Fully Automated Differential-Linear Attacks against ARX Ciphers

Emanuele Bellini[1], David Gerault[1], Juan Grados[1], Rusydi H. Makarim[1], and Thomas Peyrin[2]

[1] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE
{emanuele.bellini,david.gerault,juan.grados,rusydi.makarim}@tii.ae
[2] Nanyang Technological University, Singapore
thomas.peyrin@ntu.edu.sg

Abstract. In this paper, we present a fully automated tool for differential-linear attacks using Mixed-Integer Linear Programming (MILP) and Mixed-Integer Quadratic Constraint Programming (MIQCP) techniques, which is, to the best of our knowledge, the very first attempt to fully automate such attacks. We use this tool to improve the correlations of the best 9 and 10-round differential-linear distinguishers on `Speck32/64`, and reach 11 rounds for the first time. Furthermore, we improve the latest 14-round key-recovery attack against `Speck32/64`, using differential-linear distinguishers obtained with our MILP/MIQCP tool. The techniques we present are generic and can be applied to other ARX ciphers as well.

Keywords: `Speck32/64`· differential-linear cryptanalysis · MILP · MIQCP

1 Introduction

In differential cryptanalysis, which was originally proposed in [9], the attacker looks for a fixed input difference $\Delta_{in} = P \oplus P'$ between two plaintexts P and P' that propagates with a high probability through the target cipher to a fixed output difference $\Delta_{out} = C \oplus C'$ between the two corresponding ciphertexts C and C'. This so-called differential is denoted $\Delta_{in} \xrightarrow{p} \Delta_{out}$, where p is the probability $\Pr[C \oplus C' = \Delta_{out} | P \oplus P' = \Delta_{in}]$, and can be used for distinguishing an n-bit block cipher from a random permutation when $p \gg 2^{1-n}$. In linear cryptanalysis, which was originally proposed in [29], the attacker studies the bias of the approximation between the parity of some plaintext and ciphertext bits, selected via a plaintext input mask Γ_{in} and a ciphertext output mask Γ_{out}. For a given plaintext/ciphertext pair (P, C) the bias q of this linear approximation $\Gamma_{in} \xrightarrow{q} \Gamma_{out}$ can be computed with $\Pr[P \cdot \Gamma_{in} = C \cdot \Gamma_{out}] = 1/2 + q$, where $x \cdot y = \bigoplus_{i=0}^{n-1} x[i]y[i]$ for $x, y \in \mathbb{F}_2^n$. It can also be used for distinguishing an n-bit block cipher from a random permutation when $|q| \gg 0$.

Many variations of these two cryptanalysis techniques have been explored and even combinations of them. In Differential-Linear (DL) cryptanalysis, originally introduced in [22], an attacker seeks for a difference-mask pair $(\Delta_{in}, \Gamma_{out})$ and

studies the bias of the approximation between the parity of ciphertext difference bits selected via the mask Γ_{out}, where the ciphertexts pairs are generated from plaintexts pairs with input difference Δ_{in}. The bias q' of a DL approximation can be computed as $\Pr[\Gamma_{out} \cdot (C \oplus C') = 0 | P \oplus P' = \Delta_{in}] = 1/2 + q'$. Similarly to linear cryptanalysis, if $|q'| \gg 0$, we can distinguish the targeted cipher from a random permutation.

In this DL scenario, the cipher E is usually decomposed into two sub-ciphers $E = E_2 \circ E_1$, with a differential $\Delta_{in} \xrightarrow{p} \Delta_{out}$ for E_1 and a linear approximation $\Gamma_{in} \xrightarrow{q} \Gamma_{out}$ for E_2. In order to evaluate the bias q', it is usually assumed that E_1 and E_2 are independent. However, as pointed out in [8], this might not hold true in practice and experiments are required to get a more precise estimation. In particular, a common and handy strategy is to divide the cipher into three parts instead of two $E = E_2 \circ E_m \circ E_1$ and evaluate the correlation of the middle layer E_m experimentally [4].

As of today, the search for DL distinguishers with high correlation is mostly done manually. Cryptanalysts spent efforts and resources finding and checking DL correlations experimentally by using GPUs or a large number of CPUs, see for example the DL attacks presented in [15]. In that work, the authors used GPUs to check the complexities of their attacks with 2^{48} samples. Also, a lot of the community's efforts were spent on connecting the three parts of the DL distinguishers. For example, in [39], the authors explain that they exhausted all middle parts with one active bit in the output of the differential part to attack `Speck32/64` using DL cryptanalysis.

In this paper, we explore how to fully automate the search for DL distinguishers against Addition-Rotation-XOR (ARX) ciphers (such as `Speck32/64`) using Mixed-Integer Linear Programming (MILP) and Mixed-Integer Quadratically Constrained Programming (MIQCP) techniques, assuming that the three parts in which the distinguisher can be divided (as described above) are independent.

1.1 Related works

There are many different techniques and automated tools in the literature for finding differential, linear and DL distinguishers on ARX ciphers.

Finding differential or linear trails on ARX ciphers. A tool to find differential characteristic on ARX ciphers was proposed by Biryukov *et al.* in [11]. This paper proposes a threshold search algorithm with the notion of partial difference distribution table (pDDT): it consists in only collecting the differences from a DDT whose probabilities are greater than a certain threshold. In [12], Biryukov *et al.* adapted Matsui's algorithm and proposed another automatic search algorithm to find optimal differential and linear trails on ARX ciphers.

In [21], Kai Fu *et al.* presented both differential and linear trails obtained by modeling ARX ciphers with MILP techniques and they applied their tool to the `Speck` family of ciphers. In parallel, Song *et al.* [34] used the Mouha *et al.*'s framework [31] for finding differential trails on ARX ciphers by using SMT

solvers too. Using that technique and a counting procedure, they were able to find paths for `Speck` with better probabilities than those presented in [21].

In [1], the authors also used MILP to search for differential trails, with differential distinguishers against `ChaCha` as applications. In [20], Dwivedi *et al.*, presented a technique inspired by the nested Monte-Carlo search algorithm to find differential trails on ARX ciphers, in particular the `LEA` cipher.

In [27] Liu *et al.*, presented a new technique to search for both differential and linear trails on ARX ciphers: the idea is to split the modular additions into small modular additions, where each of these small modular additions outputs a carry bit. Each small component can then be treated as an S-Box. Splitting the modular additions helps to find all the possible differential and linear trails of larger modular additions. This allowed them to find new optimal differential trails for `Speck` and `HIGHT` ciphers.

In [10], Biryukov *et al.* presented a new differential attack technique, called meet-in-the-filter, to attack different versions of `Speck`. In a normal differential attack, generally, the attacker tries to find a distinguisher with a high probability in as many rounds as possible. However, the meet-in-the-filter technique involves using shorter differential characteristics, which results in a more complex analysis phase of the bottom rounds. A precomputation step stores the most likely output differences after additional rounds of the shortened differential characteristics, and the output difference of the observed ciphertext pairs is propagated a few rounds backwards, to check whether it forms a match with some of the precomputed intermediate differences. Using this technique, they mount the best key-recovery attacks in the literature for `Speck`.

Differential-linear distinguishers on ARX ciphers. The best distinguishers and key recovery attacks against ChaCha and Salsa stream ciphers are DL attacks. In [14], Choudhuri *et al.* present differential-linear distinguishers against `ChaCha` and `Salsa`. In that work, they used the Piling-Up Lemma to find DL distinguishers with high correlations and could mount a 6-round key-recovery attack. In [23], Leurent improves the data complexity of the DL attack against Chaskey by improving and using the partitioning technique presented in [7]. This technique helps to find new linear approximations for the modular addition under certain conditions on the data used to mount the attack. These conditions allow the creation of partitions such that some linear approximations occur with probability one. Thus, it is possible to improve the data and time complexities of the attacks against ARX ciphers that use these linear approximations. In [17], Dey *et al.* improved these complexities by using a new Probabilistic Neutral Bits (PNB) technique (originally introduced in [3] to reduce the number of guessed key bits during a key-recovery attack). In [6], Beierle *et al.* improve the complexities of these attacks against `ChaCha` by introducing new techniques in the differential and linear part construction of the DL distinguishers. In [15] Coutinho *et al.* present a 7-round DL distinguisher against `ChaCha`, by using new linear approximations with high correlation in the linear part (found using the Piling-Up Lemma). In [18], Dey *et al.* show a theoretical interpretation of

previous DL distinguishers against `ChaCha` and `Salsa`: they develop a probabilistic framework focusing on the non-linear component of the ARX cipher, the modular addition. In [26], the authors propose to replace the differential part of the DL technique using rotational XOR differentials. A limitation of that work is that those DL distinguishers are restricted to 1-bit output masks. This limitation was eventually overcome in [33], where the authors construct a framework that allows output masks with multiple active bits. They applied that framework also (beside `ChaCha`) to `Alzette`, `SipHash`, and `Speck`. Although [18] and [33], show theoretical interpretations for the DL distinguishers against `ChaCha` and `Salsa`, they do not provide a tool to search for DL distinguishers automatically.

Best attacks against Speck32/64 As we mentioned before, the best key-recovery attacks presented in the literature against `Speck32/64` are those proposed in [10]. Their authors showed attacks for reduced versions of `Speck32/64` to 11, 12, 13, 14, and 15 rounds. In Table 1, we present a comparison between the complexities they found and the complexities we found using our tool. Furthermore, in that table, we compare the complexities found by our tool and attacks published before the paper [10].

To the best of our knowledge, the best distinguishers against `Speck32/64` are those presented in [33]. The authors showed distinguishers for 9 and 10 rounds. In Table 2, we present a comparison between the complexities they found and the complexities we found using our tool. As in Table 1, in Table 2, we also show a comparison between the complexities found by our tool and attacks published before the paper [33].

1.2 Our contribution

First, in order to look for DL distinguishers with high correlations, we designed a new MILP/MIQCP model for ARX ciphers. To the best of our knowledge, this is the first attempt to fully automate the search for DL distinguishers, helping to avoid wasting time and resources (a drawback of previous works) and potentially exploring a larger search space. To accomplish this, we modeled the differential and linear parts by using MILP techniques against ARX ciphers, specifically the ones presented in [21]. Inspired by the framework given by Coutinho *et al.* [16], we have constructed a new framework to model the difference propagation between input and output differences of a cipher. Specifically, under certain independence assumptions, our framework models the correlation existing between a certain input difference and each bit of its output difference. To construct this framework we take advantage of known formulas modeling the difference propagation for ARX components, as for example those presented for modular addition in [18]. After that, we connect the DL distinguishers parts using MILP constraints. Finally, we designed a technique to model the objective function taking into account the probability of the differential part, the middle part's correlation, and the linear part's correlation.

Secondly, we used the earlier mentioned tool as an application to explore DL distinguisher attacks against `Speck32/64`. Compared to previous DL distinguisher attacks, our attacks have better correlations and complexities. Also, to the best of our knowledge, it is the first time a DL distinguisher reaches 11 rounds for `Speck32/64`.

Thirdly, we describe key-recovery attacks based on DL distinguishers and compare them to those based on DL or linear or differential distinguishers. We found that our DL attacks perform better than other DL attacks for `Speck32/64` reduced to 13 and 14 rounds. Specifically, for 13 rounds, we improve by a factor 2^9 the time complexity of the best key-recovery attack based on DL distinguishers. Similar behavior occurs for 14 rounds: we improve by a factor 2^7 the time complexity of the best key-recovery attack based on DL distinguishers. Also, we found that our key-recovery attack against `Speck32/64` reduced to 14 rounds has a better complexity than the best-known key-recovery attack presented in the literature, which is an attack based on differential distinguishers. Our results and a comparison with the state-of-the-art are given in Table 1 and in Table 2 .

Fourth, studying the previous DL attacks against `Speck32/64`, we noticed a mistake in the complexities of the key-recovery attacks presented in [39]. Specifically, we noticed an issue in how the authors computed their data complexities: they forgot to multiply it by a factor representing the number of plaintexts necessary to get the set of rights pairs satisfying the top part of the DL distinguishers. This oversight also affects the time complexities as they depend on the data complexity. This issue is further confirmed by comparing with previous DL attacks against `ChaCha` [6]: we notice that the steps of the technique presented in [39] are the same as in [6], but not the complexity formulas. After correcting the complexities, we remark that the attack for 14 rounds has now a time complexity of 2^{65}, which is larger than a plain brute force attack.

Finally, we used CPUs to verify experimentally the correlations of our new DL distinguishers against `Speck32/64` and the complexities of our key recovery attacks. Our MILP/MIQCP models have been implemented using MiniZinc and solved with Gurobi. All our code is made public for the community, and it is available at https://github.com/Crypto-TII/MILP_MIQCP-differential-linear_key-recovery_speck32.

