DLCT: A New Tool for Differential-Linear Cryptanalysis

Achiya Bar-On[1], Orr Dunkelman[2], Nathan Keller[1], and Ariel Weizman[1]

[1] Department of Mathematics, Bar-Ilan University, Israel
[2] Computer Science Department, University of Haifa, Israel

Abstract. Differential cryptanalysis and linear cryptanalysis are the two best-known techniques for cryptanalysis of block ciphers. In 1994, Langford and Hellman introduced the differential-linear (DL) attack based on dividing the attacked cipher E into two subciphers E_0 and E_1 and combining a differential characteristic for E_0 with a linear approximation for E_1 into an attack on the entire cipher E. The DL technique was used to mount the best known attacks against numerous ciphers, including the AES finalist Serpent, ICEPOLE, COCONUT98, Chaskey, CTC2, and 8-round DES.

Several papers aimed at formalizing the DL attack, and formulating assumptions under which its complexity can be estimated accurately. These culminated in a recent work of Blondeau, Leander, and Nyberg (Journal of Cryptology, 2017) which obtained an accurate expression under the sole assumption that the two subciphers E_0 and E_1 are independent.

In this paper we show that in many cases, dependency between the two subcipher s significantly affects the complexity of the DL attack, and in particular, can be exploited by the adversary to make the attack more efficient. We present the Differential-Linear Connectivity Table (DLCT) which allows us to take into account the dependency between the two subciphers, and to choose the differential characteristic in E_0 and the linear approximation in E_1 in a way that takes advantage of this dependency. We then show that the DLCT can be constructed efficiently using the Fast Fourier Transform. Finally, we demonstrate the strength of the DLCT by using it to improve differential-linear attacks on ICEPOLE and on 8-round DES, and to explain published experimental results on Serpent and on the CAESAR finalist Ascon which did not comply with the standard differential-linear framework.

1 Introduction

1.1 Background and previous work

Cryptanalysis of block ciphers. A block cipher is an encryption scheme which accepts an n-bit plaintext and transforms it into an n-bit ciphertext using a k-bit secret key. Block ciphers are the most widely used class of symmetric key primitives nowadays. Most of the modern block ciphers are *iterative*, i.e., consist of a sequence of simple operations called rounds repeated multiple times with small alterations. We denote a plaintext/ciphertext pair of a block cipher by (P, C) and the n-bit state at the beginning of the r'th round of the encryption process by X_r.

While the design of block ciphers is a well-developed field and various block cipher designs (most notably, the AES [36]) are widely accepted to provide strong security, there is no block cipher with a security proof that is fast enough for being used in practice. Instead, our confidence in the security of block ciphers stems from analyzing their resistance with respect to all known cryptanalytic techniques. Thus, development of cryptanalytic techniques is the main means for understanding the practical security of block ciphers.

Differential cryptanalysis and linear cryptanalysis. The two central statistical techniques in cryptanalysis of block ciphers are *differential cryptanalysis*, introduced by Biham and Shamir [8], and *linear cryptanalysis*, introduced by Matsui [31].

Differential cryptanalysis studies the development of differences between two encrypted plaintexts through the encryption process. An r-round *differential* with probability p of a cipher is a property of the form $\Pr[X_{i+r} \oplus X'_{i+r} = \Delta_O | X_i \oplus X'_i = \Delta_I] = p$, denoted in short $\Delta_I \xrightarrow{p} \Delta_O$. Differential attacks exploit *long* (with many rounds) differentials with a non-negligible probability.

Linear cryptanalysis studies the development of parities of subsets of the state bits through the encryption process of a single plaintext. An r-round *linear approximation* with bias q is a property of the form $\Pr[X_{i+r} \cdot \lambda_O = X_i \cdot \lambda_I] = \frac{1}{2} + q$, denoted in short $\lambda_I \xrightarrow{q} \lambda_O$. (Recall that the scalar product of $x, y \in \{0,1\}^n$ is defined as $(\sum_{i=1}^{n} x_i y_i) \bmod 2$.) Linear attacks exploit "long" approximations with a non-negligible bias.

Differential and linear cryptanalysis were used to mount the best known attacks on numerous block ciphers, most notably DES [35]. As a result, resistance to these two cryptanalytic techniques, and in particular, non-existence of high-probability differentials or high-bias linear approximations spanning many rounds of the cipher, has become a central criterion in block cipher design.

Differential-linear cryptanalysis and other combined attacks on block ciphers. While precluding long differentials and linear approximations seems to be sufficient for making the cipher immune to differential and linear attacks, it turned out that in many cases, short characteristics and approximations can also be exploited to break the cipher. The first cryptanalytic technique to demonstrate this was *differential-linear cryptanalysis* (in short: *DL technique*), introduced by Langford and Hellman [27] in 1994. Langford and Hellman showed that if the cipher E can be decomposed as a cascade $E = E_1 \circ E_0$, then a high-probability differential for E_0 and a high-bias linear approximation for E_1 can be combined into an efficient distinguisher for the entire cipher E. The DL technique was used to attack many block ciphers, and in particular, yields the best known attacks on the AES finalist Serpent [20,30], the CAESAR [16] candidate ICEPOLE [22], COCONUT98 [4], Chaskey [28], CTC2 [30], etc.

Differential-linear cryptanalysis was followed by several other combined attacks. In particular, boomerang [38], amplified boomerang [24], and rectangle [3] attacks show that high-probability differentials in E_0 an E_1 can also be combined into an attack on the entire cipher. Other combinations include differential-bilinear, higher-order differential-linear, boomerang-linear attacks, etc. [7]. These combined attacks make non-existence of high-probability short differential and linear approximations a desirable (but harder to fulfill) criterion in block cipher design.

An informal description of the differential-linear attack. The DL attack works as follows. Assume that we have a differential $\Delta_I \xrightarrow{p} \Delta_O$ for E_0 and a linear approximation $\lambda_I \xrightarrow{q} \lambda_O$ for E_1. In order to distinguish E from a random permutation, the adversary considers plaintext pairs with input difference Δ_I and checks, for each pair, whether the corresponding ciphertexts agree on the parity of the mask λ_O.

Denote the plaintexts by P, P', the ciphertexts by C, C', and the intermediate values between E_0 and E_1 by X, X', respectively. The attack combines three approximations: The values $C \cdot \lambda_O$ and $C' \cdot \lambda_O$ are correlated to $X \cdot \lambda_I$ and $X' \cdot \lambda_I$, respectively, by the linear approximation for E_1. The values $X \cdot \lambda_I$ and $X' \cdot \lambda_I$ are correlated, as consequence of the differential for E_0. Hence, $C \cdot \lambda_O$ is correlated to $C' \cdot \lambda_O$. Figure 1 depicts the relations.

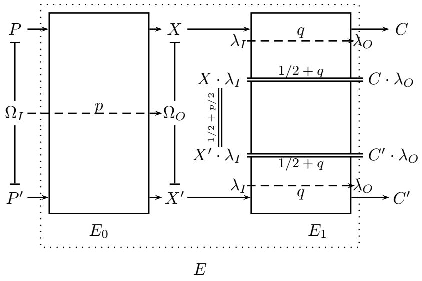


Fig. 1. Differential-Linear Cryptanalysis

Computation shows that under some randomness assumptions to be discussed below, the equality $C \cdot \lambda_O = C' \cdot \lambda_O$ holds with probability $\frac{1}{2} + 2pq^2$. Hence, if p, q are sufficiently large, then the adversary can distinguish E from a random permutation using $O(p^{-2}q^{-4})$ chosen plaintexts. As usual, the distinguisher can be transformed into a key recovery attack by guessing some key material, performing partial encryption/decryption, and applying the distinguisher.

Randomness assumptions behind the DL attack. The attack analysis described above (initially presented in [4]) crucially depends on two assumptions:

1. Among the cases where the differential is not satisfied, $X' \cdot \lambda_I = X \cdot \lambda_I$ holds in half of the cases, as the cipher behaves randomly.
2. There is independence between the two subciphers E_0 and E_1. In particular, the bias of the linear approximations in E_1 is not affected by the fact that they are applied to two intermediate values which correspond to plaintexts with a fixed difference.

As for the first assumption, already in [4] the authors noted that it may fail in many cases, and suggested to check the overall bias of the approximation experimentally whenever possible. Several subsequent papers aimed at formalizing the assumption and at taking into consideration multiple linear approximations for E_1 instead of a single one. The first of those were by Liu et al. [29] and by Lu [30]. Recently, Blondeau et al. [10] presented a formal treatment of the DL attack, based on a general link between differential and linear attacks introduced by Chabaud and Vaudenay [13] and developed by Blondeau and Nyberg [11]. The formal treatment provides an exact expression for the bias of the approximation under the sole assumption that the two parts of the cipher are independent.

Independence between the subciphers in the boomerang attack. While the assumption on independence between E_0 and E_1 was not studied in previous works on the DL attack, it was studied for another combined attack – the boomerang attack. In 2011, Murphy [33] showed that in various cases of interest, the dependency between E_0 and E_1 may significantly affect the complexity and even the possibility of success of the boomerang attack. Murphy's claims were supported by several concrete examples given in other papers. In particular, in [9] and [21], dependency between the subciphers was used to significantly reduce the complexity of the boomerang attacks on SAFER++ and on KASUMI, respectively. On the other hand, it was shown in [25] that the boomerang attack on KASUMI presented in [6] fails (i.e., never succeeds), due to dependency between the subciphers. In [21], Dunkelman et al. proposed the sandwich framework in order to take into account the dependency between the subciphers in the attack analysis.

The Boomerang Connectivity Table (BCT). The inspiration to our work comes from the *boomerang connectivity table* (BCT), proposed by Cid et al. [14] at Eurocrypt'2018 as a new tool for the boomerang attack. The BCT allows computing the complexity of the boomerang attack more accurately, and moreover, enables the adversary to choose the differentials of E_0 and E_1 in a way that exploits the dependency between the subciphers to amplify the overall probability of the boomerang distinguisher. Cid et al. applied the BCT to improve the boomerang attack on the CAESAR finalist `Deoxys` [23] and to explain an unsolved probability amplification for generating a quartet in the tweakable block cipher SKINNY [2].

1.2 Our results

In this paper we study the effect of dependency between the subciphers on differential-linear cryptanalysis.

Inaccuracy of previous analysis due to dependency between the subciphers. We show that in differential-linear attacks on several cryptosystems, due to the effect of dependency, complexity analysis using the standard DL framework led to incorrect estimates, which sometimes were very far from the correct value found experimentally. One concrete example is the attack of Dobraunig et al. [19] on a 5-round variant of the CAESAR finalist Ascon [18]. The authors of [19] state that while by the theory of the DL attack, the overall bias of their approximation is expected to be 2^{-20}, experiments show that the bias is significantly larger: 2^{-2}. The discrepancy is attributed in [19] to multiple linear approximations that affect the overall bias. We show that the huge discrepancy comes mainly from dependency between the two subciphers, and in fact, when we take dependency into account using our new tool presented below, the bias estimate is increased from 2^{-20} all the way to 2^{-5}. (The rest of the difference is indeed explained by the effect of other approximations, as claimed in [19]).

