} \times 2^{-9.15} \times 2^{-4 \times 2}$
	$(0x40004092, 0x10420040)^* \xrightarrow{3} [47] \xrightarrow{6}$ $[25, 26, 29, 37] \xrightarrow{4} (0xc410060, 0x480060)^*$	$2^{-11} \times 2^{-9.63} \times 2^{-5 \times 2}$
Speck96	$(0x20020028202, 0x120200049282) \xrightarrow{4}$ $[93] \xrightarrow{6} [10, 12, 13, 69] \xrightarrow{5}$ $(0x4d000203422b, 0xc0002030223)^*$	$2^{-20} \times 2^{-5.72} \times 2^{-8 \times 2}$
	$(0x20200200282, 0x821202000492) \xrightarrow{4}$ $[85] \xrightarrow{6} [4, 41, 44, 51, 52, 59, 60] \xrightarrow{5}$ $(0x201234d00000, 0x201630c00000)^*$	$2^{-20} \times 2^{-8.59} \times 2^{-9 \times 2}$
Speck128	$(0x40002403c012, 0x10020040000400c2)^* \xrightarrow{5}$ $[117] \xrightarrow{8} [5, 77] \xrightarrow{5}$ $(0xa49000000020343, 0x208000000020303)^*$	$2^{-30} \times 2^{-5.81} \times 2^{-10 \times 2}$
	$(0x2000120120090, 0x8010020000200610)^* \xrightarrow{5}$ $[120] \xrightarrow{8} [7, 79] \xrightarrow{5}$ $(0x82802c000000808, 0x829a40000000809)^*$	$2^{-30} \times 2^{-7.70} \times 2^{-11 \times 2}$

¹ \sum : all the possible γ_m are considered for Speck32. Since many distinguishers returned by Algorithm 2 share the same linear mask γ_{out} , we directly estimate the experimental correlation (i.e., $-2^{-10.2}$) of $\Delta_m \rightarrow \gamma_{\text{out}}$.

² * : There are many optional choices.

³ The total correlation of $\Delta_{\text{in}} \rightarrow \gamma_{\text{out}}$ is $X \times Y \times Z^2$ where X is the probability of $\Delta_{\text{in}} \rightarrow \Delta_m$, Y is the experimental correlation (estimated using 2^{30} pairs and 100 keys) of $\Delta_m \rightarrow \gamma_m$, and Z is the correlation of $\gamma_m \rightarrow \gamma_{\text{out}}$.

We are the first to report differential-linear distinguishers for Speck96 and Speck128. Moreover, when the results in [6, 23] are not considered, the two distinguishers (with a correlation $-2^{-12.2}$ and $-2^{-17.40}$) in the second and third rows of Table 8 are the best distinguishers for Speck32 and Speck48 respectively.

Comparison with fully automatic search methods in [6, 23]. The authors in [6, 23] both searched differential-linear approximations of Speck using their fully automatic MIQCP/MILP-based methods. We have compared our search method, i.e., meet-in-the-middle search (Algorithm 2) with the methods in [6, 23]. Table 9 presents the comparison results.

Table 9. Comparison of three search methods.

Method	Differential-linear approximation				
	Speck32	Speck48	Speck64	Speck96	Speck128
MIQCP/MILP [6]	$A_{10}(-12.0)$	×	×	×	×
MIQCP/MILP [23]	$A_{10}(-11.58)$	$A_{11}(-17.55)$	$A_{12}(-26.93)$	×	×
Algorithm 2	$A_{10}(-12.2)$	$A_{11}(-17.40)$	$A_{13}(-28.15)$	$A_{15}(-41.72)$	$A_{18}(-55.81)$

¹ × : not reported. $A_r(X)$: an r -round differential-linear approximation with an absolute correlation 2^X .

It is clear that our method performs better when applied to Speck48, Speck64, Speck96, and Speck128. In [6], the authors explained that their MIQCP/MILP-based method is currently slow, which is the bottleneck of the method in [23] too. Besides, our method selects good differential-linear approximation $\Delta_m \xrightarrow{E_m} \gamma_m$ by the experimental correlation while the methods in [6, 23] do this by the theoretical correlation. Since the theoretical correlation is less accurate than the experimental correlation (see the data in [6, 23]), we guess that the MIQCP/MILP-based methods may neglect some good differential-linear approximations, e.g., those we have found in this paper.