2 Preliminaries

2.1 Notation

In this article, we will use the following notations. The addition modulo 2^{16} (respectively, the addition in $\mathbb{Z}$) of x and y will be denoted $x \boxplus y$ (respectively, $x+y$). The bitwise eXclusive-OR (XOR) operation of two words x and y of equal size will be denoted $x \oplus y$. The bitwise AND operation of two words x and y of equal size will be denoted $x \odot y$. Also, we will denote as $|x|$ the number of bits of x.

X^m (respectively X^{-m}) will represent the m^{th} $2n$-bit state of `Speck` after m rounds (respectively of the inverse `Speck` after m rounds). When discussing dif-

Rounds	Time Complexity	Data Complexity	Type of Attack	References
13	2^{57}	2^{25}	Differential	[19]
	$2^{61.01}$	2^{24}	Differential-Linear	[39]
	2^{52}	2^{24}	Differential-Linear	This work
	$2^{50.16}$	$2^{31.13}$	Differential	[10]
14	2^{63}	2^{31}	Differential	[19]
	$2^{62.47}$	$2^{30.47}$	Differential	[35]
	2^{65}	2^{28}	Differential-Linear	[39]
	$2^{60.99}$	$2^{31.75}$	Differential	[10]
	2^{58}	2^{31}	Differential-Linear	This work

Table 1: Time and data complexities of our new key recovery attacks against `Speck32/64` reduced to 13 and 14 rounds, with comparison to the state-of-the-art. The complexities of [39] have been corrected in this paper (see Section 4.1).

Rounds	Practical Correlation	Theoretical Correlation	Complexity	References
9	$2^{-11.58}$	-	-	[39]
	$2^{-8.93}$	$2^{-10.23}$	-	[33]
	$2^{-7.3}$	$2^{-11.42}$	$2^{13.4}$	This work
10	$2^{-14.58}$	-	-	[39]
	$2^{-13.90}$	$2^{-15.23}$	-	[33]
	$2^{-12.0}$	$2^{-14.12}$	2^{21}	This work
11	$2^{-16.0}$	$2^{-16.12}$	2^{29}	This work

Table 2: Comparison of the practical and theoretical correlations, as well as the complexity of our new distinguishers, to the state-of-the-art, with a focus on reduced `Speck32/64` to 9, 10, and 11 rounds.. All distinguishers presented in this table are DL distinguishers. Note that the complexity has been derived from the practical correlation.

ferential attacks, the XOR-based difference observed on X^m will be denoted Δ^m and the differential starting from Δ_{in} and ending to Δ_{out} is denoted $\Delta_{in} \to \Delta_{out}$. X_i^m (respectively Δ_i^m) will stand for the i^{th} bit of the state X^m (respectively the state difference Δ^m).

Given a set $\mathcal{S} \in \mathbb{F}_2^n$ and a Boolean function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$, we define

$$\mathbf{Cor}_{x\in\mathcal{S}}\left[f(x)\right] \coloneqq \frac{1}{|\mathcal{S}|}\sum_{x\in\mathcal{S}}(-1)^{f(x)}.$$

2.2 Description of `Speck`

`Speck` and `Simon` are two families of lightweight block ciphers proposed by the National Security Agency (NSA) in 2013 [5]. The members of the `Speck` family are denoted as `Speck` $2n/mn$, where $2n$ is the block size, and mn is the key size. `Speck` is a Feistel cipher. Let (L^{i-1}, R^{i-1}) be the input of the i^{th} round, k_i be

the i^{th} round subkey, the output of the i^{th} round (L^{i-1}, R^{i-1}) is computed as follows:

$$L_i = F(L^{i-1}, R^{i-1}) \oplus k_i, \quad R_i = (R_{i-1} \lll \beta) \oplus L_i,$$

where $F(x, y) = (x \ggg \alpha) \boxplus y$, $\alpha = 7$ and $\beta = 2$ if the block size is 32-bit and $\alpha = 8$ and $\beta = 3$ otherwise. The key schedule part follows a similar process (where the round key is replaced by a constant). We refer the reader to [5] for more details of the construction.

2.3 Continuous analysis of difference propagation

In [16] Coutinho *et al.*, generalize cryptographic operations (such as the ARX operations, linear layers, S-Box, etc.), allowing to express bits as probabilities or correlations. To do this, they created continuous operators from Boolean operators. For example, let us see how they express bits as probabilities for the operator $\odot$ by creating its continuous version. Suppose we want to compute $p_3 = \Pr[a \odot b = 1]$, where a and $b \in \mathbb{F}_2$ are independent random variables. If $\Pr[a = 1] = p_1$ and $\Pr[b = 1] = p_2$, then $p_3 = p_1 p_2$. By using this expression, they defined a continuous operator from $\odot$, and called it "continuous generalization of $\odot$". Specifically, they provide the definitions using the correlation of the random variables instead of probabilities. More precisely, let $\Pr(E)$ be the probability of occurrence of an event E and $b \in \mathbb{F}_2$ be a bit, then we can write $\Pr(b = 1)$ in terms of its correlation ϵ as $\Pr(b = 1) = p = \frac{1}{2}(1 + \epsilon)$.

In some papers, ϵ is also known as deviation, bias, or imbalance. In our example, expressing p_1, p_2 and p_3 as functions of their correlations, we have $p_1 = \frac{1}{2} + \frac{\epsilon_{p_1}}{2}$ and $p_2 = \frac{1}{2} + \frac{\epsilon_{p_2}}{2}$, where the correlations ϵ_{p_1} and ϵ_{p_2} belong to $\mathcal{B} = \{x \in \mathbb{R} \colon -1 \leq x \leq 1\}$. Then, they define the continuous generalization of $\odot$ as $\epsilon_x \odot_{\mathcal{C}} \epsilon_{p_2} = \epsilon_{p_3} = \frac{\epsilon_{p_1}\epsilon_{p_2} + \epsilon_{p_1} + \epsilon_{p_2} - 1}{2}$.

They generalized various cryptographic operations by assuming similar independence properties among the input variables. This enabled them to create continuous versions of entire cryptographic algorithms. Inspired by that framework, we construct continuous functions for the difference propagation of ARX operators. Before proposing them, let us see how to construct this function for the ARX component $\oplus$. Let a and b be two random and independent bits, and let $\Delta a = a \oplus a'$ and $\Delta b = b \oplus b'$. If $\Pr(\Delta a = 1) = \frac{1+\epsilon_p}{2}$ and $\Pr(\Delta b = 1) = \frac{1+\epsilon_q}{2}$, then the probability that $(\Delta a \oplus \Delta b) \oplus (a \oplus b) = 1$ is $\frac{1-\epsilon_p\epsilon_q}{2}$. So, as the example of the previous paragraph, we can express the continuous difference propagation for $\oplus$ in terms of their input correlations ϵ_p, ϵ_q, as $-\epsilon_p\epsilon_q$. In Definition 1, we define more formally continuous difference propagation. From this definition, we created propositions describing continuous difference propagations for every ARX cipher component, and then for entire ARX ciphers. In particular, we applied this continuous difference propagation framework to `Speck32/64`.

Definition 1. *Let $f(x_1, x_2, ..., x_n)$ be a function with input variables belonging to $\mathbb{F}_2^n$, and with output in $\mathbb{F}_2^m$, the continuous difference propagation of f, denoted as $f_{\mathcal{C}\Delta}(\alpha_1, \alpha_2, ..., \alpha_n)$, is a function that maps input variables from $\mathcal{B}^n$ to $\mathcal{B}^m$,*

and describes the correlation between an input difference for f and each bit of its output difference. The exact form of the function $f_{\mathcal{CA}}(\alpha_1, \alpha_2, ..., \alpha_n)$ will depend on the specific properties of the function f.

Coutinho *et al.* present several continuous generalizations for cryptographic operations in [16]. As we mentioned before, in our case, these generalizations are related to the correlation of a certain input difference propagating to a particular bit in the output difference. Because of the linear nature of the XOR operation and rotation operations, the formulas presented by Coutinho *et al.* could model the correlation of a certain input difference propagating to a particular bit in the output difference of these operations (values and differences are behaving identically through these functions). However, the formulas presented for modular addition could not model the propagation of differences through such function since it is non-linear. Instead, we use Theorem 3 and Theorem 4 presented in [18]. These two theorems compute the probability of a certain input difference propagating to a particular bit in the output difference for the modular addition. In Proposition 1, Proposition 2, Proposition 3 and Proposition 4, we present the continuous difference propagation for the XOR, majority function, rotation, and modular addition operations, respectively.

Proposition 1 (Continuous difference propagation of XOR). *Let $x, y \in \mathcal{B}$, then the continuous difference propagation of* XOR *is given by $x \oplus_{\mathcal{CA}} y = -xy$.*

Proof. Already shown in previous paragraphs.

Proposition 2 (Continuous difference propagation of MAJ). *Let x, y and $z \in \mathcal{B}$, then the continuous difference propagation of the* MAJ *function is given by* $\mathrm{MAJ}_{\mathcal{CA}}(x, y, z) = \frac{1}{4}(x + y + z + xyz)$.

Proof. Suppose a, b, c be three independent and randomly chosen bits. Let a', b' and c' such that $\Pr(a \neq a') = p$, $\Pr(b \neq b') = q$ and $\Pr(c \neq c') = r$. Let $A = \Pr(\mathrm{MAJ}(a, b, c) \neq \mathrm{MAJ}(a', b', c'))$, then from Theorem 3 of [18], we have

$$A = r\left(1 - \frac{(1-p) + (1-q) - (1-p)(1-q)}{2}\right) + (1-r)\frac{1 - (1-p)(1-q)}{2}.$$

Replacing the probabilities with their expressions involving their respective correlations $x, y, z \in \mathcal{B}$ we have $\Pr(A) = \frac{1}{2} + \frac{1}{8}(x + y + z + xyz)$.

Proposition 3 (Continuous difference propagation of Left and Right Rotation). *Let $x = (x_0, \cdots, x_{n-1}) \in \mathcal{B}^n$ and $r \in \mathbb{Z}$ such that $0 \leq r \leq n-1$, then the continuous difference propagation of the rotation to the left, and to the right, by r, respectively, is given by*

$$(x_0, \cdots, x_{n-1}) \lll_{\mathcal{CA},r} = (x_r, ..., x_{n-1}, x_0, ..., x_{r-1})$$
$$(x_0, \cdots, x_{n-1}) \ggg_{\mathcal{CA},r} = (x_{n-r}, ..., x_{n-1}, x_0, ..., x_{n-1-r})$$

Proposition 4 (Continuous difference propagation of the Modular Addition). *Let x and y and $z \in \mathcal{B}^n$, then the continuous difference propagation of the addition modulo 2^n function is given by $x \boxplus_{\mathcal{C}\Delta} y = (z_0, \cdots, z_{n-1})$, where z_i is given recursively as follow*

$$\begin{aligned} c_0 &= -1.0, \\ z_i &= x_i \oplus_{\mathcal{C}\Delta} y_i \oplus_{\mathcal{C}\Delta} c_i, \\ c_{i+1} &= \mathrm{MAJ}_{\mathcal{C}\Delta}(x_i, y_i, c_i). \end{aligned} \tag{1}$$

Proof. Follows from Proposition 1, Proposition 2 and Theorem 4 of [18].

2.4 Differential-linear attack

Differential-linear cryptanalysis was introduced by Langford and Hellman in [22] (we will refer to this version as the classical DL attack, see left side of Figure 1). Similarly to the boomerang attack [38], the strategy of this attack consists into dividing a cipher E into two sub ciphers E_1 and E_2, such that $E = E_2 \circ E_1$. Then, one looks for a differential distinguisher and a linear distinguisher for the cipher E_1 and E_2 respectively. In particular, assume that the differential $\Delta_{in} \to \Delta_m$ holds with probability

$$\Pr_{x \in \mathbb{F}_2^n} \left[E_1(x) \oplus E_1(x \oplus \Delta_{in}) = \Delta_m \right] = p.$$

Moreover, let a certain linear trail $\Gamma_m \xrightarrow{E_2} \Gamma_{out}$ to be satisfied with correlation

$$\mathbf{Cor}_{x \in \mathbb{F}_2^n} \left[\langle \Gamma_m, x \rangle \oplus \langle \Gamma_{out}, E_2(x) \rangle \right] = q.$$

By assuming that $E_1(x)$ and $E_2(x)$ are independent random variables, the DL distinguisher exploits the property that

$$\mathbf{Cor}_{x \in \mathbb{F}_2^n} \left[\langle \Gamma_{out}, E(x) \rangle \oplus \langle \Gamma_{out}, E(x \oplus \Delta_{in}) \rangle \right] = pq^2. \tag{2}$$

Thus, by preparing $\epsilon p^{-2} q^{-4}$ pairs of chosen plaintexts $(x, \tilde{x})$ for $\tilde{x} = x \oplus \Delta_{in}$, where $\epsilon \in \mathbb{N}$ is a small constant, one can distinguish the cipher from a Pseudo-Random Permutation (PRP).