The differential-linear connectivity table. In order to (partially) take the effects of dependency into account, we introduce a new tool: the *differential-linear connectivity table* (DLCT). For a vectorial Boolean function $S : \{0,1\}^n \to \{0,1\}^m$ (e.g., an n-to-m bit S-box in a block cipher), the DLCT of S is an $2^n \times 2^m$ table whose rows correspond to input differences to S and whose columns correspond to bit masks of outputs of S. The value in the cell (Δ, λ), where Δ is a difference and λ is a mask, is

$$DLCT_S(\Delta, \lambda) = |\{x : S(x) \cdot \lambda = S(x \oplus \Delta) \cdot \lambda\}| - 2^{n-1}.$$

We replace the decomposition $E = E_1 \circ E_0$ used in the standard DL attack by the decomposition $E = E_1' \circ E_m \circ E_0'$, where E_0' is covered by the differential, E_m is covered by the DLCT, and E_1' is covered by the remainder of the linear approximation. Usually, E_m covers the first round of E_1 and thus consists of several DLCTs of individual S-boxes applied in parallel. In this case, when computing the overall bias of the DL distinguisher, we replace the biases computed in the first round of the linear approximation by the entries of the DLCT in the corresponding S-boxes. Thus, the DLCT fully addresses the issue of dependency in the switch between E_0 and E_1. Note however that it does not resolve the possible effect of other characteristics and approximations, which still has to be handled using the framework of Blondeau et al. [10] (see Section 2).

Relation of the DLCT to the Fourier transform. We show that each row of the DLCT is equal (up to normalization) to the Fourier transform of the Boolean function represented by the corresponding row of the Differential Distribution Table (DDT) constructed in differential cryptanalysis.

As a result, the DLCT can be computed efficiently using the Fast Fourier Transform. Specifically, each row of the DLCT can be constructed in time $O(2^n + m2^m)$ operations (instead of the trivial 2^{m+n}), and thus, the entire DLCT can be computed in time $O(2^{2n} + m2^{m+n})$ operations. This makes computation of the DLCT feasible even for larger encryption units (e.g., when one wants to compute a single row of the 32-bit Super S-box of AES [36]).

Applications of the DLCT. While the basic use of the DLCT is obtaining a more accurate complexity analysis of the DL attack, it can be used to obtain improved DL attacks as well. Indeed, the adversary can use the DLCT to choose the differential for E_0 and the linear approximation for E_1 in a way that exploits the dependency between the subciphers in her favor. We demonstrate this on two concrete ciphers.

Improved DL attack on ICEPOLE. ICEPOLE [32] is a hardware-oriented authenticated cipher designed by Morawiecki et al. in 2014 and submitted as a candidate to the CAESAR competition. In [22], Huang et al. presented a state-recovery attack in the repeated-nonce settings on 128-bit ICEPOLE with data and time complexity of about 2^{46}, using differential-linear cryptanalysis. This attack is the best known attack on ICEPOLE.

We show that by using better differentials which exploit the dependency between the two underlying subciphers, one can reduce the complexity of the attack to 2^{42}. Furthermore, by exploiting using a better method for choosing the plaintexts, the complexity can be further reduced 2^{36}. We have fully implemented and verified our attack.

Improved DL attack on 8-round DES. One of the first applications of the DL technique is an attack on 8-round DES [35] presented by Biham et al. [4]. The attack is based on a 7-round differential-linear distinguisher with bias $2^{-5.91}$. By analyzing the DLCT of the DES S-boxes, we show that the differential and the linear approximation used in the attack can be replaced which leads to an improved bias of $2^{-5.6}$, thus reducing the complexity of the attack from about 30,000 plaintexts to about 20,000 plaintexts.

As in the case of ICEPOLE, we have fully implemented and verified the attack. While the improvement of our attack over the result of [4] is rather modest, it is another clear example of the applicability of the DLCT and of its ability to exploit dependency in favor of the adversary.

1.3 Organization of the paper

The rest of the paper is organized as follows. In Section 2 we give an overview of the DL attack, and then we present the DLCT and prove that it can be computed efficiently. We use the newly introduced tool to revisit the cryptanalytic results on Ascon [19] in Section 3 and on Serpent [5,20] in Section 4, and explain the discrepancy between the theoretical estimate and the experimental results in these two works. In Sections 5, 6 we present improved DL attacks on ICEPOLE and reduced-round DES, respectively. We conclude the paper with a few open problems for future research in Section 7.

2 The Differential-Linear Connectivity Table

In this section we introduce and discuss the DLCT. We begin with an overview of the DL attack, then we present the DLCT and obtain a new formula for the bias of the DL distinguisher, and finally, we discuss the relation of the DLCT to the Fourier transform and its implications on the DL technique.

2.1 The differential-linear attack

Let E be a cipher that can be decomposed into a cascade $E = E_1 \circ E_0$. Assume that we have a differential $\Delta_I \xrightarrow{p} \Delta_O$ for E_0, i.e., an input difference Δ_I to E_0 leads to an output difference Δ_O from E_0 with probability p, and a linear approximation $\lambda_I \xrightarrow{q} \lambda_O$ for E_1, i.e., for $1/2+q$ of the input/output pairs (I_i, O_i) of E_1 satisfy $\lambda_I \cdot I_i = \lambda_O \cdot O_i$. Denote plaintexts by P, P', ciphertexts by C, C', and intermediate values between E_0 and E_1 by X, X', respectively.

The procedure of the DL attack. As mentioned above, the attack procedure is very simple. In order to distinguish E from a random permutation, the adversary considers plaintext pairs (P, P') such that $P \oplus P' = \Delta_I$ and checks whether the corresponding ciphertext pairs (C, C') satisfy $C \cdot \lambda_O = C' \cdot \lambda_O$. Following Blondeau et al. [10], we denote the overall bias of the DL distinguisher by

$$\mathcal{E}_{\Delta_I,\lambda_O} = \Pr[C \cdot \lambda_O = C' \cdot \lambda_O | P \oplus P' = \Delta_I]. \tag{1}$$

Naive analysis of the attack complexity. The attack uses a combination of three approximations:

1. We have $\Pr[C \cdot \lambda_O = X \cdot \lambda_I] = \frac{1}{2} + q$, by the linear approximation for E_1.
2. We have $\Pr[X' \cdot \lambda_I = X \cdot \lambda_I] = \frac{1}{2} \pm \frac{p}{2}$ (where the sign depends on the parity of $\Delta_O \cdot \lambda_I$). This is because by the differential for E_0, for fraction p of the plaintext pairs we have $X \oplus X' = \Delta_O$, and in particular, $X' \cdot \lambda_I = X \cdot \lambda_I \oplus \Delta_O \cdot \lambda_I$, and we assume that among the rest of the plaintext pairs, $X' \cdot \lambda_I = X \cdot \lambda_I$ holds in half of the cases.
3. We have $\Pr[C' \cdot \lambda_O = X' \cdot \lambda_I] = \frac{1}{2} + q$, by the linear approximation for E_1.

Note that the equality $C \cdot \lambda_O = C' \cdot \lambda_O$ holds if among the three equalities (1),(2), (3), either all three hold or exactly one holds. Using Matsui's Piling-up lemma [31] (similar analysis holds also when using correlation matrices [17]), we have

$$\mathcal{E}_{\Delta_I,\lambda_O} = \Pr[C \cdot \lambda_O = C' \cdot \lambda_O] = \frac{1}{2} + 2pq^2. \tag{2}$$

Hence, if p, q are sufficiently large, then the adversary can distinguish E from a random permutation using $O(p^{-2}q^{-4})$ chosen plaintexts (see [10,37] for the exact relation between the data complexity and the success probability of the distinguisher). As usual, the distinguisher can be transformed into a key recovery attack by guessing some key material, performing partial encryption/decryption, and applying the distinguisher.

The exact complexity analysis of Blondeau et al. [10]. As mentioned above, the naive complexity analysis crucially depends on two randomness assumptions. The first is that the equality $X' \cdot \lambda_I = X \cdot \lambda_I$ holds in approximately half of the cases in which the differential in E_0 fails; the second is that E_0 and E_1 are independent. Blondeau et al. [10] provided an exact expression for $\mathcal{E}_{\Delta_I,\lambda_O}$, relying only on the latter assumption. In order to present their result, we need a few more notations (adapted from [10]).

Consider an encryption function E' and denote its inputs by Z, Z' and its outputs by W, W'. We use the notation $\Delta_I \xrightarrow[E']{} \Delta_O$ for the differential transition $\Delta_I \rightarrow \Delta_O$ through E', and the notation $\lambda_I \xrightarrow[E']{} \lambda_O$ for the linear transition $\lambda_I \rightarrow \lambda_O$ through E'. For an input difference Δ_I and an output mask λ, we denote

$$\epsilon^{E'}_{\Delta_I,\lambda} = \Pr[W \cdot \lambda = W' \cdot \lambda | Z \oplus Z' = \Delta_I] - \frac{1}{2},$$

and for two masks λ_I, λ_O, we denote

$$c^{E'}_{\lambda_I,\lambda_O} = 2\left(\Pr[W \cdot \lambda_O = W' \cdot \lambda_O \middle| Z \cdot \lambda = Z' \cdot \lambda] - \frac{1}{2}\right).$$

Note that $c^{E'}_{\lambda_I,\lambda_O}/2$ is the bias of the linear approximation $\lambda_I \xrightarrow[E']{} \lambda_O$.

By [10, Theorem 2], assuming only independence between E_0 and E_1, we have:

$$\mathcal{E}_{\Delta_I,\lambda_O} = \sum_{\lambda_I} \epsilon^{E_0}_{\Delta_I,\lambda_I} (c^{E_1}_{\lambda_I,\lambda_O})^2. \tag{3}$$

Of course, the expression (3) is usually hard to evaluate, and thus, in practice one mostly has to rely (at least partially) on randomness assumptions and verify the results experimentally.

2.2 The differential-linear connectivity table and its properties

Definition of the DLCT. Let $S : \{0,1\}^n \rightarrow \{0,1\}^m$ be a vectorial Boolean function. The DLCT of S is an $2^n \times 2^m$ table whose rows correspond to input differences to S and whose columns correspond to bit masks of outputs of S. Formally, for $\Delta \in \{0,1\}^n$ and $\lambda \in \{0,1\}^m$, the DLCT entry (Δ, λ) is

$$DLCT_S(\Delta, \lambda) \triangleq \left| \left\{ x \middle| \lambda \cdot S(x) = \lambda \cdot S(x \oplus \Delta) \right\} \right| - 2^{n-1}.$$

Sometimes it will be more convenient for us to use the normalized DLCT entry

$$\overline{DLCT}_S(\Delta, \lambda) \triangleq \frac{DLCT_S(\Delta, \lambda)}{2^n} = \Pr[\lambda \cdot S(x) = \lambda \cdot S(x \oplus \Delta)] - \frac{1}{2}$$

instead of $DLCT_S(\Delta, \lambda)$. The DLCT of Serpent's S-box S0 is given in Table 1.