Now, consider the key recovery attack. The previous best key recovery attack against round-reduced Speck is the differential attack proposed by Dinur in [15]. Based on an h -round differential with a probability p , the adversary can attack $(h + 2)$ -round Speck with a time complexity p^{-1} using the method in [15]. We do not find differential-linear distinguishers which cover more rounds than the previous best differential distinguisher in this paper. As a result, based on the differential-linear approximations presented in Tables 8, we do not obtain key recovery attacks which attack more rounds than previous best attacks.

8 Conclusions

In this paper, based on a novel idea, we have proposed new algorithms to search for differential-linear approximations with a high correlation. Besides, a generic

tool named partition tree is proposed to efficiently generate partitions for complex ARX encryption functions. Based on our work, the cryptanalyst is able to better evaluate a cipher’s resistance to differential-linear attacks. Moreover, the powerful attack framework proposed in [5] can be applied to more ciphers. The applications to LEA and Speck have demonstrated the potential and positive influence of our work.

In our search algorithms, we can not traverse all the possible differences Δ_m , which is still a bottleneck. We believe that it is possible to combine the advantages of our method and MIQCP/MILP-based methods, for accelerating MIQCP/MILP-based methods or making our method totally unconstrained (i.e., have no constraints on the difference Δ_m too). This topic could be a future research direction. Another direction is further explaining the heuristic conclusion presented in Section 2 under no assumptions, i.e., considering various possible dependencies between different differential-linear approximations.

Acknowledgments. We thank the anonymous reviewers for their detailed and helpful comments. This work was supported by the National Natural Science Foundation of China (Nos. 62072270), the National Key R&D Program of China (2018YFA0704701, 2020YFA0309705), Shandong Key Research and Development Program (2020ZLYS09), the Major Scientific and Technological Innovation Project of Shandong, China (2019JZZY010133), the Major Program of Guangdong Basic and Applied Research (2019B030302008), the High Performance Computing Center of Tsinghua University. Y. Chen was also supported by the Shuimu Tsinghua Scholar Program.

A Classical Key Recovery Attack

Consider an $(h + 1)$ -round encryption function E . Using an h -round differential-linear distinguisher $\Delta \rightarrow \gamma$ of E , one can recover the round key $rk \in \mathbb{F}_2^n$ at round $h + 1$. Without loss of generality, assume that $\mathbf{Cor} > 0$ where \mathbf{Cor} is the correlation of the distinguisher $\Delta \rightarrow \gamma$.

Algorithm 4 summarizes the attack procedure. If N is sufficiently large, c_{rk} will be the largest one among 2^n correlations, where rk is the round key. There is a trade-off between N and the rank of c_{rk} . In [12], Blondeau et al. presented a theoretical analysis of the relation between N and the rank of c_{rk} .

The theoretical analysis in [12] is based on a concept named *advantage* introduced in [29]. If an attack on an n -bit key (eg. Algorithm 4) gets the correct value ranked among the top s out of 2^n possible candidates, we say the attack obtained an $(n - \log_2(s))$ -bit *advantage* over the exhaustive search. Then the case where c_{rk} is the largest one among 2^n correlations corresponds to obtaining an n -bit advantage over an n -bit key.

If an attack obtains a preset advantage, we say the attack succeeded. In [12], Blondeau et al. presented that the success probability of the attack is:

$$P_S = \Phi \left(2\sqrt{N}\varepsilon - \Phi^{-1} (1 - 2^{-a}) \right),$$

Algorithm 4 Key recovery

Require: $(h + 1)$ -round encryption function E , sample size N ;

h -round differential-linear distinguisher $\Delta \rightarrow \gamma$;

Ensure: List of 2^n correlations $\{c_0, \dots, c_{2^n-1}\}$.