The aforementioned assumption sometimes overestimates, or underestimates Equation 2. Therefore, to mitigate this issue, a common strategy (see right-hand side of Figure 1) is to divide the cipher into three parts instead of two $E(x) = E_2 \circ E_m \circ E_1$, effectively adding a middle layer $E_m(x)$. For more details on this strategy, see [4]. This middle part is generally evaluated experimentally. In particular let

$$r = \mathbf{Cor}_{\mathcal{S}} \left[\langle \Gamma_m, E_m(x) \rangle \oplus \langle \Gamma_m, E_m(x \oplus \Delta_m) \rangle \right],$$

where $\mathcal{S}$ denotes the set of samples over which the correlation is computed. Then, the total correlation can be estimated as prq^2. As in the classic DL attack,

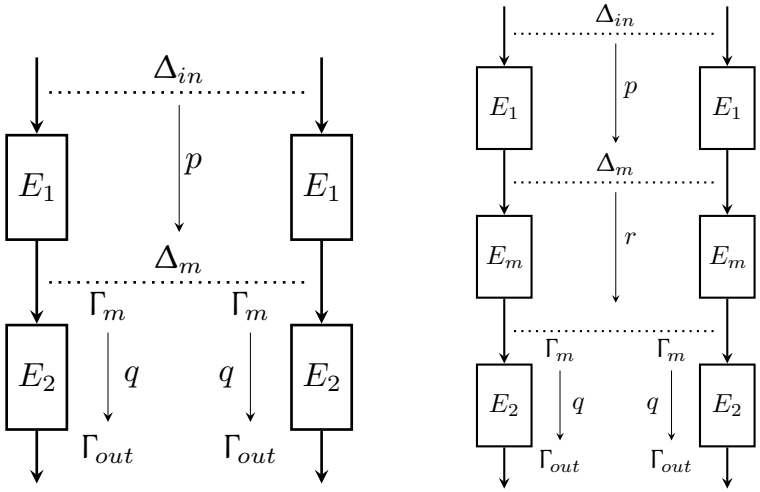

Fig. 1: On the left-hand side, the structure of a classical DL distinguisher. In this distinguisher it is assumed that E_1 and E_2 are independent. In the DL distinguisher of the right side, the middle part helps to take into account the dependency assumption made between E_1 and E_2 in the classical DL distinguisher.

by preparing $\epsilon p^{-2} r^{-2} q^{-4}$ pairs of chosen plaintexts $(x, \tilde{x})$ for $\tilde{x} = x \oplus \Delta_{in}$, where $\epsilon \in \mathbb{N}$ is a small constant, one can distinguish the cipher from a Pseudo-Random Permutation (PRP). We will also denote this improved DL distinguisher as $\Delta_{in} \to \Gamma_{out}$.

In [6], there is a technique that helps to reduce the DL attack complexities against `ChaCha`. This technique was also applied to improve the DL attack complexities against `Speck32/64` in [39]. To better understand this technique, we need to recall the explanation presented in [6] about independent bits in the differential part. Let us assume a cipher F can be parallelized by using two other sub-ciphers, $F_0 \colon \mathbb{F}_2^m \to \mathbb{F}_2^m$ and $F_1 \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ (i.e. $F = F_0 || F_1$). Also, suppose there is a differential trail $\Delta_{in} \to \Delta_{temp}$ on F_1 that occurs with probability p. That is, $\Pr[F_1(\Delta_{in} \oplus x) = \Delta_{temp}] = p$ where $x \in \mathbb{F}_2^n$. Suppose there exists $x' \in \mathbb{F}_2^n$ such that $F_1(\Delta_{in} \oplus x') = \Delta_{temp}$. Then, due F_0 and F_1 independence, we can get 2^m pairs satisfying that differential trail. In fact, those pairs have the shape $(*, x') \in \mathbb{F}_2^{m+n}$, where $*$ represent any vector belonging to $\mathbb{F}_2^m$. So, the number of independent bits, in this case, is m. Since the probability of finding x' is p then the number of pairs we need to distinguish a cipher $E = E_2 \circ E_m \circ E_1$ from a PRP using a distinguisher D with b independent bits in its differential part (i.e. E_1) is $pr^{-2}q^{-4}$ if $2^b > r^{-2}q^{-4}$. In the literature, the pairs $(x', \Delta_{in} \oplus x')$ satisfying $F_1(\Delta_{in} \oplus x') = \Delta_{temp}$ are known as *right pairs*.

The authors of [6] extended the above technique, permitting Probabilistic Independent Bits (PIBs). Specifically, they relax the independence requirement,

allowing e bits to be independent with a probability of less than a given threshold probability p'. Thus, we will choose a right pair with a probability of pp'. `Speck` does not have any explicitly independent bits as `ChaCha` (in its first round), so in [39] the authors applied this extended technique to mount DL distinguishers and to mount key-recovery attacks. We formalize this extended technique explanation in Algorithm 1.

Algorithm 1: Computing the right pairs

Data: A distinguisher $\Delta_{in} \rightarrow \Delta_{temp} \rightarrow \Gamma_{out}$ of a cipher E (with key size k), where $\Delta_{temp} \rightarrow \Gamma_{out}$ has a correlation of rq^2. l PIBs on $\Delta_{in} \rightarrow \Delta_{temp}$. The threshold p' for the l PIBs.

Result: A set of plaintexts satisfying the distinguisher

for $i \leftarrow 0$ **to** $O(\frac{1}{pp'})$ **do**
- $x \xleftarrow{\$} \mathbb{F}_2^n$;
- $x' = x \oplus \Delta_{in}$;
- $K \xleftarrow{\$} \mathbb{F}_2^k$;
- $Y = \{\}$;
- **for** $j \leftarrow 0$ **to** $O(r^{-2}q^{-4})$ **do**
  - Pick a bit set bs from all combinations of the l PIBs;
  - $y = \text{flip}(bs, x)$;
  - $y' = y \oplus \Delta_{in}$;
  - $Y = Y \cup (y, y')$;
- **if** $\mathbf{Cor}_{(t,t')\in Y} [\langle \Gamma_{out}, E_K(t)\rangle \oplus \langle \Gamma_{out}, E_K(t')\rangle] \approx rq^2$ **then**
  - **return** Y;

2.5 MILP and MIQCP

Let k, ℓ be positive integers and $n = k + \ell$. An instance of Mixed-Integer Linear Program (MILP) is the problem of determining

$$\min_{\substack{\mathbf{x}\in\mathbb{Z}^k\times\mathbb{R}^\ell \\ \mathbf{x}=(x_1,\ldots,x_n)}} \left\{ \sum_{i=1}^{n} c_i x_i \middle| A \cdot \mathbf{x}^T \leq b \right\}$$

where $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and A is an $m \times n$ matrix, i.e. it is a problem of minimizing the linear equation $\sum_{i=1}^{n} c_i x_i$ subject to the linear equality constraints defined by $A \cdot \mathbf{x}^T \leq b$. A generalization of MILP by considering the quadratic constraints is termed Mixed-Integer Quadratic Constraint Program (MIQCP). MIQCP is not only a generalization on the set of inequality constraints but also the objective function, i.e. it is defined regardless of the degree of the objective function.

The use of MILP in the cryptanalysis of symmetric-key primitives was first introduced by Mouha, Wang, Gu, and Preneel in 2011 [32]. Since then, MILP has become a standard automated tool to search for differential and linear trails on symmetric-key primitives [21,37,13,24,28]. So far, the use MILP in the cryptanalysis tend to be dedicated towards a single type of attack such as differential cryptanalysis, linear cryptanalysis, or division property.

3 Finding differential-linear distinguishers with MILP/MIQCP solvers

We use MILP/MIQCP techniques to model the entire DL distinguishers. To model the differential and linear parts, we use the techniques presented in [21]. Since these MILP techniques are well known, we detailed them in Appendix B. Recall that, in the middle part, we are working with correlations, that is with values between -1.0 and 1.0. To model the middle part, our approach consists in modeling the propositions presented in Section 2.3, i.e. the continuous difference propagation framework, using the MILP/MIQCP syntax over the real domain $\mathcal{B}$.

In what follows, we write $a \times b$ to represent the multiplication of a and b in the real domain.

Constraints of MAJ. For every modular addition operation with parameters $a \in \mathcal{B}^n$, $b \in \mathcal{B}^n$, $c \in \mathcal{B}^n$, we have the following $n-1$ recursive constraints.

$$c_j = \frac{1}{4}(a_{j-1} + b_{j-1} + c_{j-1} + a_{j-1} \times b_{j-1} \times c_{j-1}), \tag{3}$$

where $c_0 = -1.0$ and $1 \leq j \leq n-1$.

Constraints of Modular Addition Operation. For every modular addition operation with inputs $a \in \mathcal{B}^n$ and $b \in \mathcal{B}^n$, $c \in \mathcal{B}^n$ and output $d \in \mathcal{B}^n$, we have n constraints.

$$d_j = a_j \times b_j \times c_j, \tag{4}$$

where c is a vector representing the carry variables and it is computed using MAJ constraints. Also $0 \leq j \leq n-1$.

Constraints of XOR *Operation.* For every XOR operation with input $a \in \mathcal{B}^n$ and $b \in \mathcal{B}^n$ and output $c \in \mathcal{B}^n$, we have n constraints.

$$c_j = -a_j \times b_j, \tag{5}$$

for $0 \leq j \leq n-1$.

Constraints for R Rounds. For all rounds, we need $2n(R+1)$ variables belonging to $\mathcal{B}$ to represent the states of `Speck`. We do not use any intermediate variable for the XOR and rotation operations, while for the modular addition operation, we only need $(n-1)R$ to represent the carry variables. Summing up, we have a total of $3nR - R - 2n$ variables.

The count of the number of equations is as follows: nR expressions to model the XOR operations. $nR + (n-1)R$ equalities to model the modular addition operations. Summing up, we have a total of $3nR - R$ constraints to model the continuous difference propagation framework for ARX ciphers.

As the reader might have noticed, the constraints presented in this section have terms with degree greater than two. One can convert terms with degree greater than two into quadratic terms by introducing new constraints and new variables. For example, the constraint $x \times y \times z = 1.0$ over the real domain could be reformulated by introducing a new variable t in the following way: $x \times y = t$ and $t \times z = 1.0$. Actually, this procedure is automatically performed by MiniZinc.

In order to clarify how to use the constraints of continuous difference propagation, let's take a look at an example of how a specific input difference results in a difference propagation probability of $\frac{1+\epsilon_j}{2}$ at position j, for $0 \leq j \leq 15$, after one round of the `Speck32/64` cipher.

Consider the input difference $\mathcal{ID} = 0001000000000000, 0101000000000000$, expressed in binary, for the `Speck32/64` cipher. As previously mentioned, a value of 1 at a specific bit indicates that there is a difference with a probability of 1, resulting in a correlation of 1.0. In contrast, a value of 0 means that there is no difference at that bit with a probability of 1, but in this case, the correlation is -1.0. Therefore, the continuous difference propagation version of these bits can be calculated using these correlation values.

$$
\begin{aligned}
a &= (-1,-1,-1,+1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1) \in \mathbb{B}^n,\\
b &= (-1,+1,-1,+1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1) \in \mathbb{B}^n,
\end{aligned}
$$

where a and b represent the left and right side of the input respectively, after translating bits to correlation values. By rotating a seven positions to the right, and b two positions to the left, we get

$$
\begin{aligned}
a' &= (-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,+1,-1,-1,-1,-1,-1) \in \mathbb{B}^n\\
b' &= (-1,+1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,+1) \in \mathbb{B}^n.
\end{aligned}
$$

Let's suppose that a'_j, b_j, and c_j are random independent variables, where c represents the carry vector. By utilizing the MAJ constraints (Equation 3) for a', b, and c, we have

$$
\begin{aligned}
c_0 &= -1.0,\\
c_1 &= \frac{1}{4}\left(a'_0 + b_0 + c_0 + a'_0 b_0 c_0\right) = \frac{1}{4}\left(-1-1-1+(-1)(-1)(-1)\right) = \frac{-4}{4},
\end{aligned}
$$

and so on. By continuing this calculation for all values of the carry vector, we can obtain the final result of

$$
\begin{aligned}
c = (0.0, -0.5, 0.0, -0.984375, -0.96875, -0.9375, -0.875, -0.75,\\
-0.5, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0)\,.
\end{aligned}
$$

Assuming that a'_j, b_j, and c_j are independent random variables, by applying the modular addition constraints (Equation 4) to a', b, and c, we can calculate the left side values after one round of `Speck32/64`. Specifically, we have $d_0 = -1 \times -1 \times -1$, $d_1 = -1 \times -1 \times -1$, and so on, resulting in the following:

$$d = (0.0, 0.5, 0.0, 0.984375, -0.96875, -0.9375, -0.875, -0.75, -0.5, 0.0, \\ 1.0, -1.0, -1.0, -1.0, -1.0, -1.0)\,.$$

Next, assuming that d and b' are independent random variables, by applying the XOR constraints (Equation 5) to d and b', we can calculate the right side values after one round of `Speck32/64`, which results in the following:

$$(-0.0, 0.5, -0.0, -0.984375, 0.96875, 0.9375, 0.875, 0.75, 0.5, \\ -0.0, -1.0, 1.0, 1.0, 1.0, 1.0, -1.0)\,.$$

Assuming independence as stated in Section 2.3, we can interpret the value d_{12} as the correlation of the input difference $\mathcal{ID}$ propagating to the 14th position of the output difference d, with a correlation of 0.984375 (or a probability of $\frac{(1+0.984375)}{2}$). Additionally, using the Piling-Up Lemma, we can create a DL distinguisher by choosing d_7 and d_{12}. Under the same independence assumptions, we can say that the input difference $\mathcal{ID}$ propagates to $d_7 \oplus d_{12}$ with a correlation of 0.984375×-0.5 after one round of the `Speck32/64` encryption algorithm.