A natural interpretation of the DLCT is the following. Assume that S is equal to the entire encryption function E. Then $DLCT_S(\Delta, \lambda)$ is equal (up to

Table 1. The DLCT of Serpent's S0

$\Delta \setminus \lambda$	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1_x	8	0	-4	0	-4	-4	0	4	0	-4	0	0	0	4	0	0
2_x	8	0	0	0	-4	0	0	-4	-8	0	0	0	4	0	0	4
3_x	8	-4	0	0	4	-4	0	-4	0	0	-4	0	0	4	0	0
4_x	8	0	0	-8	0	0	0	0	-8	0	0	8	0	0	0	0
5_x	8	4	0	0	0	0	-4	0	0	0	4	0	-4	0	-4	-4
6_x	8	-4	-4	0	0	0	0	0	8	-4	-4	0	0	0	0	0
7_x	8	0	4	0	0	0	-4	0	0	4	0	0	-4	0	-4	-4
8_x	8	-4	0	0	-4	0	-4	4	0	0	-4	0	0	0	4	0
9_x	8	0	0	-8	0	0	0	0	0	0	0	0	0	0	0	0
A_x	8	0	-4	0	4	0	-4	-4	0	-4	0	0	0	0	4	0
B_x	8	0	0	0	-4	0	0	-4	0	0	0	-8	4	0	0	4
C_x	8	0	4	0	0	-4	0	0	0	4	0	0	-4	-4	0	-4
D_x	8	-4	-4	8	0	4	4	0	0	-4	-4	0	0	-4	-4	0
E_x	8	4	0	0	0	-4	0	0	0	0	4	0	-4	-4	0	-4
F_x	8	0	0	0	0	4	4	0	0	0	0	-8	0	-4	-4	0

normalization) to the bias we obtain when we apply to E a DL distinguisher with $\Delta_I = \Delta$ and $\lambda_O = \lambda$ (that is, to the bias $\mathcal{E}_{\Delta,\lambda}$). Thus, if we could construct a DLCT for the entire encryption scheme E, then the DLCT would completely capture the DL attack. As such a construction is mostly infeasible, we construct the DLCT for small components of the cryptosystem (usually, single S-boxes or Super S-boxes) that lie on the boundary between E_0 and E_1, in order to obtain accurate analysis of the transition between the two subciphers.[3]

The DLCT framework. Like in the sandwich [21] and the BCT [14] frameworks of the boomerang attack, when we use the DLCT, we divide the cipher E into three subciphers: $E = E_1' \circ E_m \circ E_0'$, where E_0' is covered by the differential $\Delta_I \rightarrow \Delta$, E_m is covered by the DLCT (or by several DLCTs applied in parallel), and E_1' is covered by the remainder of the linear approximation $\lambda \rightarrow \lambda_O$. Usually, E_m covers the first round of E_1 and thus it consists of several DLCTs of single S-boxes applied in parallel. However, if it is feasible to cover by the DLCT a larger part of the cipher, this is advantageous, as the DLCT gives the exact result for the part of the cipher it covers. For example, we construct such a (partial) DLCT for three rounds in our improved DL attack on 8-round DES presented in Section 6.

Complexity analysis. Assume that we have a differential $\Delta_I \xrightarrow{p} \Delta$ for E_0' and a linear approximation $\lambda \xrightarrow{q'} \lambda_O$ for E_1'. (Note that since E_1' is typically a subcipher

[3] An important independence assumption on the transition is that the active S-boxes (with non-zero input difference and non-zero output) of the transition are independent of each other.

of E_1, it is expected that $|q'| > |q|$). Denote the intermediate values after E_0' by X, X' and the intermediate values after E_m by Y, Y'.

Adapting the naive analysis of the DL attack complexity presented above (i.e., Eq. (2)), we obtain

$$\mathcal{E}_{\mathit{\Delta}_I,\lambda_O} = 4p \cdot \frac{DLCT_{E_m}(\mathit{\Delta},\lambda)}{2^n} \cdot (q')^2 = 4p \cdot \overline{DLCT}_{E_m}(\mathit{\Delta},\lambda) \cdot (q')^2. \quad (4)$$

Note that in the degenerate case where $E_m = Id$, we have $DLCT_{E_m}(\mathit{\Delta},\lambda) = 2^{n-1}$ for all $(\mathit{\Delta},\lambda)$ and $q' = q$, and thus, we obtain $\mathcal{E}_{\mathit{\Delta}_I,\lambda_O} = 2pq^2$ which is equivalent to Eq. (2) above. Interestingly, when $E_1' = Id$, the resulting bias is $\mathcal{E}_{\mathit{\Delta}_I,\lambda_O} = p \cdot \overline{DLCT}_{E_m}(\mathit{\Delta},\lambda_O)$.

In order to adapt the exact analysis of [10] (i.e., Eq. (3)), a bit more computation is needed. Note that for any λ, we have

$$\begin{aligned}\epsilon^{E_m \circ E_0'}_{\mathit{\Delta}_I,\lambda} &= \sum_{\mathit{\Delta}} (\Pr[\mathit{\Delta}_I \xrightarrow[E_0']{} \mathit{\Delta}] \Pr[Y \cdot \lambda = Y' \cdot \lambda | X \oplus X' = \mathit{\Delta}]) - \frac{1}{2} \\ &= \frac{1}{2^n} \sum_{\mathit{\Delta}} \Pr[\mathit{\Delta}_I \xrightarrow[E_0']{} \mathit{\Delta}] \cdot DLCT_{E_m}(\mathit{\Delta},\lambda) = \sum_{\mathit{\Delta}} \Pr[\mathit{\Delta}_I \xrightarrow[E_0']{} \mathit{\Delta}] \cdot \overline{DLCT}_{E_m}(\mathit{\Delta},\lambda).\end{aligned}$$

Plugging this expression into Eq. (3) (where E_1' is used instead of E_1 and $E_m \circ E_0'$ is used instead of E_0), we obtain that the exact bias of the DL distinguisher is

$$\mathcal{E}_{\mathit{\Delta}_I,\lambda_O} = \sum_{\mathit{\Delta},\lambda} \Pr[\mathit{\Delta}_I \xrightarrow[E_0']{} \mathit{\Delta}] \cdot \overline{DLCT}_{E_m}(\mathit{\Delta},\lambda) (c^{E_1'}_{\lambda,\lambda_O})^2. \quad (5)$$

Note that Eq. (5) is still not free of randomness assumptions; e.g., it relies on round independence within E_0' and E_1' (see [10]). However, it is the most accurate expression for the bias of the DL distinguisher obtained so far. On the other hand, Eq. (5) is usually hard to evaluate, and in the actual applications of the DLCT we do rely on some randomness assumptions and verify our results experimentally.

Properties of the DLCT. A trivial property of the DLCT is that for any S, we have $DLCT_S(0,\lambda) = 2^{n-1}$ for all λ. Indeed, if two inputs to S are equal then the corresponding outputs agree on any bit mask. This means that in the DL attack, if for some S-box in the first round of E_1, the difference $\mathit{\Delta}_O$ is zero in the entire S-box, then the linear approximation in that S-box holds for sure, while without the dependency between the intermediate values X, X' it was anticipated to hold only probabilistically. This trivial property corresponds to the *middle round S-box trick* used in [9] to speed up the boomerang attack and covered by the BCT [14]. Surprisingly, this feature was not noted before explicitly in the context of the DL attack. For example, even if we take into account only this trivial type of dependency, the bias of the DL distinguisher of Dobraunig et al. [19] on Ascon increases from 2^{-20} to 2^{-8}. Interestingly, the authors of [19] chose the linear approximation deliberately in such a way that the active S-boxes in its first round correspond to inactive S-boxes in $\mathit{\Delta}$. However, they did not take this dependency into account in the computation of the bias, as it is neglected in the classical DL model.

Another trivial property of the DLCT is that for any S, we have $DLCT_S(\mathit{\Delta}, 0) = 2^{n-1}$ for all $\mathit{\Delta}$'s. Indeed, if the the input mask is zero (i.e., no output bits are approximated), then their actual value (and by proxy, their input difference), is of no importance.

Inspection of the DLCT of Serpent's S-box S0 presented in Table 1 shows that it contains the value $\pm 2^{n-1}$ not only in the trivial entries of the form $DLCT_S(0, \lambda)$ or $DLCT_S(\mathit{\Delta}, 0)$, but also in 9 additional places. Moreover, it contains many very high / very low values that can be used by an adversary, if she can adjust the differential and the linear approximation such that these high/low values are used. The existence of such high/low value entries is not surprising, as current design of S-boxes does not take the DLCT into account. Hence, the DLCT can serve as a new design criterion for S-boxes, partially measuring immunity of the cipher with respect to DL attacks.

2.3 Relation of the DLCT to the Fourier transform

We now show that the DLCT is closely related to the Fourier transform of the DDT and that this relation can be used to efficiently compute the DLCT. We begin with a few preliminaries.

The Fourier-Walsh transform of Boolean functions. Let $f : \{0,1\}^m \to \mathbb{R}$ be a Boolean function. (Note that f does not have to be two-valued.) The Fourier-Walsh transform of f is the function $\hat{f} : \{0,1\}^m \to \mathbb{R}$ defined by

$$\hat{f}(y) = \frac{1}{2^m} \sum_{x \in \{0,1\}^m} f(x) \cdot (-1)^{x \cdot y} = \frac{1}{2^m} \left(\sum_{\{x : x \cdot y = 0\}} f(x) - \sum_{\{x : x \cdot y = 1\}} f(x) \right).$$

The DDT and the LAT. The DLCT resembles in its structure the two central tools used in differential and linear cryptanalysis – the Difference Distribution Table (DDT) and the Linear Approximation Table (LAT). For a vectorial Boolean function $S : \{0,1\}^n \to \{0,1\}^m$, the DDT of S is an $2^n \times 2^m$ table whose rows correspond to input differences to S and whose columns correspond to output differences of S. Formally, for $\mathit{\Delta}_I \in \{0,1\}^n$ and $\mathit{\Delta}_O \in \{0,1\}^m$, we have

$$DDT_S(\mathit{\Delta}_I, \mathit{\Delta}_O) = \left| \left\{ x \middle| S(x) \oplus S(x \oplus \mathit{\Delta}_I) = \mathit{\Delta}_O \right\} \right|.$$

Similarly, the LAT of S is an $2^n \times 2^m$ table whose rows correspond to bit masks of inputs to S and whose columns correspond to bit masks of outputs of S. Formally, for $\lambda_I \in \{0,1\}^n$ and $\lambda_O \in \{0,1\}^m$, we have

$$LAT_S(\lambda_I, \lambda_O) = \left| \left\{ x \middle| \lambda_O \cdot S(x) = \lambda_I \cdot x \right\} \right| - 2^{n-1}.$$

Relation of the DLCT to the Fourier-Walsh transform of the DDT. We assert that each row of the DLCT is equal (up to normalization) to the Fourier-Walsh transform of the corresponding row of the DDT. Formally, for each $\varDelta \in \{0,1\}^n$, denote the Boolean function which corresponds to the $\varDelta$'s row of the DDT by $f_\varDelta$. That is, $f_\varDelta : \{0,1\}^m \to \mathbb{R}$ is defined by

$$f_\varDelta(\varDelta') = DDT_S(\varDelta, \varDelta') = \left|\{x \in \{0,1\}^n \middle| S(x) \oplus S(x \oplus \varDelta) = \varDelta'\}\right|.$$

Proposition 1. *For any $\lambda \in \{0,1\}^m$, we have $DLCT(\varDelta, \lambda) = 2^{m-1}\hat{f}_\varDelta(\lambda)$.*

Proof. By the definitions of the DLCT and of the Fourier-Walsh transform, we have

$$\begin{aligned}
DLCT_S(\varDelta, \lambda) &= \left|\left\{x \middle| \lambda \cdot S(x) = \lambda \cdot S(x \oplus \varDelta)\right\}\right| - 2^{n-1} \\
&= \frac{1}{2}\left(\left|\left\{x \middle| \lambda \cdot S(x) = \lambda \cdot S(x \oplus \varDelta)\right\}\right| - \left|\left\{x \middle| \lambda \cdot S(x) \neq \lambda \cdot S(x \oplus \varDelta)\right\}\right|\right) \\
&= \frac{1}{2}\left(\left|\left\{x \middle| \lambda \cdot (S(x) \oplus S(x \oplus \varDelta)) = 0\right\}\right| - \left|\left\{x \middle| \lambda \cdot (S(x) \oplus S(x \oplus \varDelta)) = 1\right\}\right|\right) \\
&= \frac{1}{2}\left(\sum_{\{\varDelta' : \varDelta' \cdot \lambda = 0\}} f_\varDelta(\varDelta') - \sum_{\{\varDelta' : \varDelta' \cdot \lambda = 1\}} f_\varDelta(\varDelta')\right) = 2^{m-1}\hat{f}_\varDelta(\lambda),
\end{aligned}$$

as asserted. ■

A theoretical implication. The relation of the DLCT to the Fourier-Walsh transform of the DDT yields an interesting theoretical insight on the differential-linear attack.