- 1: $c_i \leftarrow 0$ for $i \in \{0, \dots, 2^n - 1\}$;
 - 2: **for** $j \in \{1, \dots, N\}$ **do**
 - 3: Randomly choose a plaintext x ;
 - 4: $(C_{h+1,0}^j, C_{h+1,1}^j) \leftarrow (E(x), E(x \oplus \Delta))$;
 - 5: **for** $i \in \{0, \dots, 2^n - 1\}$ **do**
 - 6: $(C_{h,0}^j, C_{h,1}^j) \leftarrow (\text{ORD}(C_{h+1,0}^j, i), \text{ORD}(C_{h+1,1}^j, i))$;
 /* $\text{ORD}(C, i)$ stands for one round decryption and i is the key guess. */
 - 7: $c_i \leftarrow c_i + (-1)^{\langle \gamma, C_{h,0}^j \rangle \oplus \langle \gamma, C_{h,1}^j \rangle}$;
 - 8: **end for**
 - 9: **end for**
-

where Φ is the cumulative distribution function of the standard normal distribution, a is the preset advantage of the attack in bits, and ε is the bias of the differential-linear distinguisher $\Delta \rightarrow \gamma$. Note that $\varepsilon = \frac{1}{2} \mathbf{Cor}$ where \mathbf{Cor} is the correlation of $\Delta \rightarrow \gamma$. From this estimate, the data complexity of the differential-linear attack is deduced.

Lemma 4. [12] *Given the bias $\varepsilon = \frac{1}{2} \mathbf{Cor}$ of a differential-linear approximation, the data complexity of a key-recovery attack with advantage a and success probability P_S can be given as*

$$N = \frac{(\Phi^{-1}(P_S) + \Phi^{-1}(1 - 2^{-a}))^2}{4\varepsilon^2}.$$

B Partitions for Parallel Modular Additions

References

1. Aumasson, J., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New features of latin dances: Analysis of salsa, chacha, and rumba. In: Nyberg, K. (ed.) FSE 2008, Revised Selected Papers. LNCS, vol. 5086, pp. 470–488. Springer (2008)
2. Bar-On, A., Dunkelman, O., Keller, N., Weizman, A.: DLCT: A new tool for differential-linear cryptanalysis. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Proceedings. LNCS, vol. 11476, pp. 313–342. Springer (2019)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: DAC 2015, Proceedings. pp. 175:1–175:6. ACM (2015)
4. Beierle, C., Broll, M., Canale, F., David, N., Flórez-Gutiérrez, A., Leander, G., Naya-Plasencia, M., Todo, Y.: Improved differential-linear attacks with applications to ARX ciphers. J. Cryptol. **35**(4), 29 (2022)
5. Beierle, C., Leander, G., Todo, Y.: Improved differential-linear attacks with applications to ARX ciphers. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Proceedings. LNCS, vol. 12172, pp. 329–358. Springer (2020)

Table 10. The partitions for three parallel modular additions.