Objective Function of the Differential-Linear Model. Using the framework presented in Section 2.3, we can compute the correlation of every bit on the output for a given input difference. Recall that one can estimate the correlation of a DL distinguisher by applying the Piling-Up lemma. In fact, assuming independence between the output bits and knowing that the output mask is linear, we can estimate the correlation by multiplying the correlation of the active bits in the output mask.

In order to have a "good" distinguisher, we need a DL correlation different from zero and as high as possible in absolute value. In other words, given the correlation r, we need to maximize the function $F(r) = |r|$, where $0 < |r| \leq 1$. To do that, it is more convenient to express r as a power of two. Since the goal is to maximize the correlation, we need to minimize $-\log_2(|r|)$. However, this can be difficult as many optimization solvers, such as Gurobi, do not support logarithmic functions in their objective functions. So, let $|r| = 2^{-\log_2(|r|)}$, a crucial step is to find a linear function g to approximate $-\log_2(|r|)$ such that $g(r) \leq -\log_2(|r|)$ (i.e. a lower bound). Indeed, let us show this with the example presented at the beginning of this section. That is, that one starting in $\mathcal{ID}$ and with a output difference of $d \in \mathbb{F}_2^n$ in the left side. In Figure 2 and Figure 3, we show two approximations for $-\log_2(|r|)$. Specifically, we use $g_1(r) = 1 - |r|$ and $g_2(r) = 2 - 2.2|r|$. The approximation of the DL correlation found by using g_1 on the output mask $d_7 \oplus d_{12}$ was $1 - |0.984375 \times 0.5| = 2^{-0.977}$, while the approximation of the DL correlation found by using g_2 was $2 - 2.2|0.984375 \times 0.5| = 2^{-0.125}$. That is, in this case, g_2 approximates $-\log_2(|0.984375 \times 0.5|)$ better than g_1. So, finding a good approximation for $-\log_2(|r|)$ is an important step in our DL model. In the next paragraph, we study how to approximate the $\log_2$ function using the first-order derivative.

Approximating $f(r) = -\log_2(|r|)$ by using the first order derivative. It is common to approximate non-linear functions using piece-wise linear functions to

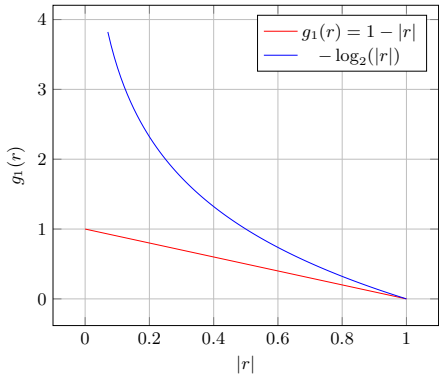

Fig. 2: $g_1(r) = 1 - |r|$.

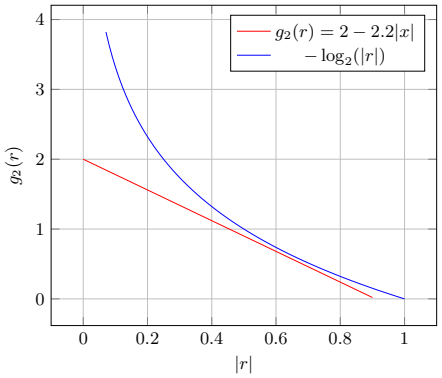

Fig. 3: $g_2(r) = 2 - 2.2|r|$.

have functions suitable for MILP techniques [25]. So, below we explain a simple method to approximate $-\log_2(|r|)$ by using piece-wise linear functions. Specifically, to approximate $-\log_2(|r|)$, we follow the next steps:

- We randomly select M points $((r_0, f(r_0)), \ldots, (r_{M1}, f(r_{M-1})))$ from f. After that, we compute the first-order derivative of f in each of the M points.
- Let g'_i, for $0 \leq i \leq M-1$ be the function corresponding to the result of that first-order derivative. To approximate $-\log_2(|r|)$ using piece-wise linear functions, we find the intersection points between the linear functions g'_i. These intersection points serve as bounds for the piece-wise linear function. Specifically, these intersection points are the common bounds of two consecutive linear functions. So, we have a piece-wise linear function composed of M linear functions. For $0 \leq i \leq (M-1)$, we call those piece-wise linear functions g_i.

Let g be the piece-wise linear function created by using g_i. For measuring the accuracy of the approximation found by this method, we simply compute the difference between the areas under both functions $-\log_2(|r|)$ and g. In Equation 6 we show an example of this approximation by using four random points with an error of 0.54. Also, in Figure 4, we depict this approximation.

$$g(x) = \begin{cases} -19931.57x + 29.9, & 0 \leq x \leq 0.001 \\ -1.87x + 1.82, & 0.001 \leq x \leq 0.77 \\ -1.87x + 1.82, & 0.77 \leq x \leq 0.87 \\ -1.44x + 1.44, & 0.87 \leq x \leq 0.998 \end{cases} \tag{6}$$

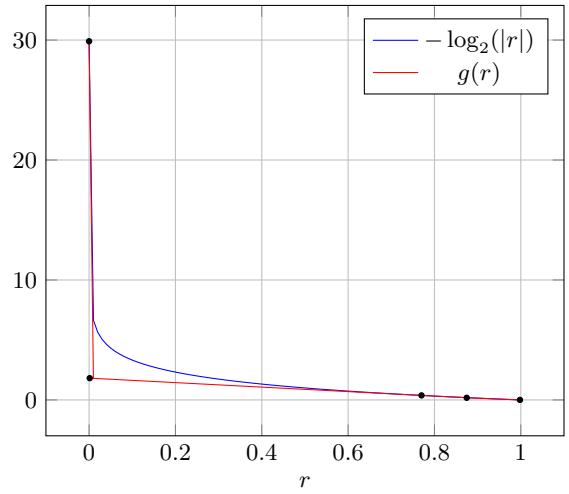

Fig. 4: Approximating $-\log_2(|r|)$ by using piecewise linear functions.

Experimentally, we try several piece-wise linear approximations for $-\log_2(|r|)$ by varying the number of random points. As expected, if we approximate $-\log_2(|r|)$ by using too many random points, our model's time is affected, and if we use only a few points, our model's accuracy is affected. We found that 8 random points give us a balance between time performance and accuracy. In Equation 7, we show the piece-wise linear approximation for $-\log_2(|r|)$ used in the objective function and that gives us a good accuracy among the functions we try with $M = 8$.

$$g(x) = \begin{cases} -19931.570x + 29.897, & 0 \leq x \leq 0.001 \\ -584.962x + 10.135, & 0.001 \leq x \leq 0.004 \\ -192.645x + 8.506, & 0.004 \leq x \leq 0.014 \\ -50.626x + 6.575, & 0.014 \leq x \leq 0.053 \\ -11.87x + 4.483, & 0.053 \leq x \leq 0.142 \\ -8.613x + 4.020, & 0.142 \leq x \leq 0.246 \\ -3.761x + 2.825, & 0.246 \leq x \leq 0.595 \\ -1.444x + 1.444, & 0.595 \leq x \leq 0.998 \end{cases} \tag{7}$$

Modeling the full differential-linear attack against ARX ciphers. To model the three parts, we need to recall that we can have three parts in the DL distinguishers with improved structure, namely the differential part (top part), the DL part (middle part), and the linear part (bottom part). We show how to connect these

three parts in the MILP/MIQCP model setting. Also, since we now have three models, here we explain the new objective function of the model, considering these three parts. As we mentioned before, for the top part, we use the differential MILP model presented in Section B.1, and for the linear part, we use the linear MILP model presented in Section B.2. Both models return a characteristic (either differential or linear) and its probability for a specified number of rounds.

To connect the top part with the middle part, we need to translate the differential output bits into numbers belonging to $\mathcal{B}$. As we see in the example at the beginning of this section, we can translate position values with active differences to 1.0 and positions with non-active differences to -1.0. Also, recalling that the value 1 in a specific position in the output of the differential part means that with certainty, we know there is an active bit, so the probability is 1.0. 0 means that with certainty, we know there is no active bit in that position, so the probability is 0.0. In other words, the correlation in that certain position with output bit 1 is 1.0, and the correlation in that certain position with output bit 0 is -1.0. Considering the output differences of the top part as $t_j \in \mathbb{F}_2$, for $0 \leq j \leq n-1$, where n is the size of the output difference. In similar way, considering the input difference of the middle part as $m_j^{input} \in \mathcal{B}$, for $0 \leq j \leq n-1$. We create the constraints $m_j^{input} = 1.0$ if $t_j = 0$, otherwise $m_j^{input} = -1.0$. To connect the middle part with the linear part, we need to apply the input mask of the linear part to the output of the middle part. Suppose $l_j \in \mathbb{F}_2$, for $0 \leq j \leq n-1$, is the input mask of the linear part. Also, suppose $m_j^{output} \in \mathcal{B}$, for $0 \leq j \leq n-1$ is the output of the middle part, since the correlation of the middle part $r = \prod_{i=0}^{n-1} l_j \times m_j^{output}$ can not be 0, we create the constraint $r > 0.0$. Additionally, we added constraints to approximate $-\log_2(|r|)$ through the function $g(r)$ presented in equation Equation 7.

Once we have the connections among the three parts of the DL distinguisher, we need to minimize the exponents of the three parts. Specifically, suppose x and y are the exponents of the differential and linear part, respectively, then we need to minimize $x + g(r) + 2y$, where $g(r)$ is the approximation of the log function explained in the previous paragraph.

4 Differential-linear attacks against `Speck32/64`

In this section, we review previous DL attacks against `Speck32/64`. Also, we show our new DL distinguishers and key-recovery attacks against `Speck32/64`. For all the key-recovery attacks presented in this section, some rounds are appended to the end of the DL distinguisher, which is below the linear part. Afterward, some round key bits associated with these newer rounds are guessed. The number of these guessed key bits follows the rule presented in section 3.4 of [41].

4.1 Reviewing previous differential-linear attacks against Speck32/64

In [39], the first DL attack against Speck was presented. Specifically against Speck32/64. Here the authors presented two DL distinguishers and used them to mount two key-recovery attacks.

To come up with those distinguishers, they observed that good DL distinguishers in Speck generally have a special structure called "hourglass structure"[30]. In their distinguishers, there is only one active bit in the input of the middle part and a high correlation in the output bits of the middle part. So, they traverse all the middle parts with only one active bit in the input of the middle part and search for high correlations on the output bits of the middle part. In DL Distinguisher 1 and DL Distinguisher 2, we present those distinguishers. .

Differential-Linear Distinguisher 1 ([39]) *The following 9-round DL distinguisher*

$$(\Delta^0_{10}\Delta^0_{17}\Delta^0_{19}) \rightarrow (x^9_{10} \oplus x^9_{11} \oplus x^9_{25} \oplus x^9_{26} \oplus x^9_{27})$$

holds with a correlation of $2^{-11.58}$.

Differential-Linear Distinguisher 2 ([39]) *The following 10-round DL distinguisher*

$$(\Delta^0_1, \Delta^0_8, \Delta^0_{15}, \Delta^0_{22}, \Delta^0_{26}, \Delta^0_{31},) \rightarrow (x^{10}_{10} \oplus x^{10}_{11} \oplus x^{10}_{25} \oplus x^{10}_{26} \oplus x^{10}_{27})$$

holds with a correlation of $2^{-14.58}$.