It is well-known that the DDT and the LAT have the following mathematical interpretations.

- *The DDT:* If we model the evolution of differences through the encryption process of a plaintext pair as a *Markov chain*, then the DDT is simply the *transition matrix* of the chain (see, e.g., [26]). In this regard, a differential attack utilizes a classical probability-theoretic tool for cryptanalysis.
- *The LAT:* For each mask λ, the λ's column of the LAT is equal (up to normalization) to the Fourier-Walsh transform of the Boolean function $x \mapsto \lambda \cdot S(x)$ (see, e.g., [15]). In this regard, linear cryptanalysis studies the function S via its Fourier transform, as is commonly done in Boolean function analysis (see, e.g., [34]).

Proposition 1 shows that each row of the DLCT is equal (up to normalization) to the Fourier-Walsh transform of the corresponding row of the DDT. Since the DLCT of the entire encryption scheme E completely captures DL attacks as shown above, this implies that the differential-linear attack utilizes an interesting combination of probabilistic and Fourier-analytic techniques: it considers the probability-theoretic transition matrix of a Markov chain associated with the function, and studies it via its Fourier-Walsh transform.

A practical implication. It is well-known that the Fourier-Walsh transform of a function $f : \{0,1\}^m \to \mathbb{R}$ can be computed in time $O(m \cdot 2^m)$ operations. Since each row of the DDT of S can be easily constructed in time $O(2^n)$ operations, Proposition 1 implies that each row of the DLCT can be computed in time $O(2^n + m2^m)$ operations, and that the entire DLCT can be computed in time $O(2^{2n} + m2^{m+n})$ operations. This significantly improves over the trivial algorithm which requires $O(2^{2n+m})$ operations.

This speedup is practically important as it allows us to compute the DLCT for larger parts of the cipher, and thus, obtain a more accurate estimate of the complexity of the DL attack. For example, in the attack on 8-round DES presented in Section 6, we compute one DLCT entry for three rounds of DES as a single unit, and so the ability to compute the DLCT efficiently is crucial.

3 Differential-Linear Cryptanalysis of Ascon, Revisited

Ascon [18] is an authenticated encryption algorithm that was recently selected to the final round of the CAESAR [16] competition. In [19], Dobraunig et al. presented a practical differential-linear attack on up to 5 rounds of the Ascon permutation, based on a 4-round DL distinguisher. The authors of [19] stated that while by the theory of the DL attack, the overall bias of the approximation is expected to be 2^{-20}, experiments show that the bias is significantly higher – 2^{-2}. They attributed the difference between practice and the theoretical estimate to multiple linear approximations that affect the overall bias, and used the correct value in their attack. We recompute the bias of the distinguisher using the DLCT and show that a large part of the discrepancy results from dependency between the two subciphers.

In order to recompute the bias, we have to provide some more details on the specific distinguisher used in [19]. We present the distinguisher only schematically.

The theoretical analysis of [19]. The DL distinguisher of [19] targets a 4-round reduced variant of Ascon denoted by E and decomposed as $E = E_1 \circ E_0$, where E_0 consists of rounds 1–2 and E_1 consists of rounds 3–4. For E_0, the distinguisher uses a differential characteristic of the form

$$\varDelta_0 \xrightarrow[L\circ S]{p_0=2^{-2}} \varDelta_1 \xrightarrow[L\circ S]{p_1=2^{-3}} \varDelta_2,$$

where $\varDelta_2$ is a truncated difference. For E_1, the distinguisher uses a linear approximation of the form

$$\lambda_0 \xrightarrow[L\circ S]{q_0=2^{-7}} \lambda_1 \xrightarrow[L\circ S]{q_1=2^{-2}} \lambda_2,$$

where λ_2 consists of a single bit, and all nonzero bits of the mask λ_0 are included in S-boxes in which the all the input bits are known to be zero in $\varDelta_2$. Using the naive complexity analysis of the DL attack (i.e., Equation (2) above), the authors of [19] concluded that the theoretical estimate for the overall bias of the approximation is $2 \cdot 2^{-5} \cdot (2^{-8})^2 = 2^{-20}$. On the other hand, they found experimentally that the bias is as high as 2^{-2}.[4]

Partial explanation of the discrepancy using only the trivial property of the DLCT. As mentioned above, the linear approximation of E_1 was chosen by Dobraunig et al. in such a way that all nonzero bits of the mask λ_0 are included in S-boxes in which all the input bits are known to be zero in $\varDelta_2$. By the trivial property of the DLCT presented in Section 2, this implies that the linear approximation in round 3 holds with bias 1/2, instead of the theoretical bias $q_0 = 2^{-7}$. Therefore, the estimated bias of the approximation is increased to $2 \cdot 2^{-5} \cdot (2^{-2})^2 = 2^{-8}$, which is already much higher than 2^{-20}.

Analysis using the DLCT. We now obtain a higher bias of 2^{-5} by revisiting the analysis using the DLCT. Let us decompose E into $E = E_1' \circ E_m \circ E_0'$, where E_0' consists of rounds 1–3, E_m consists of round 4, and $E_1' = Id$. Note that since in the DL distinguisher of [19], the output mask λ_2 consists of the MSB in the output of S-box no. 9, we are only interested in the entries of the DLCT of that S-box (which we denote by S). For E_0, we use a differential of the form $\varDelta_0 \xrightarrow[\text{3 rounds}]{p=2^{-3}} \varDelta_3$, where $\varDelta_0$ is the input difference of the DL distinguisher of [19]. In our value of $\varDelta_3$, three of the input bits to S-box no. 9 are known to be zero; specifically, the input is of the form ?0?00. (Note that the S-box is from 5 bits to 5 bits). The relevant normalized entries of the DLCT of S satisfy:

$$\overline{DLCT}_S(16,16) = 0, \overline{DLCT}_S(4,16) = 0,$$
$$\overline{DLCT}_S(20,16) = 2^{-1}, \text{ and } \overline{DLCT}_S(0,16) = 2^{-1}.$$

Hence, assuming that each input difference of S occurs in $\varDelta_3$ with the same probability 2^{-2} and using Equation (4), we obtain the estimate

$$4 \cdot 2^{-3} \cdot 2^{-2}(0 + 0 + 2^{-1} + 2^{-1}) \cdot (2^{-1})^2 = 2^{-5}$$

for the overall bias of the DL distinguisher of [19]. This value is, of course, much lower than the experimentally obtained bias of 2^{-2} (which may be explained by the effect of other differentials and linear approximations). On the other hand, it is significantly higher than the value 2^{-20} which follows from the classical DL framework. This demonstrates the importance of taking the dependency between subciphers into account, which the DLCT facilitates in an easy manner.

4 Differential-Linear Cryptanalysis of Serpent, Revisited

One of the first applications of the DL technique is an attack on the AES finalist Serpent [1] presented in [5]. The attack is based on a 9-round DL distinguisher

[4] We emphasize that the results of [19] were not affected by the theoretical estimate, since the authors of [19] used the experimentally verified value instead of the theoretically computed value.

with bias of 2^{-59} and targets an 11-round variant of the cipher. An improved attack was presented in [20]. The authors of [20] performed experiments with reduced round variants of Serpent, and concluded that the actual bias of the approximation is $2^{-57.75}$ and not 2^{-59}. Using the improved bias, they extended the attack to 12 rounds of Serpent (out of its 32 rounds) yielding the best currently known attack on the cipher.

In [20], the increased bias was attributed to the existence of other approximations that affect the overall bias. In this section we recompute the bias of the distinguisher using the DLCT and obtain the value $2^{-57.68}$, which is very close to the experimental value. Thus, we conclude that the increased bias in the experiment results mostly from the dependency between the two subciphers.

In order to recompute the bias, we have to provide some more details on the specific distinguisher used in [5]. For sake of clarity, we present it only schematically, and refer the reader to [5] for the exact difference and mask values.

The analysis of [5]. The DL distinguisher of [5] targets a 9-round reduced variant of Serpent that starts with round 2 of the cipher. This variant is denoted by E and decomposed as $E = E_1 \circ E_0$, where E_0 consists of rounds 2–4 and E_1 consists of rounds 5–10. For E_0, the distinguisher uses a differential characteristic of the form

$$\Delta_0 \xrightarrow[L\circ S_2]{p_0=2^{-5}} \Delta_1 \xrightarrow[L\circ S_3]{p_1=2^{-1}} \Delta_2 \xrightarrow[L\circ S_4]{p_2=1} \Delta_3,$$

where Δ_2, Δ_3 are truncated differences. For E_1, the distinguisher uses a linear approximation of the form

$$\lambda_0 \xrightarrow[L\circ S_5]{q_0=2^{-5}} \lambda_1 \xrightarrow[\text{5 rounds}]{q_1=2^{-23}} \lambda_7,$$

where all nonzero bits of the mask λ_0 are included in the bits that are known to be zero in Δ_3. Using the naive complexity analysis of the DL attack (i.e., Equation (2) above), the authors of [5] concluded that the overall bias of the approximation is $2 \cdot 2^{-6} \cdot (2^{-27})^2 = 2^{-59}$. (Actually, in their attack they used the lower value of 2^{-60} due to the effect of other differentials.)

The experimental results of [20]. The authors of [20] checked experimentally the first 4 rounds of the DL distinguisher of [5] (i.e., a 4-round distinguisher which starts with the difference Δ_0 and ends with the mask λ_1) and found that its bias is $2^{-13.75}$, instead of the theoretical estimate $2 \cdot 2^{-6} \cdot (2^{-5})^2 = 2^{-15}$. They concluded that the overall bias of the 9-round distinguisher is $2^{-57.75}$ instead of 2^{-59}, and used the conclusion to extend the key-recovery attack based on the distinguisher from 11 rounds to 12 rounds.