$b_0b_1b_2b_3b_4b_5$	$z_3[i]$		$b_0b_1b_2b_3b_4b_5$	$z_3[i]$	
	γ	Cor		γ	Cor
000000	101110011001111001	$2^{-2.194}$	100000	110101011100111100	$-2^{-1.178}$
000001	110110011001111001	$2^{-1.217}$	100001	101101011100111100	$-2^{-2.182}$
000010	101110011001111001	$2^{-2.647}$	100010	110100111100111100	$-2^{-1.534}$
000011	110110011001111001	$2^{-1.509}$	100011	101100111100111100	$-2^{-2.556}$
000100	110101011001111001	$2^{-1.184}$	100100	101110011100111100	$-2^{-2.165}$
000101	101101011001111001	$2^{-2.174}$	100101	110110011100111100	$-2^{-1.178}$
000110	110100111001111001	$2^{-1.539}$	100110	101110011100111100	$-2^{-2.479}$
000111	101100111001111001	$2^{-2.574}$	100111	110110011100111100	$-2^{-1.626}$
001000	101110011001111001	$-2^{-1.017}$	101000	110101011100111100	-1
001001	110110011001111001	-1	101001	101101011100111100	$-2^{-1.015}$
001010	101110011001111001	$-2^{-1.428}$	101010	110100111100111100	$-2^{-0.417}$
001011	110110011001111001	$-2^{-0.401}$	101011	101100111100111100	$-2^{-1.412}$
001100	110101011001111001	-1	101100	101110011100111100	$-2^{-0.988}$
001101	101101011001111001	$-2^{-0.989}$	101101	110110011100111100	-1
001110	110100111001111001	$-2^{-0.408}$	101110	101110011100111100	$-2^{-1.424}$
001111	101100111001111001	$-2^{-1.478}$	101111	110110011100111100	$-2^{-0.392}$
010000	101110011010111010	$-2^{-1.522}$	110000	110100111100111100	$-2^{-0.546}$
010001	110110011010111010	$-2^{-0.547}$	110001	101100111100111100	$-2^{-1.519}$
010010	101110011010111010	$-2^{-1.203}$	110010	110101011100111100	$-2^{-0.176}$
010011	110110011010111010	$-2^{-0.191}$	110011	101101011100111100	$-2^{-1.179}$
010100	110100111010111010	$-2^{-0.53}$	110100	101110011100111100	$-2^{-1.623}$
010101	101100111010111010	$-2^{-1.558}$	110101	110110011100111100	$-2^{-0.531}$
010110	110101011010111010	$-2^{-0.191}$	110110	101110011100111100	$-2^{-1.158}$
010111	101101011010111010	$-2^{-1.181}$	110111	110110011100111100	$-2^{-0.193}$
011000	101110011010111010	$-2^{-1.388}$	111000	110100111100111100	$-2^{-0.419}$
011001	110110011010111010	$-2^{-0.411}$	111001	101100111100111100	$-2^{-1.452}$
011010	101110011010111010	$-2^{-0.967}$	111010	110101011100111100	-1
011011	110110011010111010	-1	111011	101101011100111100	$-2^{-0.987}$
011100	110100111010111010	$-2^{-0.406}$	111100	101110011100111100	$-2^{-1.433}$
011101	101100111010111010	$-2^{-1.405}$	111101	110110011100111100	$-2^{-0.415}$
011110	110101011010111010	-1	111110	101110011100111100	$-2^{-1.004}$
011111	101101011010111010	$-2^{-1.012}$	111111	110110011100111100	-1

6. Bellini, E., G erault, D., Grados, J., Makarim, R.H., Peyrin, T.: Fully automated differential-linear attacks against ARX ciphers. In: Rosulek, M. (ed.) CT-RSA 2023, Proceedings. LNCS, vol. 13871, pp. 252–276. Springer (2023)
7. Bernstein, D.J.: The salsa20 family of stream ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 84–97. Springer (2008)
8. Biham, E., Carmeli, Y.: An improvement of linear cryptanalysis with addition operations with applications to FEAL-8X. In: Joux, A., Youssef, A.M. (eds.) SAC 2014, Revised Selected Papers. LNCS, vol. 8781, pp. 59–76. Springer (2014)

Table 11. The partitions for two parallel modular additions.

$b_0b_1b_2b_3b_4$	$z_2[i]$		$b_0b_1b_2b_3b_4$	$z_2[i]$	
	γ	Cor		γ	Cor
00000	11001100111001	$2^{-1.009}$	10000	10101110011100	$-2^{-1.408}$
00001	11001100111001	$2^{-0.964}$	10001	10011110011100	$-2^{-2.546}$
00010	10101100111001	$2^{-1.389}$	10010	11001110011100	$-2^{-0.994}$
00011	10011100111001	$2^{-2.379}$	10011	11001110011100	$-2^{-0.997}$
00100	11001100111001	-1	10100	10101110011100	-1
00101	11001100111001	-1	10101	10011110011100	$-2^{-0.993}$
00110	10101100111001	-1	10110	11001110011100	-1
00111	10011100111001	$-2^{-0.984}$	10111	11001110011100	-1
01000	11001101011010	-1	11000	10011110011100	$-2^{-1.416}$
01001	11001101011010	-1	11001	10101110011100	$-2^{-0.419}$
01010	10011101011010	$-2^{-1.416}$	11010	11001110011100	-1
01011	10101101011010	$-2^{-0.406}$	11011	11001110011100	-1
01100	11001101011010	-1	11100	10011110011100	$-2^{-0.996}$
01101	11001101011010	-1	11101	10101110011100	-1
01110	10011101011010	$-2^{-1.009}$	11110	11001110011100	-1
01111	10101101011010	-1	11111	11001110011100	-1