With these distinguishers, they mount two key-recovery attacks by adding one round before (backward) and three rounds after (forward) the distinguisher. It is possible to prepend one round before the differential part because of the technique presented in [2]. To extend the three rounds behind, they guess b bits by observing the three rounds appended after the DL distinguisher. Thus, using DL Distinguisher 1 they mount a key-recovery attack against 13 rounds of Speck. Using DL Distinguisher 2 they mount a key-recovery attack against 14 rounds of Speck. They got a key-recovery attack on 13 rounds of Speck with data complexity of 2^{22} and time complexity of 2^{59}. Using DL Distinguisher 2 they got a key-recovery attack on 14 rounds of Speck with data complexity of 2^{25} and time complexity of 2^{62}. To see more details of the attack, we refer to Appendix C.

We believe that the complexities claimed in [39] need to be corrected since the authors did not take into account to multiply them by the number of times required to obtain a correct right pair for the first round. Also, one can check this by looking at Algorithm 1 and the complexities obtained in the first paper presenting this technique against ARX ciphers [6]. So, correcting these complexities and using DL Distinguisher 1 they should obtain a key-recovery attack on 13 rounds of Speck32/64 with data complexity of 2^{24} and time complexity of 2^{61}. Also, using the method above on DL Distinguisher 2 they should obtain a key-recovery attack on 14 rounds of Speck32/64 with data complexity of 2^{28}

and time complexity of 2^{65}. Notice that these corrections make the last time complexity worse than brute force for `Speck32/64`.

Another technique to find DL distinguishers against `Speck` appears in [33]. Here the authors build a framework to compute the correlation of a certain DL distinguisher. That framework is based on a technique comprising partitions of $\mathbb{F}_2^n \times \mathbb{F}_2^n$ into subsets where their elements satisfy certain equations. These equations involve the carry bits and the input and output differences of the modular addition operation. For more detail, we refer to Section 2.2 of [33]. To mount the distinguishers, the authors fixed the differential part to the following 4-round differential $(0211, 0a04) \rightarrow (0008, 0008)$. After that they obtained a 8-round DL distinguisher by traversing overall 4-bit masks in the middle part. Finally, they create a 9-round DL distinguisher by extending the linear part by 1 round. To obtain the 10-round DL distinguisher they extended backward the previous 9-round differential-linear distinguisher by 1 round. We refer to Appendix C to see the details of these distinguishers.

4.2 New differential-linear attacks against `Speck32/64`

Using our tool, we observe that the better DL distinguishers do not always have only a single active bit in the output of the differential part. In fact, we found three DL distinguishers for 9, 10 and 11 rounds with 3, 2 and 3 active bits respectively, in the output of the differential part. These distinguishers, presented in DL Distinguisher 3, DL Distinguisher 4, and DL Distinguisher 6, and detailed in Table 4, Table 5 and Table 7 in Appendix A, have theoretical correlations of $2^{-11.42}$, $2^{-14.12}$, $2^{-16.12}$ respectively. In the next paragraph we give more details about the strategies and running timing to obtain these DL distinguishers, and an additional one with a theoretical correlation of $2^{-13.36}$, namely DL Distinguisher 5.

To obtain DL Distinguisher 3 and DL Distinguisher 5, we try several configurations regarding the number of rounds for the top, middle, and bottom parts. The configuration that gives us the best theoretical correlation was 4, 2, and 3 rounds respectively for both distinguishers. To obtain DL Distinguisher 4, we also tried several configurations regarding the number of rounds, in this case the best theoretical correlation was found using 3, 3, and 4 rounds respectively. Also, to obtain DL Distinguisher 4, we needed to add a constraint regarding the number of active bits in the input mask of the linear part. Otherwise, we get a distinguisher with a theoretical correlation of $2^{-15.12}$ (instead of $2^{-14.12}$). To obtain, DL Distinguisher 6, we extended DL Distinguisher 4 one round backward. We also tried to search for a 12-round DL but we did not find a significant theoretical correlation.

The timing results of the proposed tool, under the mentioned conditions, are as follows: The time to find the optimal value for 9 rounds (DL Distinguisher 3) was 70 minutes. The time to find the value $2^{-14.12}$ for the 10 rounds (DL Distinguisher 4) was 2 days. On the first day, we attempted to find the optimal solution but the program did not finish. As a result, a non-optimal solution value of $2^{-15.12}$ was obtained. To improve the results, constraints were added

on the number of active bits in the input mask of the bottom part, which is the most expensive part in the correlation formula for DL distinguishers (see Section 2.4) with an exponent of two. Since "good" DL distinguishers in Speck have a hourglass structure, we constrained the number of active bits of the input linear mask first to one, then to two, and finally to three, resulting in the values $2^{-14.35}$, $2^{-14.35}$ and $2^{-14.12}$ respectively after 2 days. We also tried constraining the number of active bits to 4, but we did not obtain a significant correlation. The time to find DL Distinguisher 5 was 70 minutes, the same as DL Distinguisher 3 since DL Distinguisher 5 is an intermediate value of the experiment we run to obtain DL Distinguisher 3. The time to find DL Distinguisher 6 was 2 days, as to obtain this DL distinguisher we extended one round backwards from DL Distinguisher 4.

For every distinguisher in this section, we conduct an experimental calculation of their correlations. We show them in Table 2. Also, for each distinguisher, we conduct an experimental calculation of the correlation of the middle part. The results of these calculations are compared to the results produced by our tool in Table 3. As shown, our tool provides an lower bound on the experimental results. For example, the experimental result for DL Distinguisher 3 was 0.82, while our tool produced a result of 0.75. As expected, the difference is due to the reliance of our tool on certain independence conditions, as stated in Proposition 1, Proposition 2, Proposition 3, Proposition 4 and the Piling-Up Lemma.

DL distinguishers	Experimental correlation	Theoretical correlation
Middle part of DL Distinguisher 3	0.82	0.75
Middle part of DL Distinguisher 4	0.47	0.23
Middle part of DL Distinguisher 5	0.84	0.78
Middle part of DL Distinguisher 6	0.47	0.23

Table 3: Comparison between the theoretical and experimental correlations of the middle part for every DL distinguisher.

Differential-Linear Distinguisher 3 *The following 9-round DL distinguisher*

$$\left(\Delta_4^0, \Delta_{22}^0, \Delta_{27}^0, \Delta_{29}^0, \Delta_{31}^0\right) \rightarrow \left(x_2^9 \oplus x_9^9 \oplus x_{16}^9 \oplus x_{18}^9 \oplus x_{25}^9\right)$$

holds with a practical correlation of $2^{-7.3}$.

Differential-Linear Distinguisher 4 *The following 10-round DL distinguisher*

$$\left(\Delta_6^0, \Delta_{13}^0, \Delta_{20}^0, \Delta_{22}^0, \Delta_{29}^0\right) \rightarrow \left(x_2^{10} \oplus x_6^{10} \oplus x_{11}^{10} \oplus x_{12}^{10} \oplus x_{13}^{10} \oplus x_{18}^{10} \oplus x_{20}^{10} \oplus x_{22}^{10} \oplus x_{27}^{10} \oplus x_{28}^{10} \oplus x_{29}^{10}\right)$$

holds with a practical correlation of $2^{-12.0}$.

We use DL Distinguisher 4 to mount a key-recovery attack on `Speck32/64` reduced to 13 rounds. Precisely, it is possible to prepend one round before the differential part using the technique presented in [2]. We can also extend two rounds after the distinguisher, and thus guess one full round key (16 bits) and one partial round key (12 bits), for a total of $b = 28$ bits. The attacks work as follows.

1. Compute the l PIBs for the first round of the differential part. That is
$$\left(\Delta_6^0, \Delta_{13}^0, \Delta_{20}^0, \Delta_{22}^0, \Delta_{29}^0\right) \rightarrow \left(\Delta_8^1, \Delta_{31}^1\right).$$
Experimentally, we checked that the first round of our distinguisher has a probability of $p = 2^{-2}$, and has 28 PIBs with probability $p' = 1$. From those 28 PIBs, $l = 21$ are enough to mount the attack.
2. Use Algorithm 1 to compute the set of plaintexts $\mathcal{P}$ satisfying the DL distinguisher
$$\left(\Delta_8^1, \Delta_{31}^1\right) \rightarrow \left(x_2^{10} \oplus x_6^{10} \oplus x_{11}^{10} \oplus x_{12}^{10} \oplus x_{13}^{10} \oplus x_{18}^{10} \oplus x_{20}^{10} \oplus x_{22}^{10} \oplus x_{27}^{10} \oplus x_{28}^{10} \oplus x_{29}^{10}\right).$$
This distinguisher has a correlation of 2^{-10}, so we have enough PIBs to mount the attack.
3. Request the ciphertext pairs of the set $\mathcal{P}$. For DL Distinguisher 4, we request ciphertext pairs generated after 13 rounds. Let $\mathcal{C}$ be the set of these ciphertext pairs.
4. Initialize 2^b counters to zero. For each element (C_i, C_i') in $\mathcal{C}$, try all the 2^b possible values generated by those b key bits. Partially decrypt (C_i, C_i') (3 rounds backwards) to the intermediate state corresponding to the output mask of our DL distinguisher. Compute the XOR sum of the subset of bits contained in the output mask of DL Distinguisher 4, if the values in both pairs are equal, increase the current counter.
5. Sort the counter by the correlation. The right sub-key is expected to be in the first 2^b values of the list.

We have that DL Distinguisher 4 allows to mount a 13 round key recovery attack with a 10 round distinguisher. In this case we target one full round key and 12 bits of the round key after the distinguisher, for a total of $b = 28$ bits. Precisely, the data complexity of the key-recovery attack explained above is 2^{1+21}, and its time complexity is 2^{22+28}. Multiplying by $1/pp' = 2^2$, we got a key-recovery attack on 13 rounds of `Speck32/64` with data complexity of 2^{22+2} and time complexity of 2^{50+2}.

Using similar strategy, but with a 9 round distinguisher, namely DL Distinguisher 5, we obtain a key-recovery attack targeting 3 round keys (two full round keys and 5 bits of the round key after the distinguisher), as done in [39]. In this case $l = 19$ and $b = 37$ obtaining a data complexity of $2^{20.15}$ and a time complexity of $2^{60.15}$. That is, still better than the key-recovery attack for 13 rounds presented in [39].

Differential-Linear Distinguisher 5 *The following 9-round DL distinguisher*

$$(\Delta^0_{11}, \Delta^0_{18}, \Delta^0_{20}, \Delta^0_{22}, \Delta^0_{29}) \rightarrow (x^9_0 \oplus x^9_9 \oplus x^9_{11} \oplus x^9_{24} \oplus x^9_{27})$$

holds with a practical correlation of $2^{-12.0}$.

Differential-Linear Distinguisher 6 *The following 11-round DL distinguisher*

$$(\Delta^0_2, \Delta^0_{20}, \Delta^0_{25}, \Delta^0_{27}, \Delta^0_{29}) \rightarrow (x^{11}_2 \oplus x^{11}_6 \oplus x^{11}_{11} \oplus x^{11}_{12} \oplus x^{11}_{13} \oplus x^{11}_{18} \oplus x^{11}_{20} \oplus x^{11}_{22} \oplus x^{11}_{27} \oplus x^{11}_{28} \oplus x^{11}_{29})$$

holds with a practical correlation of $2^{-16.0}$.

We use DL Distinguisher 6 to mount a key-recovery attack against `Speck32/64` reduced to 14 rounds. To come with this result we use the same strategy as before, where we prepend one round and append two rounds to DL Distinguisher 6 and we target $b = 28$ key bits. On the other hand, we have the following differences:

- $l = 25$ instead $l = 21$;
- $p = 2^{-4}, p' = 0.499$ instead of $p = 2^{-2}, p' = 1$;
- use DL Distinguisher 6 instead of DL Distinguisher 4

The data complexity of the key-recovery attack explained above is 2^{1+25}, and its time complexity is $2^{(25)+28}$. Multiplying for $1/(pp')$, we got a key-recovery attack on 14 rounds of `Speck` with data complexity of 2^{26+5} and time complexity of $2^{5+25+28} = 2^{58}$.

Notice that, using the PIBs, we can have a better data complexity for DL Distinguisher 3, DL Distinguisher 4, and DL Distinguisher 6 than solely applying the formula $\epsilon p^{-2} r^{-2} q^{-4}$ (see Section 2.4). In fact, by using the PIBs computed above we get a data complexity of $2^{13.4}$ for DL Distinguisher 3, 2^{21} for DL Distinguisher 4 and 2^{29} for DL Distinguisher 6. We summarize these data complexities in Table 2.