Analysis using the DLCT. We considered a 3-round variant of Serpent that starts at round 3, denoted it by E', and computed the normalized DLCT entry $\overline{DLCT}_{E'}(\Delta_1, \lambda_1)$. (Note that computing the entire DLCT for E' is infeasible. However, due to the low diffusion of Serpent, one can compute efficiently part of the entries, including the entry we needed). We obtained $\overline{DLCT}_S(\Delta_1, \lambda_1) = 2^{-8.68}$.

Using Equation (4) (for the case of $E_1' = Id$) we conclude that the bias of the 4-round distinguisher examined in [20] is $p_1 \cdot \overline{DLCT}_S(\Delta_1, \lambda_1) = 2^{-5} \cdot 2^{-8.68} = 2^{-13.68}$, which is very close to the experimental result $2^{-13.75}$ of [20].

Note that we obtained an estimate which is very close to the actual value, using only the naive Equation (4) and not the more accurate Equation (5) that takes into account the effect of other differentials. This shows that the increased bias found experimentally in [20] follows almost solely from dependency between the subciphers, and demonstrates how the DLCT methodology can be used for obtaining an accurate estimate of the DL attack complexity.

5 Improved Differential-Linear Attack on ICEPOLE

ICEPOLE is an authenticated encryption cipher based on the duplex construction proposed by Morawiecki et al. submitted to the CAESAR competition [32]. The main two versions, ICEPOLE128 and ICEPOLE128a are initialized with a 128-bit key. In addition ICEPOLE128 accepts 128-bit nonce and 128-bit secret message number, in comparison, ICEPOLE128a accepts 96-bit nonce and 0-bit secret message number (to serve as a drop-in replacement for AES-128-GCM).[5] After initialization, the associated data is absorbed into the 1280-bit state. For the processing of the plaintext (encryption and authentication), a block of 1024 bits is extracted (to be XORed to the plaintext) and the plaintext is XORed into 1024 bits of the state.[6] This state is then updated using 6-round Permutation P_6 which iterates a round function P 6 times over 1280 bits. After the entire plaintext has been encrypted, a tag is produced by extracting bits of the internal state. The entire process is depicted in Figure 2.[7]

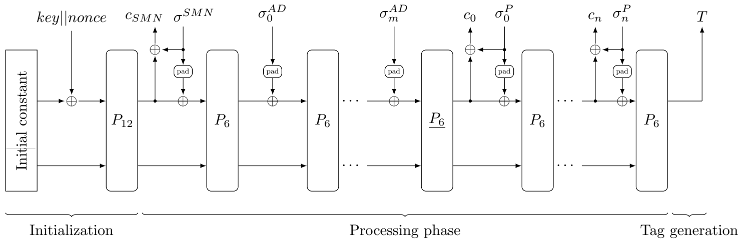
After ICEPOLE has been introduced, Huang et al. presented a differential-linear attack against ICEPOLE-128 and ICEPOLE-128a [22]. The attack recovers the internal state using a differential-linear attack, where the bias of the differential-linear depends on the value of some bits. Hence, observing the bias in the output allows identifying internal state bits. After full recovery of the internal state, one can extract the secret key (as long as the scheme is not using secret message numbers) or forge new messages (when using secret message numbers).

[5] We note that ICEPOLE256a is a variant designed to serve as a drop-in replacement for AES-256-GCM, thus it has the same parameters as AES-256-GCM.

[6] Actually, two additional bits are appended – the frame bit which is set to 0 in all blocks but the last authenticated data block and the last message block, and a padding bit, but their rule and effect on the attack are negligible.

[7] We disregard the exact initialization and the handling of associated data which are of no relevance to this paper. The interested reader is referred to [32] for more information.

Fig. 2. General Structure of ICEPOLE



5.1 A Short Description of ICEPOLE-128

We first note that there are three variants of ICEPOLE (ICEPOLE-128, ICEPOLE-128a, and ICEPOLE-256), but our attacks and description concern only the 128-bit variants, ICEPOLE-128 and ICEPOLE-128a.

The internal state of ICEPOLE, denoted by S is composed of 20 64-bit words organized in a 4-by-5 matrix. We follow [32] notations: the bit $S[x][y][z]$ is the z'th bit of the word at position (x, y) where $0 \leq x \leq 3$, $0 \leq y \leq 4$, and $0 \leq z \leq 63$. This bit is considered to be bit $64(x + 4y) + z$ of the state. The first n bits of S are denoted by by $S_{\lfloor n \rfloor}$. The z'th *slice* of S is a 4-by-5 binary matrix $(S[x][y][z])_{x,y}$. When z and x are fixed, the 5-bit vector $S([x][y][z])_y$ is called a row.

The round function P is composed of five operations, $P = \kappa \circ \psi \circ \pi \circ \rho \circ \mu$ which are:

- μ operates on each of the 64 slices independently by treating each 20-bit slice as a vector $(Z_0, Z_1, Z_2, Z_3) \in GF(2^5)$ and multiplying this vector by the MDS matrix
$$M = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 1 & 18 & 2 \\ 1 & 2 & 1 & 18 \\ 1 & 18 & 2 & 1 \end{pmatrix}.$$
The multiplication is done over $GF(2^5)$ (with the irreducible polynomial $x^5 + x^2 + 1$).
- ρ is a cyclic rotation applied to each of the 20 64-bit words of S. Each word (x, y) is rotated by a different constant, i.e., $S[x][y] = S[x][y] \lll \text{offsets}[x][y]$ where the table of offsets$[x][y]$ can be found in [32].
- π reorders the words in S by moving the word $S[x][y]$ into $S[x'][y']$ according to the formula:
$$\begin{cases} x' = (x + y) \bmod 4 \\ y' = (((x + y) \bmod 4) + y + 1) \bmod 5 \end{cases}$$
- ψ applies a 5-bit S-box to each of the 256 rows of the state.
- κ adds a round constant (constant[round]) to $S[0][0]$. The constants are generated by an LFSR and can be found in [32].

For the sake of clarity, we shall denote the linear parts of the round function by $\mathcal{L} = \pi \circ \rho \circ \mu$.

As mentioned before, the internal state S is initialized using a constant which is XORed with $(key||nonce)$ value. After than P_{12} (12 rounds of P are used to mix the key and nonce into the state). If secret message numbers are used, they are encrypted using the duplex operation. The associated data is chopped into blocks of 1026-bit each (after padding). They are absorbed into the state S, and then the encryption/authentication of the plaintext takes place following the duplex operation using P_6. After the entire plaintext is processed, P_6 is applied to the internal state (or P_{12} in ICEPOLEv2), and the 128-bit tag is computed as $T_0 = S[0][0]$, $T_1 = S[1][0]$.

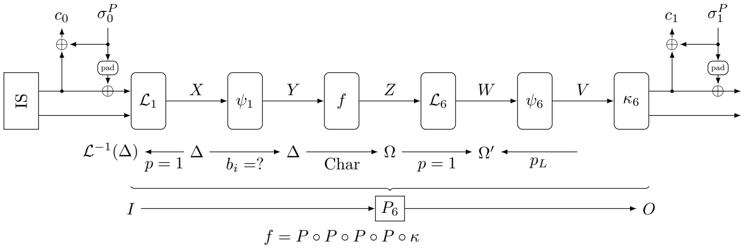
In the followings, we use e_i to denote a 64-bit word which is 0 in all bits, besides the i'th bit.

5.2 Huang et al.'s Differential-Linear Attack on ICEPOLE-128/ICEPOLE-128a

Huang et al. have presented a differential-linear attack against ICEPOLE-128 and ICEPOLE-128a in the repeated nonce settings [22]. As the attack recovers the internal state, if the scheme is not using a secret message number, then one can obtain the key by inverting the internal state. Otherwise, the recovery of the internal state allows encrypting/authenticating any data.

The attack targets the first application of P_6 after the plaintext is introduced by injecting differences through the plaintext block σ_0 and observing biases in the key stream used to encrypt σ_1. Its general structure is depicted in Figure 3. For the sake of its description, we denote by I the input to P_6 (after the XOR with the plaintext) and by O the output of P_6. Moreover, we denote by ψ_i, $\mathcal{L}_i$, and κ_i the ψ, $\mathcal{L}$, and κ of the i'th round.

Fig. 3. Differential-Linear Attack on the "First" P_6 of ICEPOLE



The attack of [22] uses the 1024 bits of I and O which can be easily obtained by knowing the plaintext and ciphertext blocks for a 2-block long message. The attack introduces differences in the first plaintext block, which does not affect the fifth column of I. This difference is transformed into an input difference Δ after the application of the linear layer $\mathcal{L}$ (i.e., the introduced difference into the state is $\mathcal{L}^{-1}(\Delta)$). Due to the MDS property of μ and the zero difference in the fifth column, Δ must have at least two active S-boxes. The behavior of these two active S-boxes, namely, the probability of the differential transition through them highly depends on the actual value of some bit b_i (or two bits). Hence the input difference Δ^* of the differential-linear approximation appears with different probabilities, depending on the value of this bit. Luckily for us, this bit (or pair of bits) is the outcome of XORing an unknown input bit (from the fifth column of I) with known bits (which are controlled by the adversary). This allows the adversary to partition the plaintext pairs into sets according to the possible values of b_i, where for the "correct" set, we expect a significantly higher bias.

The differential-linear approximation $\Delta^* \to \lambda$ covers the 4 rounds until round 6, and can be extended until the end of the linear layer $\mathcal{L}_6$. We note that in ψ_1, the differential characteristic in use is $\Delta^* \to \Delta = \Delta^*$. Interestingly, ψ has a very useful property – given the 4 least significant bits of the output, one can partially recover the input. Table 2 suggests the values, and the probability that partial information can be found given these 4 output bits. Hence, any differential-linear whose output mask can be deduced from the partial information can be used with some probability, which we denote by p_L.

Table 2. Deducing Input Bits of ψ from the Four LSBs of the Output

	Input				
Output	MSB	Bit 3	Bit 2	Bit 1	LSB
?0000	1	?	1	?	1
?0001	?	?	?	?	?
?0010	?	0	?	1	0
?0011	?	?	?	?	1
?0100	?	?	?	0	?
?0101	?	0	?	0	?
?0110	?	0	?	1	0
?0111	?	?	?	1	1
?1000	?	1	0	1	0
?1001	?	?	0	?	1
?1010	?	1	0	0	0
?1011	?	?	0	?	1
?1100	?	?	1	0	1
?1101	?	1	1	0	0
?1110	?	1	1	1	0
?1111	0	0	?	?	?
Probability (p_L)	1/8	1/2	1/2	5/8	3/4

? – unknown value

Due to the structure of ICEPOLE, any differential characteristic and any linear approximation can be rotated (by rotating each word, respectively). Hence, the attack is repeated with the 64 rotated versions of the differential-linear approximation. Each time, it recovers the bit b_i (which affects bias of the approximation). This is done by encrypting multiple pairs of plaintexts with input difference Δ with two active S-boxes (and covering all possible values of the recovered bit b_i), and observing the set which has the highest bias.