9. Biham, E., Chen, R.: Near-collisions of SHA-0. In: Franklin, M.K. (ed.) CRYPTO 2004, Proceedings. LNCS, vol. 3152, pp. 290–305. Springer (2004)
10. Biham, E., Dunkelman, O., Keller, N.: Enhancing differential-linear cryptanalysis. In: Zheng, Y. (ed.) ASIACRYPT 2002, Proceedings. LNCS, vol. 2501, pp. 254–266. Springer (2002)
11. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990, Proceedings. LNCS, vol. 537, pp. 2–21. Springer (1990)
12. Blondeau, C., Leander, G., Nyberg, K.: Differential-linear cryptanalysis revisited. *J. Cryptol.* **30**(3), 859–888 (2017)
13. Carlet, C. (ed.): *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press (2020)
14. Chen, Y., Bao, Z., Shen, Y., Yu, H.: A deep learning aided key recovery framework for large-state block ciphers. *Cryptology ePrint Archive*, Paper 2022/1659 (2022)
15. Dinur, I.: Improved differential cryptanalysis of round-reduced speck. In: Joux, A., Youssef, A.M. (eds.) SAC 2014, Proceedings. LNCS, vol. 8781, pp. 147–164. Springer (2014)
16. Dunkelman, O., Indestege, S., Keller, N.: A differential-linear attack on 12-round serpent. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008, Proceedings. LNCS, vol. 5365, pp. 308–321. Springer (2008)
17. Hawkes, P.: Differential-linear weak key classes of IDEA. In: Nyberg, K. (ed.) EUROCRYPT 1998, Proceedings. LNCS, vol. 1403, pp. 112–126. Springer (1998)
18. 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: Kim, Y., Lee, H., Perrig, A. (eds.) WISA 2013, Revised Selected Papers. LNCS, vol. 8267, pp. 3–27. Springer (2013)

19. Kim, D., Kwon, D., Song, J.: Efficient computation of boomerang connection probability for arx-based block ciphers with application to SPECK and LEA. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **103-A**(4), 677–685 (2020)
20. Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Desmedt, Y. (ed.) *CRYPTO 1994, Proceedings*. LNCS, vol. 839, pp. 17–25. Springer (1994)
21. Leurent, G.: Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In: Fischlin, M., Coron, J. (eds.) *EUROCRYPT 2016, Proceedings*. LNCS, vol. 9665, pp. 344–371. Springer (2016)
22. Liu, Y., Wang, Q., Rijmen, V.: Automatic search of linear trails in ARX with applications to SPECK and chaskey. In: Manulis, M., Sadeghi, A., Schneider, S.A. (eds.) *ACNS 2016, Proceedings*. LNCS, vol. 9696, pp. 485–499. Springer (2016)
23. Lv, G., Jin, C., Cui, T.: A miqcp-based automatic search algorithm for differential-linear trails of arx ciphers(long paper). *Cryptology ePrint Archive*, Paper 2023/259 (2023), <https://eprint.iacr.org/2023/259>
24. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseht, T. (ed.) *EUROCRYPT 1993, Proceedings*. LNCS, vol. 765, pp. 386–397. Springer (1993)
25. Mouha, N., Mennink, B., Herrewege, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In: Joux, A., Youssef, A.M. (eds.) *SAC 2014, Proceedings*. LNCS, vol. 8781, pp. 306–323. Springer (2014)
26. Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for arx: Application to salsa20. *Cryptology ePrint Archive*, Paper 2013/328 (2013)
27. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C., Yung, M., Lin, D. (eds.) *Inscrypt 2011, Proceedings*. LNCS, vol. 7537, pp. 57–76. Springer (2011)
28. Niu, Z., Sun, S., Liu, Y., Li, C.: Rotational differential-linear distinguishers of ARX ciphers with arbitrary output linear masks. In: Dodis, Y., Shrimpton, T. (eds.) *CRYPTO 2022, Proceedings*
29. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. *J. Cryptol.* **21**(1), 131–147 (2008)
30. Song, L., Huang, Z., Yang, Q.: Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In: Liu, J.K., Steinfeld, R. (eds.) *ACISP 2016, Proceedings*
31. Sun, L., Wang, W., Wang, M.: Accelerating the search of differential and linear characteristics with the SAT method. *IACR Trans. Symmetric Cryptol.* **2021**(1), 269–315 (2021)
32. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) *ASIACRYPT 2014, Proceedings*. LNCS, vol. 8873, pp. 158–178. Springer (2014)