5 Conclusions and future work

In this work, we considered DL attacks against ARX ciphers and how to model these ciphers in the real domain. Specifically, we studied how to compute the correlation of the output bits of a DL distinguisher modeled in the real domain. We proposed a new automatic tool to search for DL distinguishers. This automatic tool uses MILP and MIQCP techniques, and, to the best of our knowledge, it is the first attempt to fully automate the search for DL distinguishers. By using this tool, we improve previous DL distinguishers against `Speck32/64` reduced to 9 and 10 rounds. Furthermore, we reach an 11-rounds distinguisher for the first time. Using these distinguishers, we improved previous key-recovery attacks against `Speck32/64` reduced to 14 rounds. We aimed to find DL distinguishers for larger instances of Speck, however, our tool is currently slow and thus this is a subject for future investigation. Since, the framework presented in Section 2.3 is generic, we believe that our tool can be applied to other ARX ciphers or even to non-ARX ciphers, for example, SPN ciphers.

References

1. Aaraj, N., Caullery, F., Manzano, M.: MILP-aided Cryptanalysis of Round Reduced ChaCha. IACR Cryptol. ePrint Arch. p. 1163 (2017). URL http://eprint.iacr.org/2017/1163
2. Abed, F., List, E., Lucks, S., Wenzel, J.: Differential Cryptanalysis of Round-Reduced Simon and Speck. In: C. Cid, C. Rechberger (eds.) Fast Software Encryption - FSE 2014, *LNCS*, vol. 8540, pp. 525–545. Springer (2014). https://doi.org/10.1007/978-3-662-46706-0_27. URL https://doi.org/10.1007/978-3-662-46706-0_27
3. Aumasson, J., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In: K. Nyberg (ed.) Fast Software Encryption - FSE 2008, *LNCS*, vol. 5086, pp. 470–488. Springer (2008). https://doi.org/10.1007/978-3-540-71039-4_30. URL https://doi.org/10.1007/978-3-540-71039-4_30
4. Bar-On, A., Dunkelman, O., Keller, N., Weizman, A.: DLCT: A New Tool for Differential-Linear cryptanalysis. In: Y. Ishai, V. Rijmen (eds.) Advances in Cryptology - EUROCRYPT 2019, Part I, *LNCS*, vol. 11476, pp. 313–342. Springer (2019). https://doi.org/10.1007/978-3-030-17653-2_11. URL https://doi.org/10.1007/978-3-030-17653-2_11
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015, pp. 175:1–175:6. ACM (2015). https://doi.org/10.1145/2744769.2747946. URL https://doi.org/10.1145/2744769.2747946
6. Beierle, C., Leander, G., Todo, Y.: Improved Differential-Linear Attacks with Applications to ARX Ciphers. In: D. Micciancio, T. Ristenpart (eds.) Advances in Cryptology - CRYPTO 2020, Part III, *LNCS*, vol. 12172, pp. 329–358. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_12. URL https://doi.org/10.1007/978-3-030-56877-1_12
7. Biham, E., Carmeli, Y.: An Improvement of Linear Cryptanalysis with Addition Operations with Applications to FEAL-8X. In: A. Joux, A.M. Youssef (eds.) Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 8781, pp. 59–76. Springer (2014). https://doi.org/10.1007/978-3-319-13051-4_4. URL https://doi.org/10.1007/978-3-319-13051-4_4
8. Biham, E., Dunkelman, O., Keller, N.: Enhancing Differential-Linear Cryptanalysis. In: Y. Zheng (ed.) Advances in Cryptology - ASIACRYPT 2002, *LNCS*, vol. 2501, pp. 254–266. Springer (2002). https://doi.org/10.1007/3-540-36178-2_16. URL https://doi.org/10.1007/3-540-36178-2_16
9. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: A. Menezes, S.A. Vanstone (eds.) Advances in Cryptology - CRYPTO '90, *LNCS*, vol. 537, pp. 2–21. Springer (1990). https://doi.org/10.1007/3-540-38424-3_1. URL https://doi.org/10.1007/3-540-38424-3_1
10. Biryukov, A., dos Santos, L.C., Teh, J.S., Udovenko, A., Velichkov, V.: Meet-in-the-Filter and Dynamic Counting with Applications to Speck. IACR Cryptol. ePrint Arch. p. 673 (2022). URL https://eprint.iacr.org/2022/673
11. Biryukov, A., Velichkov, V.: Automatic Search for Differential Trails in ARX Ciphers. In: J. Benaloh (ed.) Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, *LNCS*, vol. 8366, pp. 227–250.

Springer (2014). https://doi.org/10.1007/978-3-319-04852-9_12. URL https://doi.org/10.1007/978-3-319-04852-9_12
12. Biryukov, A., Velichkov, V., Corre, Y.L.: Automatic Search for the Best Trails in ARX: Application to Block Cipher Speck. In: T. Peyrin (ed.) Fast Software Encryption - FSE 2016, *LNCS*, vol. 9783, pp. 289–310. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_15. URL https://doi.org/10.1007/978-3-662-52993-5_15
13. Boura, C., Coggia, D.: Efficient MILP modelings for sboxes and linear layers of SPN ciphers. IACR Trans. Symmetric Cryptol. **2020**(3), 327–361 (2020). https://doi.org/10.13154/tosc.v2020.i3.327-361. URL https://doi.org/10.13154/tosc.v2020.i3.327-361
14. Choudhuri, A.R., Maitra, S.: Significantly Improved Multi-bit Differentials for Reduced Round Salsa and ChaCha. IACR Trans. Symmetric Cryptol. **2016**(2), 261–287 (2016). https://doi.org/10.13154/tosc.v2016.i2.261-287. URL https://doi.org/10.13154/tosc.v2016.i2.261-287
15. Coutinho, M., Neto, T.C.S.: Improved Linear Approximations to ARX Ciphers and Attacks Against ChaCha. IACR Cryptol. ePrint Arch. p. 224 (2021). URL https://eprint.iacr.org/2021/224
16. Coutinho, M., de Sousa Júnior, R.T., Borges, F.: Continuous Diffusion Analysis. IEEE Access **8**, 123735–123745 (2020). https://doi.org/10.1109/ACCESS.2020.3005504. URL https://doi.org/10.1109/ACCESS.2020.3005504
17. Dey, S., Sarkar, S.: Improved analysis for reduced round Salsa and Chacha. Discret. Appl. Math. **227**, 58–69 (2017). https://doi.org/10.1016/j.dam.2017.04.034. URL https://doi.org/10.1016/j.dam.2017.04.034
18. Dey, S., Sarkar, S.: A theoretical investigation on the distinguishers of Salsa and ChaCha. Discret. Appl. Math. **302**, 147–162 (2021). https://doi.org/10.1016/j.dam.2021.06.017. URL https://doi.org/10.1016/j.dam.2021.06.017
19. Dinur, I.: Improved Differential Cryptanalysis of Round-Reduced Speck. In: A. Joux, A.M. Youssef (eds.) Selected Areas in Cryptography - SAC 2014, *LNCS*, vol. 8781, pp. 147–164. Springer (2014). https://doi.org/10.1007/978-3-319-13051-4_9. URL https://doi.org/10.1007/978-3-319-13051-4_9
20. Dwivedi, A.D., Srivastava, G.: Differential Cryptanalysis of Round-Reduced LEA. IEEE Access **6**, 79105–79113 (2018). https://doi.org/10.1109/ACCESS.2018.2881130
21. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In: T. Peyrin (ed.) Fast Software Encryption - FSE 2016, *LNCS*, vol. 9783, pp. 268–288. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_14. URL https://doi.org/10.1007/978-3-662-52993-5_14
22. Langford, S.K., Hellman, M.E.: Differential-Linear Cryptanalysis. In: Y. Desmedt (ed.) Advances in Cryptology - CRYPTO '94, *LNCS*, vol. 839, pp. 17–25. Springer (1994). https://doi.org/10.1007/3-540-48658-5_3. URL https://doi.org/10.1007/3-540-48658-5_3
23. Leurent, G.: Improved Differential-Linear Cryptanalysis of 7-Round Chaskey with Partitioning. In: M. Fischlin, J. Coron (eds.) Advances in Cryptology - EUROCRYPT 2016, Part I, *LNCS*, vol. 9665, pp. 344–371. Springer (2016). https://doi.org/10.1007/978-3-662-49890-3_14. URL https://doi.org/10.1007/978-3-662-49890-3_14

24. Li, T., Sun, Y.: Superball: A new approach for MILP modelings of boolean functions. IACR Trans. Symmetric Cryptol. **2022**(3), 341–367 (2022). https://doi.org/10.46586/tosc.v2022.i3.341-367. URL https://doi.org/10.46586/tosc.v2022.i3.341-367
25. Lin, M.H., Carlsson, J., Ge, D., Shi, J., Tsai, J.F.: A Review of Piecewise Linearization Methods. Mathematical Problems in Engineering **2013**, 1–8 (2013). https://doi.org/10.1155/2013/101376
26. Liu, Y., Sun, S., Li, C.: Rotational Cryptanalysis from a Differential-Linear Perspective - Practical Distinguishers for Round-Reduced FRIET, Xoodoo, and Alzette. In: A. Canteaut, F. Standaert (eds.) Advances in Cryptology - EUROCRYPT 2021, Part I, *LNCS*, vol. 12696, pp. 741–770. Springer (2021). https://doi.org/10.1007/978-3-030-77870-5_26. URL https://doi.org/10.1007/978-3-030-77870-5_26
27. Liu, Z., Li, Y., Jiao, L., Wang, M.: A New Method for Searching Optimal Differential and Linear Trails in ARX Ciphers. IEEE Trans. Inf. Theory **67**(2), 1054–1068 (2021). https://doi.org/10.1109/TIT.2020.3040543. URL https://doi.org/10.1109/TIT.2020.3040543
28. Makarim, R.H., Rohit, R.: Towards tight differential bounds of ascon A hybrid usage of SMT and MILP. IACR Trans. Symmetric Cryptol. **2022**(3), 303–340 (2022). https://doi.org/10.46586/tosc.v2022.i3.303-340. URL https://doi.org/10.46586/tosc.v2022.i3.303-340
29. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: T. Helleseth (ed.) Advances in Cryptology - EUROCRYPT '93, *LNCS*, vol. 765, pp. 386–397. Springer (1993). https://doi.org/10.1007/3-540-48285-7_33. URL https://doi.org/10.1007/3-540-48285-7_33
30. Mouha, N., Mennink, B., Herrewege, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In: A. Joux, A.M. Youssef (eds.) Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 8781, pp. 306–323. Springer (2014). https://doi.org/10.1007/978-3-319-13051-4_19. URL https://doi.org/10.1007/978-3-319-13051-4_19
31. Mouha, N., Preneel, B.: Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Paper 2013/328 (2013). URL https://eprint.iacr.org/2013/328. https://eprint.iacr.org/2013/328
32. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In: C. Wu, M. Yung, D. Lin (eds.) Information Security and Cryptology - Inscrypt 2011, *LNCS*, vol. 7537, pp. 57–76. Springer (2011). https://doi.org/10.1007/978-3-642-34704-7_5. URL https://doi.org/10.1007/978-3-642-34704-7_5
33. Niu, Z., Sun, S., Liu, Y., Li, C.: Rotational Differential-Linear Distinguishers of ARX Ciphers with Arbitrary Output Linear Masks. IACR Cryptol. ePrint Arch. p. 765 (2022). URL https://eprint.iacr.org/2022/765
34. Song, L., Huang, Z., Yang, Q.: Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA. In: J.K. Liu, R. Steinfeld (eds.) Information Security and Privacy - ACISP 2016, Part II, *LNCS*, vol. 9723, pp. 379–394. Springer (2016). https://doi.org/10.1007/978-3-319-40367-0_24. URL https://doi.org/10.1007/978-3-319-40367-0_24
35. Song, L., Huang, Z., Yang, Q.: Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA. IACR Cryptol. ePrint Arch. p. 209 (2016). URL http://eprint.iacr.org/2016/209

36. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Towards Finding the Best Characteristics of Some Bit-oriented Block Ciphers and Automatic Enumeration of (Related-key) Differential and Linear Characteristics with Predefined Properties. Cryptology ePrint Archive, Paper 2014/747 (2014). URL https://eprint.iacr.org/2014/747. https://eprint.iacr.org/2014/747
37. 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: P. Sarkar, T. Iwata (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 8873, pp. 158–178. Springer (2014). https://doi.org/10.1007/978-3-662-45611-8_9. URL https://doi.org/10.1007/978-3-662-45611-8_9
38. Wagner, D.A.: The Boomerang Attack. In: L.R. Knudsen (ed.) Fast Software Encryption - FSE '99, *LNCS*, vol. 1636, pp. 156–170. Springer (1999). https://doi.org/10.1007/3-540-48519-8_12. URL https://doi.org/10.1007/3-540-48519-8_12
39. Wang, F., Wang, G.: Improved Differential-Linear Attack with Application to Round-Reduced Speck32/64. In: G. Ateniese, D. Venturi (eds.) Applied Cryptography and Network Security - ACNS 2022, *LNCS*, vol. 13269, pp. 792–808. Springer (2022). https://doi.org/10.1007/978-3-031-09234-3_39. URL https://doi.org/10.1007/978-3-031-09234-3_39
40. Wu, S., Wang, M.: Security Evaluation against Differential Cryptanalysis for Block Cipher Structures. IACR Cryptol. ePrint Arch. p. 551 (2011). URL http://eprint.iacr.org/2011/551
41. Yao, Y., Zhang, B., Wu, W.: Automatic Search for Linear Trails of the SPECK Family. In: J. López, C.J. Mitchell (eds.) Information Security - ISC 2015, *LNCS*, vol. 9290, pp. 158–176. Springer (2015). https://doi.org/10.1007/978-3-319-23318-5_9. URL https://doi.org/10.1007/978-3-319-23318-5_9

A Details of the differential-linear distinguishers

In Table 4, Table 5, Table 6, and Table 7 the first column shows the number of rounds. The second column shows the differential, differential-linear, or linear trails of the distinguishers presented in Section 4. In the middle parts, we have rows and sub-rows. Each row represents a state of the differential-linear trail, and each sub-row represents 4 "real bits" of that state. Sub-rows in the highest positions represent the most significant "real bits", and sub-rows in the lowest positions represent the least significant "real bits".