Actually, instead of taking all ciphertext pairs, only the pairs which can be used to predict the input of ψ_6 from their 1024-bit outputs O are considered (as each ciphertext can be used with probability, this is actually probability p_L^2). The different approach of extending the differential-linear approximation to cover ψ_6 leads to much worse performance (as there are many active S-boxes in the ψ_6 layer).

After studying the differentials and linear approximations that can be used for the attack, Huang et al. [22] identified 5 input difference patterns $\Delta_1, \Delta_2, \ldots, \Delta_5$ that after $\mathcal{L}$ activate only two S-boxes, as well as two good linear approximations $\lambda_1^{mid} \to \lambda_1$ and $\lambda_2^{mid} \to \lambda_2$. Given the word-oriented nature of the permutation P, one can rotate the differences (or masks) by rotating each word of the mask/difference. Hence, for each bit position it is possible to consider all the combinations of Δ_i and λ_j and experimentally calculate the bias of the resulting differential-linear.

The actual attack tries to find the last column of I (as the first four can be trivially deduced). Denote this fifth column by the four words (U_0, U_1, U_2, U_3). The first phase recovers U_0 and U_3, the second phase recovers U_2, and finally the third phase recovers U_1. All phases follow an essentially similar process – a differential-linear approximation is built from a differential characteristic which probability (in the first round) depends on the value of some specific (unknown) bits. Then, by observing enough plaintext/ciphertext pairs and evaluating the bias, one can determine the unknown bits, from which corresponding bits of U_i are computed.

Following the above steps, we give detailed explanation on how to find the first bits of U_0 and U_3 in the attack. For that phase, Huang et al. propose to use the following Δ_2:

$$\Delta_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & e_{10} & e_{41} & 0 & 0 \end{pmatrix}$$

which under $\mathcal{L}^{-1}$ has differences in the LSBs of the words $S[0][2]$, $S[1][0]$, $S[1][1]$, $S[1][2]$, $S[1][3]$, $S[2][1]$, $S[2][3]$, $S[3][0]$, and $S[3][2]$. Another technicality is that the adversary fixes 18 bits (by knowing the first four columns of I she can select the corresponding σ^0), then the probability of the differential transition depends on two unknown bits – for one of the four possibilities it is significantly higher probability than for the rest, as presented in Table 3. The specific fixed bits and their values is given in [22].

The linear mask in use ends with the mask

$$\lambda_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & e_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow{\mathcal{L}_6} \lambda_1' = \begin{pmatrix} e_{18} & e_0 & 0 & e_{43} & 0 \\ 0 & e_2 & 0 & 0 & 0 \\ 0 & e_{21} & e_{61} & 0 & 0 \\ 0 & 0 & e_{41} & e_{56} & 0 \end{pmatrix}.$$

Using Table 2, this suggests that with probability $p_L = 2^{-6.45}$ one can compute the output mask from a given output O. As each pair requires the evaluation of two O's, the probability that a pair can be used for the analysis is $p_L^2 = 2^{-11.9}$.

For the specific differential-linear characteristics presented above the two unknown at the entrance of ψ_1 which can be recovered are $b_1 = U_3^{31} \oplus a_0$ and $b_3 = U_0^{43} \oplus U_3^{43} \oplus a_1$, where a_0 and a_1 can be computed from the four known columns of I. Table 3 offers the different biases as a function of b_1 and b_3. These biases were experimentally computed in [22] by taking 2^{30} plaintexts pairs with the required values fixed.

To conclude, given the above differential-linear characteristic, the recovery of two bits b_1 and b_3 is as follows:

1. Collect N plaintext pairs with the required input difference and the 18 bits fixed.
2. For each pair, check whether one can deduce the bits that enter the linear mask of ψ_6 for both ciphertexts.
3. Divide the remaining pairs into four sets according to the value of unknown values of b_1 and b_3.[8]
4. Find the set with the maximal bias (which suggests the correct values of b_1 and b_3). Compute from b_1, b_3 and the known bits the value of the unknown bits.

The analysis shows that when taking $N = 2^{33.9}$ plaintext pairs (with 18 bits fixed) we obtain about $2^{33.9} \cdot p_L^2 = 2^{21}$ pairs which can be analyzed (as we know the input linear mask to ψ_6). These pairs can be divided into four sets of about 2^{19} pairs each, one of which with a bias of $2^{-7.3}$, which is significantly higher than the rest, and thus can be easily detected. The data complexity is thus $2^{33.9}$ pairs of 2-block messages for each pair of bits b_1, b_3 recovered, or a total of $64 \cdot 2 \cdot 2 \cdot 2^{33.9} = 2^{41.9}$ 1024-bit data blocks. We list in Table 3 the different bits recovered in each phase, the relevant p_L, and the data complexity. The full details are available at [22].

5.3 Our New Results on ICEPOLE-128/ICEPOLE-128a

The main reason the attack of [22] used a single-bit mask for the output is to ensure a low hamming weight mask. This was chosen to optimize the two conflicting effects of λ on the complexity of the attack – the more active bits in $\mathcal{L}_6(\lambda)$ there are (which translates to more active bits in λ when λ is of low hamming weight), the lower p_L is. At the same time, λ affects the bias as it sets the output mask of ψ_5, suggesting that constraints on λ may lead to sub-optimal linear approximations.

Moreover, as the actual biases were measured experimentally (rather than analytically) we decided to pick a slightly different approach. Instead of studying single-bit λ we decided to try output masks with a single active S-box. This

[8] We remind the reader that these bits are the XOR of a fixed unknown bits from U_0 and U_3 with bits that are known to us).

Table 3. The Different Phases of the Attack of [22]

Phase	Recovered Bits	Value/$\log_2(bias)$	p_L	Data Complexity
1	$b_1^i = U_3^{31+i}, b_2^i = U_0^{43+i} \oplus U_3^{43+i}$ $i \in \{0, 1, \dots 63\}$	$(b_1 = 0, b_3 = 0)$ -13 $(b_1 = 0, b_3 = 1)$ -7.3 $(b_1 = 1, b_3 = 0)$ -13.9 $(b_1 = 1, b_3 = 1)$ -11.9	$2^{-6.45}$	$64 \cdot 2 \cdot 2 \cdot 2^{33.9} = 2^{41.9}$
2	$b_2^i = U_2^{24+i}$ $i \in \{0, 1, \dots 63\}$	$b_2 = 0$ -11 $b_2 = 1$ -15.4	$2^{-5.86}$	$64 \cdot 2 \cdot 2 \cdot 2^{36.7} = 2^{44.7}$
3	$b_{0,3}^i = U_1^{12+2i}, b_{1,1}^i = U_1^{13+2i}$ $i \in \{0, 1, \dots 31\}$	$(b_{0,3} = 0, b_{1,1} = 0)$ -11.2 $(b_{0,3} = 0, b_{1,1} = 1)$ -15.2 $(b_{0,3} = 1, b_{1,1} = 0)$ -16.4 $(b_{0,3} = 1, b_{1,1} = 1)$ -14.8	$2^{-5.86}$	$32 \cdot 2 \cdot 2 \cdot 2^{37.7} = 2^{44.7}$
Total				$2^{45.8}$

allowed raising the bias of the transition in ψ_5 from at most 3/16 to 4/16 (which is significant due to the quadratic effect on the bias, which translates to a quadratic effect on the data and time complexities).

The increase in the number of possible output masks carries with it a computational problem – one needs to cover more masks in the process of computing the bias, by a factor of almost 6, for any chosen input difference. Hence, instead of relying on multiple time consuming experiments for each input difference Δ, we use the DLCT of ψ to obtain estimates for the bias of the differential-linear approximation. This is done by taking the input difference Δ and computing for each S-box in ψ_5 the distribution of input differences (i.e., if the input difference is δ, determaining p_δ). Then, for all the active S-boxes in the mask leaving ψ_5, for each S-box' mask ω we compute $\sum_\delta p_\delta \cdot DLCT(\delta, \omega)$ to evaluate the probability of the differential-linear transition in ψ_5. As the evaluation of p_d for each S-box is independent of the mask ω, and as the DLCT is computed once, this offers a very efficient procedure.

The result is the discovery of better differential-linear approximation for the second and third phase. We give in Table 4 the new differential-linear approximations used in the second and third phase. Due to the reduced data required in the later phases, we also reduce the data complexity in the first round (to reduce the total data complexity) and change a bit the constraints on the actual values (but they serve the same purpose as in the original attack). One main difference is that the constraints are not on the values, but rather on the parity of some subsets of bits. We list these subsets in Table 5.

Using the new differential-linear approximations (and bit-fixing) we obtain an attack on the full ICEPOLE in complexity of $2^{41.58}$ data and time. Its phases are listed in Table 6. We note that the first phase has a slightly lower success rate. After presenting a new approach for generating the data that further reduces the data and time (to $2^{35.85}$) we discuss how to mitigate this slightly lower success rate.

Table 4. Our New Different-Linear Approximations for Phases 2 and 3 of the Attack

Phase	Δ	λ	λ'	p_L	Bias
2	$\begin{pmatrix} e_8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e_0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & e_{49} & e_{49} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} e_3 & e_{49} & 0 & 0 & 0 \\ 0 & e_{51} & 0 & 0 & 0 \\ 0 & 0 & e_{46} & 0 & 0 \\ 0 & 0 & e_{26} & e_{41} & 0 \end{pmatrix}$	$2^{-4.77}$	$2^{-8.88}$
3	$\begin{pmatrix} 0 & 0 & e_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e_1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & e_{43} & e_{43} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} e_{61} & e_{43} & 0 & 0 & 0 \\ 0 & e_{45} & 0 & 0 & 0 \\ 0 & 0 & e_{40} & 0 & 0 \\ 0 & 0 & e_{20} & e_{35} & 0 \end{pmatrix}$	$2^{-4.77}$	$2^{-9.49}$

Efficient Data Generation We note that each of the three phases of the attack is composed of 64 applications of the same attack up to rotating the differences/masks. Hence, if each such attack requires about N plaintext pairs, a trivial implementation requires $64N$ plaintext pairs. Luckily, there is a more efficient way to do so.

Our key observation is that one can select P_i in advance to satisfy all the conditions of the 64 possible rotations. This can be easily done when there is at least one word which has no conditions/restrictions. For such a P_i we test for each of the 64 rotations whether one can deduce the needed bits at the input of ψ_6. If so, we generate its counterpart P_i^* which satisfies the required difference, and apply the attack as before (with probability of p_L that P_i^* allows recovering the input of ψ_6).

This reduces the data complexity of Phase 1 from $2 \cdot 2 \cdot 64 \cdot 2^{32.8}$ plaintexts to $2 \cdot (2^{32.8} + 64 \cdot 2^{32.8-6.45}) = 2^{34.6}$ plaintexts (as for each of the 64 rotations of the differential-linear approximation there is probability of p_L that P_i is useful). Similar analysis reduces the data complexity of the second phase to $2 \cdot (2^{31.33} + 64 \cdot 2^{31.33-4.77}) = 2^{34.07}$ which is the same also for the third phase. Hence, the total data complexity of the attack is $2^{35.85}$ chosen plaintexts.