Differential Part				
0	10101000010000000000000000010000			
1	10000001010000001000000100000000			
2	00000000000000100000010000000000			
3	00000000000000000001000000000000			
4	00010000000000000101000000000000			
Differential-Linear Part				
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$2^{-0.0}$	$-2^{-0.0}$	$2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
5	2^{-inf}	$2^{-1.0}$	2^{-inf}	$2^{-0.0227}$
	$-2^{-0.0457}$	$-2^{-0.0931}$	$-2^{-0.1926}$	$-2^{-0.415}$
	$-2^{-1.0}$	2^{-inf}	$2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	2^{-inf}	$-2^{-1.0}$	2^{-inf}	$2^{-0.0227}$
	$-2^{-0.046}$	$-2^{-0.0931}$	$-2^{-0.1926}$	$-2^{-0.415}$
	$-2^{-1.0}$	2^{-inf}	$2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$2^{-0.0}$
6	2^{-inf}	$2^{-2.9896}$	2^{-inf}	$2^{-0.235}$
	$-2^{-0.4534}$	$-2^{-0.939}$	$-2^{-2.6126}$	2^{-inf}
	$-2^{-8.2002}$	2^{-inf}	$-2^{-0.1889}$	$-2^{-0.3485}$
	$-2^{-0.6627}$	$-2^{-1.3853}$	2^{-inf}	$2^{-1.0}$
	2^{-inf}	$-2^{-3.0128}$	2^{-inf}	$2^{-0.3281}$
	$-2^{-0.6461}$	$-2^{-1.354}$	$-2^{-3.6118}$	2^{-inf}
	$2^{-8.2002}$	2^{-inf}	$-2^{-0.1889}$	$-2^{-0.3485}$
	$-2^{-0.6627}$	$2^{-1.3853}$	2^{-inf}	$2^{-2.0}$
Linear Part				
	00010000000000000000000000100000			
7	00000000000000000000000000100000			
8	00000000100000000000000010000000			
9	00000010000001010000001000000100			

Table 4: 9-round differential-linear distinguisher for `Speck32/64` with theoretical correlation of $2^{-11.42}$ and practical correlation of $2^{-7.3}$.

Differential Part				
0	00100000010100000010000001000000			
1	10000000000000000000000100000000			
2	00000000000000000000010000000000			
3	00000100000000000001010000000000			
Differential-Linear Part				
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$2^{-0.0}$
	$-2^{-0.0}$	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
4	$-2^{-0.415}$	$-2^{-1.0}$	2^{-inf}	$2^{-1.0}$
	2^{-inf}	$2^{-0.0227}$	$-2^{-0.0457}$	$-2^{-0.0931}$
	$-2^{-0.1926}$	$-2^{-0.415}$	$-2^{-1.0}$	2^{-inf}
	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.415}$	$2^{-1.0}$	2^{-inf}	$-2^{-1.0}$
	2^{-inf}	$2^{-0.0228}$	$-2^{-0.0457}$	$-2^{-0.0931}$
	$-2^{-0.1926}$	$-2^{-0.415}$	$-2^{-1.0}$	2^{-inf}
	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
5	$2^{-8.6439}$	$-2^{-7.4297}$	2^{-inf}	$2^{-2.9908}$
	2^{-inf}	$2^{-0.3698}$	$-2^{-0.7904}$	$-2^{-1.7909}$
	$-2^{-3.61}$	2^{-inf}	$-2^{-9.4804}$	2^{-inf}
	$-2^{-0.0906}$	$-2^{-0.1375}$	$-2^{-0.1862}$	$-2^{-0.1926}$
	2^{-inf}	$-2^{-8.4297}$	$2^{-29.5955}$	$-2^{-3.0128}$
	2^{-inf}	$2^{-0.4634}$	$2^{-0.9828}$	$-2^{-2.2056}$
	$-2^{-4.6082}$	2^{-inf}	$2^{-9.4804}$	2^{-inf}
	$-2^{-0.0905}$	$2^{-0.1375}$	$-2^{-0.6012}$	$2^{-1.1926}$
6	2^{-inf}	$-2^{-21.4561}$	2^{-inf}	$-2^{-5.1329}$
	2^{-inf}	$2^{-2.1016}$	$-2^{-5.3324}$	$2^{-17.5129}$
	$2^{-19.229}$	2^{-inf}	$-2^{-18.2}$	2^{-inf}
	$2^{-1.5212}$	$-2^{-2.7524}$	$-2^{-5.1078}$	$2^{-4.8039}$
	2^{-inf}	$-2^{-24.4264}$	2^{-inf}	$2^{-5.5942}$
	2^{-inf}	$2^{-4.3076}$	$-2^{-9.9658}$	2^{-inf}
	2^{-inf}	2^{-inf}	$-2^{-18.2904}$	2^{-inf}
	$2^{-2.1222}$	$2^{-3.9456}$	2^{-inf}	$2^{-13.2877}$
Linear Part				
	00000000000000000000000000001000			
7	00000000001000000000000000100000			
8	00000000100000010100000010000001			
9	00001100000000000000111000000001			
10	00111000010101000011100001000100			

Table 5: 10-round differential-linear distinguisher for `Speck32/64` with theoretical correlation of $2^{-14.12}$ and practical correlation of $2^{-12.0}$.

Differential Part				
0	00100000010101000000100000000000			
1	10100000010000001000000001000000			
2	00000001000000000000000000000010			
3	00000000000000000000000000001000			
4	00000000000010000000000000101000			
Differential-Linear Part				
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$2^{-0.0}$	$-2^{-0.0}$
	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
5	$-2^{-0.415}$	$-2^{-1.0}$	2^{-inf}	$2^{-0.0227}$
	$-2^{-0.0457}$	$-2^{-0.0931}$	$-2^{-0.1926}$	$-2^{-0.415}$
	$-2^{-1.0}$	2^{-inf}	$2^{-1.0}$	2^{-inf}
	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.415}$	$-2^{-1.0}$	2^{-inf}	$2^{-0.0227}$
	$-2^{-0.0457}$	$-2^{-0.0931}$	$-2^{-0.1926}$	$-2^{-0.415}$
	$2^{-1.0}$	$-2^{-36.9324}$	$-2^{-1.0}$	2^{-inf}
	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
6	$-2^{-38.2712}$	$-2^{-8.1178}$	$2^{-29.9588}$	$-2^{-0.2004}$
	$-2^{-0.3746}$	$-2^{-0.7286}$	$-2^{-1.614}$	$2^{-8.7027}$
	$-2^{-7.4548}$	2^{-inf}	$2^{-3.0722}$	$2^{-44.5122}$
	$2^{-0.3661}$	$-2^{-0.5778}$	$-2^{-0.8301}$	$-2^{-1.0}$
	$2^{-38.2712}$	$2^{-8.1584}$	$2^{-30.0046}$	$-2^{-0.2935}$
	$-2^{-0.5673}$	$-2^{-1.1437}$	$2^{-2.6144}$	2^{-inf}
	$-2^{-8.4804}$	2^{-inf}	$-2^{-3.0734}$	$2^{-44.5122}$
	$2^{-0.3661}$	$-2^{-0.5778}$	$-2^{-1.245}$	$-2^{-2.0}$
Linear Part				
	00000000000000000000000000001000			
7	00000000001000000000000000100000			
8	00000000100000010100000010000001			
9	00001100000000000000111000000001			

Table 6: 9-round differential-linear distinguisher for `Speck32/64` with theoretical correlation of $2^{-13.36}$, and practical correlation of $2^{-12.0}$, used to mount a 13-round key recovery attack.

Differential Part				
0	00101010000100000000000000000100			
1	00100000010100000010000001000000			
2	10000000000000000000000100000000			
3	00000000000000000000010000000000			
4	00000100000000000001010000000000			
Differential-Linear Part				
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$2^{-0.0}$
	$-2^{-0.0}$	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
5	$-2^{-0.415}$	$-2^{-1.0}$	2^{-inf}	$2^{-1.0}$
	2^{-inf}	$2^{-0.0227}$	$-2^{-0.0457}$	$-2^{-0.0931}$
	$-2^{-0.1926}$	$2^{-0.415}$	$-2^{-1.0}$	2^{-inf}
	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
	$-2^{-0.415}$	$2^{-1.0}$	2^{-inf}	$-2^{-1.0}$
	2^{-inf}	$2^{-0.0227}$	$-2^{-0.0457}$	$-2^{-0.0931}$
	$-2^{-0.1926}$	$-2^{-0.415}$	$-2^{-1.0}$	2^{-inf}
	$2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$	$-2^{-0.0}$
6	$2^{-8.6439}$	$-2^{-7.4297}$	2^{-inf}	$2^{-2.9896}$
	2^{-inf}	$2^{-0.3702}$	$-2^{-0.7904}$	$-2^{-1.7909}$
	$-2^{-3.61}$	2^{-inf}	$-2^{-9.4804}$	2^{-inf}
	$-2^{-0.0906}$	$-2^{-0.1375}$	$-2^{-0.1862}$	$-2^{-0.1926}$
	2^{-inf}	$-2^{-8.4297}$	$2^{-29.5955}$	$-2^{-3.0128}$
	2^{-inf}	$2^{-0.4634}$	$2^{-0.9828}$	$-2^{-2.2056}$
	$-2^{-4.6082}$	2^{-inf}	$2^{-9.4804}$	2^{-inf}
	$-2^{-0.0905}$	$2^{-0.1375}$	$-2^{-0.6012}$	$2^{-1.1926}$
7	2^{-inf}	$2^{-21.4561}$	2^{-inf}	$-2^{-5.1329}$
	2^{-inf}	$2^{-2.1016}$	$2^{-5.3335}$	$2^{-17.5131}$
	$2^{-19.229}$	2^{-inf}	$2^{-18.2}$	2^{-inf}
	$2^{-1.5212}$	$2^{-2.7524}$	$2^{-5.1078}$	$2^{-4.8039}$
	2^{-inf}	$2^{-24.4264}$	2^{-inf}	$2^{-5.5942}$
	2^{-inf}	$2^{-4.3076}$	$2^{-9.9658}$	2^{-inf}
	2^{-inf}	2^{-inf}	$2^{-18.2904}$	2^{-inf}
	$2^{-2.1222}$	$2^{-3.9456}$	2^{-inf}	$2^{-13.2877}$
Linear Part				
	00000000000000000000000000001000			
8	00000000001000000000000000100000			
9	00000000100000010100000010000001			
10	00001100000000000000111000000001			
11	00111000010101000011100001000100			

Table 7: 11-round differential-linear distinguisher for `Speck32/64` with theoretical correlation of $2^{-16.12}$, and practical correlation of $2^{-16.0}$.

B Finding differential and linear trails with MILP

Automated tools for finding differential and linear characteristics have become increasingly important help for cryptanalysts and designers, as they save a lot of time and reduce the possibility of mistakes when the modeling is not too complex. Different works exist for automating the search for differential or linear characteristics or for finding security bounds for resisting differential and linear cryptanalysis, and among them are those using Mixed Integer Linear Programming (MILP). In [32,40] were proposed methods using MILP to address the problem of counting the minimal number of differentially and linearly active S-boxes. This is an important problem since its solution help designers to establish security bounds against differential and linear cryptanalysis. In [37] Sun *et al.* proposed heuristic methods based on MILP to automatically search for differential characteristics, and in [36] Sun *et al.* improved the technique presented in [37] by making it an exact method. Also, in [36] Sun *et al.* automate the search for linear characteristics using MILP. Since then, many different improvements of MILP models for various ciphers have been proposed, often leading to improved cryptanalysis results.

B.1 MILP-based automatic search for differential trails

We implement a MILP-based automatic search for differential trails on `Speck`, modeling their components using inequalities. In the following paragraphs, we review the inequalities used to model each component of the `Speck` cipher. All those inequalities were presented in [21].

Constraints of XOR *Operation.* For every XOR operation with input differences $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$ and output difference $c \in \mathbb{F}_2^n$, the constraints at bit level for j in $\{0 \cdots n-1\}$ are

$$\begin{aligned} d_{\oplus_j} &\geq a_j \\ d_{\oplus_j} &\geq b_j \\ d_{\oplus_j} &\geq c_j \\ a_j + b_j + c_j &\geq 2d_{\oplus_j} \\ a_j + b_j + c_j &\leq 2 \end{aligned} \tag{8}$$

where $d_{\oplus_j}$ is a dummy variable used to verify there are at least two active terms in $a_j \oplus b_j = c_j$ every time $a_j \neq 0$, $b_j \neq 0$, or $c_j \neq 0$.

Constraints of Modular Addition. The authors of [21] used $13(n-1)+5$ inequalities to model the modular addition operation modulus 2^n with input differences $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$, and output difference $c \in \mathbb{F}_2^n$. For j in $\{0 \cdots n-2\}$ these

inequalities are

$$
\begin{aligned}
a_{j+1} + b_{j+1} - c_{j+1} + a_j + b_j + c_j + d_j &\geq -0 \\
a_{j+1} - b_{j+1} + c_{j+1} + a_j + b_j + c_j + d_j &\geq -0 \\
-a_{j+1} + b_{j+1} + c_{j+1} + a_j + b_j + c_j + d_j &\geq -0 \\
-a_{j+1} - b_{j+1} - c_{j+1} + a_j + b_j + c_j - d_j &\geq -3 \\
a_{j+1} + b_{j+1} + c_{j+1} + a_j + b_j + c_j - d_j &\geq -0 \\
a_{j+1} + b_{j+1} + c_{j+1} - a_j - b_j - c_j + d_j &\geq -6 \\
-a_{j+1} - b_{j+1} - c_{j+1} + a_j - b_j - c_j + d_j &\geq -6 \\
-a_{j+1} - b_{j+1} - c_{j+1} - a_j + b_j - c_j + d_j &\geq -6 \\
-a_{j+1} - b_{j+1} - c_{j+1} - a_j - b_j + c_j + d_j &\geq -6 \\
a_{j+1} + b_{j+1} + c_{j+1} + a_j + b_j - c_j + d_j &\geq -0 \\
a_{j+1} + b_{j+1} + c_{j+1} + a_j - b_j + c_j + d_j &\geq -0 \\
a_{j+1} + b_{j+1} + c_{j+1} - a_j + b_j + c_j + d_j &\geq -0 \\
a_{j+1} + b_{j+1} - c_{j+1} + a_j + b_j + c_j + d_j &\geq -0.
\end{aligned}
\tag{9}
$$

And also the inequalities

$$
\begin{aligned}
d_+ &\geq a_{n-1} \\
d_+ &\geq b_{n-1} \\
d_+ &\geq c_{n-1} \\
a_{n-1} + b_{n-1} + c_{n-1} &\geq 2d_+ \\
a_{n-1} + b_{n-1} + c_{n-1} &\leq 2,
\end{aligned}
\tag{10}
$$

where d_+ is a dummy variable and d_j in Equation 9 represents the probability weight variable [21].

Constraints for R rounds We need $2n(R+1)$ variables to represent the states in R rounds, and nR variables to represent $d_{\oplus_j}$ in R rounds. For the modular addition we need $(n-1)R$ variables to represent the probability weight variables. And R variables to represent d_+. Summing up, we need a total of $4nR + 2n$ variables.

Regarding the number of inequalities, we need $5nR$ inequalities to model the R XOR operations (see Equation 8); we need $13(n-1)R + 5R$ inequalities to model the constraints for the modular addition across R rounds (see Equation 9 and Equation 10). Summing up, we need a total of $18nR - 8R$ inequalities.

Objective Function. Let R be the number of rounds that we are modeling. Also, let d_j^r be the variable representing the probability weight variables at round r and bit j. Then, we need to minimize the following expression

$$
\sum_{r=1}^{R} \sum_{i=0}^{n-2} d_j^r \tag{11}
$$

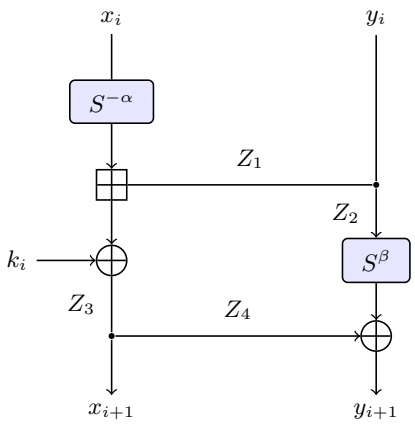

Fig. 5: 1-round of `Speck`.

B.2 Finding linear trails with MILP solvers

This section describes a MILP model to find linear trails for `Speck`. The techniques used to build this model are based on the work from [21].

We modeled the components of the `Speck` cipher using inequalities, and we describe in this section the inequalities used to model the linear behavior of each component.

Constraints of the XOR Operation. For every XOR operation with input masks $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^n$ and output mask $\gamma \in \mathbb{F}_2^n$, the constraints at bit level for j in $\{0, \cdots, n-1\}$ are

$$\alpha_j = \beta_j = \gamma_j.$$

Constraints of the Three-Forked Branch. For every three-forked branch with input masks $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^n$ and output mask $\gamma \in \mathbb{F}_2^n$, we used the following inequalities

$$\begin{aligned} d_{\mathcal{K}_j} &\geq \alpha_j \\ d_{\mathcal{K}_j} &\geq \beta_j \\ d_{\mathcal{K}_j} &\geq \gamma_j \\ \alpha_j + \beta_j + c_j &\geq 2d_{\mathcal{K}_j} \\ \alpha_j + \beta_j + c_j &\leq 2 \end{aligned} \tag{12}$$

where $d_{\mathcal{K}_j}$ is a dummy variable used to verify that there are at least two active terms in $a_j \oplus b_j = c_j$ whenever $a_j \neq 0$, $b_j \neq 0$, or $c_j \neq 0$.

Constraints of the Modular Addition. To model this component, we follow the transition state approach presented in [21]. Therefore, for every modular addition with input masks $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^n$ and output mask $\gamma \in \mathbb{F}_2^n$, the constraints at bit level for i in $\{0, \cdots, n-1\}$ are

$$\begin{aligned} s_i - \gamma_i - \alpha_i + \beta_i + s_{i+1} &\geq 0 \\ s_i + \gamma_i + \alpha_i - \beta_i - s_{i+1} &\geq 0 \\ s_i + \gamma_i - \alpha_i - \beta_i + s_{i+1} &\geq 0 \\ s_i - \gamma_i + \alpha_i - \beta_i + s_{i+1} &\geq 0 \\ s_i + \gamma_i - \alpha_i + \beta_i - s_{i+1} &\geq 0 \\ s_i - \gamma_i + \alpha_i + \beta_i - s_{i+1} &\geq 0 \\ \gamma_i - d_i + \alpha_i + \beta_i + s_{i+1} &\geq 0 \\ s_i + \gamma_i + \alpha_i + \beta_i + s_{i+1} &\leq 4 \\ s_{n-1} &= 0. \end{aligned} \tag{13}$$

where s_i is the probability weight variable [21].

Intermediate Variables. Because of the three-forked branch operation, we need four auxiliary variables. In Figure 5, we illustrate these new auxiliary variables and we can observe that the inequalities for the modular addition operation are the inequalities in Equation 13 where

$$(\alpha, \beta, \gamma) = (x_i \ggg S^{-\alpha}, Z_1, Z_3) \tag{14}$$

and the inequalities for the three-forked branch operations are the inequalities in Equation 12 where

$$(\alpha, \beta, \gamma) \in \{(y_i, Z_1, Z_2), (Z_3, Z_4, x_{i+1})\}.$$

Constraints for R Rounds. We need to use $2n \times (R+1)$ variables to model the input and output differences of each round; $n \times R$ variables for modeling d_{λ} of Equation 12; $(n+1) \times R$ variables for modeling the probability weight variables s_i Equation 13; and $4nR$ variables to model the intermediate variables Z_1, Z_2, Z_3 and Z_4. Summing up, this gives us a total of $8nR + 2n + R$ variables.

The number of inequalities are distributed as follows: $9nR$ inequalities for the modular addition operations; $5 \times n \times R$ inequalities for the three-forked branch operations. Summing up this gives us a total $14nR$ inequalities.

Objective Function. Let R be the number of rounds that we are modeling. Also, let s_j^r be the variable representing the probability weight variable at round r and bit j of the modular addition, then according to [21], we need to minimize the following expression

$$\sum_{r=1}^{R} \sum_{j=0}^{n-1} s_j^r$$

C Previous differential-linear distinguishers

In this appendix we provide details about Differential-Linear distinguishers presented in [39] and in [33].

Using DL Distinguisher 1 and DL Distinguisher 2, Wang *et al* mount two key-recovery attacks in [39]. In Figure 6, we give the details of the distinguishers' top, middle, and bottom parts. The left side corresponds to DL Distinguisher 1, while the right side corresponds to Differential-Linear Distinguisher DL Distinguisher 2. They mount these key-recovery attacks by adding one round before (backward) and three rounds after (forward) the distinguisher. It is possible to prepend one round before the differential part because of the technique presented in [2]. To extend the three rounds in the forward direction, they guess b bits by observing the three rounds appended after the DL distinguisher. Thus, using DL Distinguisher 1 they mount a key-recovery attack against 13 rounds of `Speck`. Using DL Distinguisher 2 they mount a key-recovery attack against 14 rounds of `Speck`. The attacks work as follows.

1. The attack targets the round key at round 10 (only 5 bits), 11 (all 16 bits), and 12 (all 16 bits), for a total of $b = 37$ bits. Compute the l PIBs for the differential part (top part), and use Algorithm 1 to compute the set of plaintexts $\mathcal{P}$ satisfying the differential-linear distinguisher. $l = 21$ for the 13-round attack, and $l = 24$ for the 14-round attack.
2. Request the ciphertext pairs of the set $\mathcal{P}$. For DL Distinguisher 1 they requested ciphertext pairs generated with 13 rounds, and for DL Distinguisher 2 they request ciphertext pairs generated with 14 rounds. Let $\mathcal{C}$ be the set of these ciphertext pairs
3. Append three rounds at the end of DL Distinguisher 1 and DL Distinguisher 2, and check how many key bits are possible to guess by looking the newer three rounds. In this case, they observed they can guess $b = 28$ key bits after DL Distinguisher 1 and DL Distinguisher 2.
4. Initialize 2^b counters to zero. For each element (C_i, C_i') in $\mathcal{C}$, try all the 2^b possible values generated by those b bits. Partially decrypt (C_i, C_i') to the intermediate state corresponding to the output mask of the differential-linear distinguisher. Compute the XOR sum of the subset of bits contained in that output mask, if the values in both pairs are equal, increase the current counter. Sort the counter by the correlation. The right sub-key is expected to be in the first 2^b values of the list.

r	Differential Part
0	000a 0400
1	1000 0000
Differential-Linear Part	
Linear Part	
8	0001 0000
9	0e00 0c00

r	Differential Part
0	8440 8102
1	000a 0400
2	1000 0000
Differential-Linear Part	
Linear Part	
9	0001 0000
10	0e00 0c00

Fig. 6: (a) 9-round DL distinguisher for `Speck32/64` presented in [39]. The experimental correlation of this distinguisher is $2^{-11.58}$. (b) 10-round DL distinguisher for `Speck32/64` presented in [39]. The experimental correlation of this distinguisher is $2^{-14.58}$.

r	Differential Part
0	0211 0a04
4	8100 8102
Differential-Linear Part	
Linear Part	
8	0008 0008
9	5820 4020

r	Differential Part
0	0a20 4205
1	0211 0a04
5	8100 8102
Differential-Linear Part	
Linear Part	
9	0008 0008
10	5820 4020

Fig. 7: (a) 9-round DL distinguisher for `Speck32/64` presented in [33]. The experimental correlation of this distinguisher is $2^{-8.93}$. (b) 10-round DL distinguisher for `Speck32/64` presented in [33]. The experimental correlation of this distinguisher is $2^{-13.90}$.