We note that the reduced data complexity of the first phase may negatively affect the success rate. Moreover, an error in the first phase is expected to cause errors in the next phases. However, we note that one can easily test the obtained values for correctness. If the recovered internal state is not accurate, the adversary can exhaustively test internal states of hamming distance up to 5 from the extracted one in time of $\begin{pmatrix} 256 \\ 5 \end{pmatrix} \approx 2^{33.1}$ recomputations (using simple linear algebra). In other words, as long as the attack has at most 5 wrong bits, the correct internal value can be extracted.

5.4 Experimental Verification of Our Attack

We have experimentally verified our attack. We run the full attack 16 times, using a random key and nonce. The machine was a virtual machine on the Azure infrastructure (instance Standard_F64s_v2). The machine has 64 vCPUs

Table 5. Bit Subsets Fixed for Our Attack

Phase	Subset	Parity(Subset)	Subset	Parity(Subset)
1	$\begin{pmatrix} 0 & e_4 & 0 & 0 & 0 \\ e_4 & 0 & 0 & 0 & 0 \\ 0 & e_4 & 0 & 0 & 0 \\ e_4 & 0 & e_4 & 0 & 0 \end{pmatrix}$	1	$\begin{pmatrix} 0 & e_{35} & 0 & 0 & 0 \\ e_{35} & 0 & 0 & 0 & 0 \\ 0 & e_{35} & 0 & 0 & 0 \\ e_{35} & 0 & e_{35} & 0 & 0 \end{pmatrix}$	1
	$\begin{pmatrix} 0 & 0 & e_{33} & 0 & 0 \\ 0 & 0 & 0 & e_{33} & 0 \\ 0 & 0 & 0 & e_{33} & 0 \\ 0 & 0 & 0 & e_{33} & 0 \end{pmatrix}$	0	$\begin{pmatrix} 0 & 0 & e_0 & 0 & 0 \\ 0 & 0 & 0 & e_0 & 0 \\ 0 & 0 & 0 & e_0 & 0 \\ 0 & 0 & 0 & e_0 & 0 \end{pmatrix}$	0
2	$\begin{pmatrix} 0 & 0 & 0 & e_{27} & 0 \\ 0 & 0 & 0 & e_{27} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e_{27} & 0 & 0 \end{pmatrix}$	1	$\begin{pmatrix} 0 & e_{17} & 0 & 0 & 0 \\ e_{17} & 0 & e_{17} & 0 & 0 \\ e_{17} & 0 & 0 & 0 & 0 \\ 0 & e_{17} & 0 & 0 & 0 \end{pmatrix}$	1
	$\begin{pmatrix} 0 & e_{58} & 0 & 0 & 0 \\ e_{58} & 0 & 0 & 0 & 0 \\ 0 & e_{58} & 0 & 0 & 0 \\ e_{58} & 0 & e_{58} & 0 & 0 \end{pmatrix}$	1	$\begin{pmatrix} 0 & 0 & 0 & 0 & e_8 \\ e_8 & 0 & 0 & 0 & 0 \\ e_8 & 0 & 0 & 0 & 0 \\ e_8 & 0 & 0 & 0 & 0 \end{pmatrix}$	0
	$\begin{pmatrix} 0 & 0 & e_{23} & 0 & 0 \\ 0 & 0 & 0 & e_{23} & 0 \\ 0 & 0 & 0 & e_{23} & 0 \\ 0 & 0 & 0 & e_{23} & 0 \end{pmatrix}$	0		
3	$\begin{pmatrix} 0 & 0 & 0 & 0 & e_0 \\ e_0 & 0 & 0 & 0 & 0 \\ e_0 & 0 & 0 & 0 & 0 \\ e_0 & 0 & 0 & 0 & 0 \end{pmatrix}$	1	$\begin{pmatrix} 0 & 0 & 0 & e_{19} & 0 \\ 0 & 0 & 0 & e_{19} & 0 \\ 0 & 0 & 0 & 0 & e_{19} \\ 0 & 0 & e_{19} & 0 & 0 \end{pmatrix}$	1
	$\begin{pmatrix} 0 & 0 & e_{24} & 0 & 0 \\ 0 & 0 & 0 & e_{24} & 0 \\ 0 & 0 & 0 & e_{24} & 0 \\ 0 & 0 & 0 & e_{24} & 0 \end{pmatrix}$	1	$\begin{pmatrix} 0 & 0 & e_{21} & 0 & 0 \\ 0 & e_{21} & 0 & 0 & 0 \\ 0 & 0 & e_{21} & 0 & 0 \\ 0 & 0 & 0 & e_{21} & 0 \end{pmatrix}$	0
	$\begin{pmatrix} 0 & 0 & e_{55} & 0 & 0 \\ 0 & 0 & e_{55} & 0 & 0 \\ 0 & 0 & 0 & e_{55} & 0 \\ 0 & e_{55} & 0 & 0 & e_{55} \end{pmatrix}$	0		

(Intel Xeon 8168 processor) with 128 GB RAM running Ubuntu 18.04.1 TLS. We have used the official ICEPOLE code (written in C), while our attack was written in C++. Compiling with gcc-7.3.0 using the optimization flag -O3, each of the attack's instances took about an hour. Its code is available at `https://github.com/cryptobiu`.

Out of the 16 experiments, 11 recovered the exact internal state. In 4 of them, a single-bit error took place in phase 3 of the attack (resulting in a single-bit error in the proposed internal state). Finally, in one experiment, a single-bit error took place in the first phase, resulting in three bits error (single-bit error in the second phase and in the third-phase). Of course, once the single-bit error in the first phase is fixed, then the errors in the other phases are resolved as well.

Table 6. The Different Phases of the New Attack

Phase	Recovered Bits	Value/$\log_2(bias)$	p_L	Data Complexity
1	$b_1^i = U_3^{31+i}, b_2^i = U_0^{43+i} \oplus U_3^{43+i}$ $i \in \{0, 1, \ldots 63\}$	$(b_1 = 0, b_3 = 0)$ -13 $(b_1 = 0, b_3 = 1)$ -7.3 $(b_1 = 1, b_3 = 0)$ -13.9 $(b_1 = 1, b_3 = 1)$ -11.9	$2^{-6.45}$	$64 \cdot 2 \cdot 2 \cdot 2^{32.8} = 2^{40.8}$
2	$b_4^i = U_2^{27+i}$ $i \in \{0, 1, \ldots 63\}$	$b_4 = 0$ -14.32 $b_4 = 1$ -8.8	$2^{-4.77}$	$64 \cdot 2 \cdot 2 \cdot 2^{31.33} = 2^{39.33}$
3	$b_3^i = U_1^{21+i}$ $i \in \{0, 1, \ldots 63\}$	$b_3 = 0$ -9.49 $b_3 = 1$ -13	$2^{-4.77}$	$64 \cdot 2 \cdot 2 \cdot 2^{31.33} = 2^{39.33}$
Total				$2^{41.58}$

Hence, we conclude that all experiments succeeded to recover the internal state (or were sufficiently close to the correct one) using $2^{34.85}$ 2-block plaintexts.

6 Improved Differential-Linear Attack on 8-Round DES

refined analysis of the DLCT, we found out that the attack can be improved by replacing the differential characteristic and the linear approximation with another combination of a characteristic and an approximation, which leads to a higher bias due to dependency between the two underlying subciphers. First we briefly recall the structure of DES and describe the attack of [4], and then we present our improved attack.

In this section we use the DLCT methodology to revisit the DL attack on 8-round DES [35] presented by Biham et al. [4]. We show that the attack can be improved by replacing the differential characteristic and the linear approximation with another combination of a characteristic and an approximation, which leads to a higher bias due to dependency between the two underlying subciphers.

6.1 The DL attack of [4] on 8-round DES

The attack of [4] is based on a 7-round DL distinguisher. Denote a 7-round variant of DES by E. The distinguisher uses the decomposition $E = E_1 \circ E_0$, where E_0 consists of rounds 1–4 and E_1 consists of rounds 5–7. For E_0, it uses the truncated differential

$$0x00808200\ 60000000 = \Delta_I \xrightarrow[E_0]{p=\frac{14}{64}} \Delta_O = 0x????M???\ 00W0XY0Z,$$

where $M \in \{0, 1, 2, \ldots, 7\}, W, X \in \{0, 8\}$, and $Y, Z \in \{0, 2\}$. The characteristic is composed of a 1-round characteristic with probability $\frac{14}{64}$ and a 3-round truncated characteristic with probability 1.

For E_1 (rounds 5–7), it uses the linear approximation

$$0x21040080\ 00008000 = \lambda_I \xrightarrow[E_1]{q=2\cdot(\frac{-20}{64})^2} \lambda_I.$$

Note that all nonzero bits of λ_I are included in the bits that are known to be 0 in Δ_O. Using the naive complexity analysis of the DL attack (i.e., Equation (2) above), the authors of [4] concluded that the overall bias of the approximation is $2pq^2 = 2^{-5.91}$.

6.2 Our improved DL attack on 8-round DES

At a first glance, it seems unlikely that the distinguisher of [4] can be improved. Indeed, the linear approximation it uses is known to be the best 3-round linear approximation of DES, and the only round in the differential characteristic whose probability is less than 1, is almost the best possible (the highest possible probability being $\frac{16}{64}$). In fact, we verified experimentally that for any other combination of a differential with probability p' and a linear approximation with bias q', we have $2p'(q')^2 < 2^{-5.91}$.

Nevertheless, we obtain a higher bias, using the dependency between the subciphers. We decompose E as $E = E_1' \circ E_m \circ E_0'$, where E_0' consists of rounds 1–2, E_m consists of rounds 3–5, and E_1' consists of rounds 6–7. For E_0', we use the differential

$$0x00200008\ 00000400 = \Delta_I \xrightarrow[E_0']{p'=\frac{16}{64}} \Delta_O = 0x60000000\ 00000000.$$

For E_1', we use the linear approximation

$$0x00808202\ 00000000 = \lambda_I \xrightarrow[E_1']{q'=\frac{-18}{64}} \lambda_O = 00808202\ 80000000.$$

For E_m, we use the DLCT entry[9]

$$\overline{DLCT}_{E_m}(0x60000000\ 00000000, 0x00808202\ 00000000) \approx 0.26.$$

Using Equation (4) above, we find that the overall bias of our approximation is

$$4p' \cdot \overline{DLCT}_{E_m}(\Delta_O, \lambda_I) \cdot (q')^2 = 4 \cdot \left(\frac{16}{64}\right)^2 \cdot 0.26 \cdot \left(\frac{-18}{64}\right)^2 = 2^{-5.6}. \qquad (6)$$

Since the data complexity of the DL attack is quadratic in the bias, the improvement from $2^{-5.91}$ to $2^{-5.6}$ reduces the data complexity of the attack of [4] on 8-round DES by a factor of about 1.5.

[9] This entry was computed by looking at all 3-round differential characteristics starting at input difference $0x60000000\ 00000000$, computing their output difference δ_i (and probability), and evaluating the bias of $\lambda_I \cdot \delta_i$. After summing over all differential characteristics, we have experimentally verified that this DLCT entry is indeed about 0.26.

Comparison between our distinguisher and the distinguisher of [4]. In order to compare our distinguisher to that of [4], we present the latter within the DLCT framework. It is composed of the differential $0x00200008\ 00000400 \xrightarrow{p'=\frac{14}{64}} 0x00000400\ 00000000$ for E'_0, the linear approximation $0x21040080\ 00000000 \xrightarrow{q'=\frac{-20}{64}} 0x21040080\ 00008000$ for E'_1, and the DLCT entry $\overline{DLCT}_{E_m}(0x00000400\ 00000000, 0x21040080\ 00000000) \approx 0.24$. Using Equation (4), its overall bias is $2^{-5.81}$. Note that while the value $p'(q')^2$ in the distinguisher of [4] is larger than the corresponding value in our distinguisher, the overall bias we obtain is higher due to the larger value in the DLCT. This emphasizes that the advantage of our new DL distinguisher stems mainly from dependency between the two subciphers, reflected in the DLCT.

Experimental verification. We experimentally verified the bias of our DL distinguisher, using 100 different keys and 500,000 plaintext pairs for each key. The average bias found in the experiments was $2^{-5.58}$, and the standard deviation was $2^{-10.43}$. This shows that the theoretical estimate of the bias using Equation (4) is tight in our case, and thus, demonstrates the strength of the DLCT as a tool for accurate evaluation of the DL attack complexity.

For sake of completeness, we verified experimentally also the distinguisher of Biham et al. We checked 100 different keys and 500,000 plaintext pairs for each key. The average bias found in the experiments was $2^{-5.72}$, and the standard deviation was $2^{-10.56}$. In addition, we verified that our DL distinguisher has the maximal bias among all 7-round DL distinguishers that start and end with a single active S-box. While we could not check 7-round DL distinguishers with more than one active S-box in the input difference or in the output bias, it seems highly unlikely that such a distinguisher will have a higher bias, even if it exploits the dependency between the subciphers.

Another 7-round DL distinguisher used in [4]. The authors of [4] present another 7-round DL distinguisher of DES, which they use in the key recovery attack on 9-round DES. (Its bias is somewhat lower, but it activates less S-boxes in the round before the distinguisher). We checked this distinguisher using the DLCT framework and found that its bias is $2^{-5.95}$, instead of $2^{-6.13}$ computed in [4]. We verified experimentally this result as well, and obtained average bias of $2^{-5.94}$ and standard deviation of $2^{-10.53}$. This slightly improved bias reduces the data complexity of the attack of [4] on 9-round DES by a factor of about 1.3.

7 Summary and Conclusions

In this paper we studied the effect of the dependency between the subciphers on the differential-linear attack. We showed that in various cases of interest, including previously published DL attacks on Ascon and Serpent, the dependency significantly affects the attack's complexity. We presented a new tool – the *differential-linear connectivity table* (DLCT) – which allows to (partially) take the dependency into account and to use it for making DL attacks more efficient. We showed a relation of the DLCT to the Fourier transform and deduced from it a new theoretical insight on the differential-linear attack. Finally, we demonstrated the strength of our new tool, by improving previously published DL attacks against ICEPOLE and 8-round DES.

Our objective in this paper was to introduce the DLCT and to present a few initial applications. Thus, several natural research directions are left for future work. The first is formalizing the relation of the DLCT with consideration of multiple linear approximations, as was done for the basic DL framework by Blondeau, Leander, and Nyberg [10]. The second is finding a way to extend the DLCT methodology so that it will cover more rounds at the boundary between E_0 and E_1. The third direction is studying properties of the DLCT, in a similar way to the way the properties of the BCT were recently studied by Boura and Canteaut [12]. The fourth direction is finding other applications of the DLCT. We believe that the DLCT is a useful generic tool, and so, we expect more applications to be found.

Acknowledgements

The authors thank Ran Proshan from the BIU Center for Research in Applied Cryptography and Cyber Security for his implementation skills and efforts.

The research was partially supported by European Research Council under the ERC starting grant agreement n. 757731 (LightCrypt) and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. Orr Dunkelman was supported in part by the Israel Ministry of Science and Technology, the Center for Cyber, Law, and Policy in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office and by the Israeli Science Foundation through grant No. 880/18.

References

1. Ross Anderson, Eli Biham, and Lars R. Knudsen. Serpent: A proposal for the Advanced Encryption Standard, NIST AES Proposal, 1998.
2. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *Advances in Cryptology - proceedings of CRYPTO 2016, Lecture Notes in Computer Science 9815*, pages 123–153. Springer, 2016.
3. Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In *Advances in Cryptology - proceedings of EUROCRYPT 2001, Lecture Notes in Computer Science 2045*, pages 340–357. Sprginer, 2001.
4. Eli Biham, Orr Dunkelman, and Nathan Keller. Enhancing Differential-Linear Cryptanalysis. In *Advances in Cryptology - proceedings of ASIACRYPT 2002, Lecture Notes in Computer Science 2501*, pages 254–266. Springer, 2002.

5. Eli Biham, Orr Dunkelman, and Nathan Keller. Differential-linear cryptanalysis of serpent. In *proceedings of Fast Software Encryption, FSE 2003, Lecture Notes in Computer Science 2887*, pages 9–21. Springer, 2003.
6. Eli Biham, Orr Dunkelman, and Nathan Keller. A Related-Key Rectangle Attack on the Full KASUMI. In *Advances in Cryptology - proceedings of ASIACRYPT 2005, Lecture Notes in Computer Science 3788*, pages 443–461. Springer, 2005.
7. Eli Biham, Orr Dunkelman, and Nathan Keller. New Combined Attacks on Block Ciphers. In *proceedings of Fast Software Encryption, FSE 2005, Lecture Notes in Computer Science 3557*, pages 126–144. Springer, 2005.
8. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.
9. Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. Cryptanalysis of SAFER++. In *Advances in Cryptology - proceedings of CRYPTO 2003, Lecture Notes in Computer Science 2729*, pages 195–211. Springer, 2003.
10. Céline Blondeau, Gregor Leander, and Kaisa Nyberg. Differential-Linear Cryptanalysis Revisited. *J. Cryptology*, 30(3):859–888, 2017.
11. Céline Blondeau and Kaisa Nyberg. New Links between Differential and Linear Cryptanalysis. In *Advances in Cryptology - proceedings of EUROCRYPT 2013, Lecture Notes in Computer Science 7881*, pages 388–404. Springer, 2013.
12. Christina Boura and Anne Canteaut. On the Boomerang Uniformity of Cryptographic S-boxes. *IACR Trans. Symmetric Cryptol.*, 2018(3), 2018.
13. Florent Chabaud and Serge Vaudenay. Links Between Differential and Linear Cryptanalysis. In *Advances in Cryptology - proceedings of EUROCRYPT '94, Lecture Notes in Computer Science 950*, pages 356–365. Sprginer, 1994.
14. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang Connectivity Table: A New Cryptanalysis Tool. In *Advances in Cryptology - proceedings of EUROCRYPT 2018, Lecture Notes in Computer Science 10821*, pages 683–714. Springer, 2018.
15. Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improving the Time Complexity of Matsui's Linear Cryptanalysis. In *proceedings of Information Security and Cryptology - ICISC 2007, Lecture Notes in Computer Science 4817*, pages 77–88. Springer, 2007.
16. The CAESAR committee. Caesar: Competition for authenticated encryption: Security, applicability, and robustness, http://competitions.cr.yp.to/caesar.html, 2014.
17. Joan Daemen, René Govaerts, and Joos Vandewalle. Correlation Matrices. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 275–285. Springer, 1994.
18. C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer. Ascon. Submission to the CAESAR competition: http://ascon.iaik.tugraz.at, 2014.
19. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Cryptanalysis of Ascon. In *Proceedings of Topics in Cryptology - CT-RSA 2015, Lecture Notes in Computer Science 9048*, pages 371–387. Springer, 2015.
20. Orr Dunkelman, Sebastiaan Indesteege, and Nathan Keller. A Differential-Linear Attack on 12-Round Serpent. In *proceedings of Progress in Cryptology - INDOCRYPT 2008, Lecture Notes in Computer Science 5365*, pages 308–321. Springer, 2008.
21. Orr Dunkelman, Nathan Keller, and Adi Shamir. A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. *J. Cryptology*, 27(4):824–849, 2014.

22. Tao Huang, Ivan Tjuawinata, and Hongjun Wu. Differential-Linear Cryptanalysis of ICEPOLE. In *proceedings of Fast Software Encryption - FSE 2015, Lecture Notes in Computer Science 9054*, pages 243–263. Springer, 2015.
23. Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Deoxys v1.41. Submission to the CAESAR competition, 2016.
24. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In *proceedings of Fast Software Encryption, FSE 2000, Lecture Notes in Computer Science 1978*, pages 75–93. Springer, 2000.
25. Jongsung Kim, Seokhie Hong, Bart Preneel, Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Boomerang and Rectangle Attacks: Theory and Experimental Analysis. *IEEE Trans. Information Theory*, 58(7):4948–4966, 2012.
26. Xuejia Lai, James L. Massey, and Sean Murphy. Markov Ciphers and Differential Cryptanalysis. In *Advances in Cryptology - proceedings of EUROCRYPT '91, Lecture Notes in Computer Science 547*, pages 17–38. Springer, 1991.
27. Susan K. Langford and Martin E. Hellman. Differential-Linear Cryptanalysis. In *Advances in Cryptology - proceedings of CRYPTO '94, Lecture Notes in Computer Science 839*, pages 17–25. Springer, 1994.
28. Gaëtan Leurent. Improved Differential-Linear Cryptanalysis of 7-Round Chaskey with Partitioning. In *Advances in Cryptology - proceedings of EUROCRYPT 2016, Lecture Notes in Computer Science 9665*, pages 344–371. Springer, 2016.
29. Zhiqiang Liu, Dawu Gu, Jing Zhang, and Wei Li. Differential-Multiple Linear Cryptanalysis. In *proceedings of Information Security and Cryptology, Inscrypt 2009, Lecture Notes in Computer Science 6151*, pages 35–49. Springer, 2009.
30. Jiqiang Lu. A methodology for differential-linear cryptanalysis and its applications. *Des. Codes Cryptography*, 77(1):11–48, 2015.
31. Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology - proceedings of EUROCRYPT '93, Lecture Notes in Computer Science 765*, pages 386–397. Springer, 1993.
32. Pawel Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. ICEPOLE: High-Speed, Hardware-Oriented Authenticated Encryption. In *proceedings of Cryptographic Hardware and Embedded Systems - CHES 2014, Lecture Notes in Computer Science 8731*, pages 392–413. Springer, 2014.
33. Sean Murphy. The Return of the Cryptographic Boomerang. *IEEE Trans. Information Theory*, 57(4):2517–2521, 2011.
34. Ryan O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, 2014.
35. US National Bureau of Standards. Data Encryption Standard, Federal Information Processing Standards publications no. 46, 1977.
36. US National Institute of Standards and Technology. Advanced Encryption Standard, Federal Information Processing Standards publications no. 197, 2001.
37. Ali Aydin Selçuk. On Probability of Success in Linear and Differential Cryptanalysis. *J. Cryptology*, 21(1):131–147, 2008.
38. David A. Wagner. The Boomerang Attack. In *proceedings of Fast Software Encryption, Lecture Notes in Computer Science 1636*, pages 156–170. Springer, 1999.