Further Improving Differential-Linear Attacks: Applications to Chaskey and Serpent

Marek Broll[1], Federico Canale[1], Nicolas David[2], Antonio Florez-Gutierrez[2], Gregor Leander[1], María Naya-Plasencia[2], and Yosuke Todo[3]

[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Bochum, Germany,
{marek.broll,federico.canale,gregor.leander}@rub.de
[2] Inria, France,
{nicolas.david,antonio.florez-gutierrez,maria.naya_plasencia}@inria.fr
[3] NTT Secure Platform Laboratories, Tokyo, Japan,
yosuke.todo.xt@hco.ntt.co.jp

Abstract. Differential-linear attacks are a cryptanalysis family that has recently benefited from various technical improvements, mainly in the context of ARX constructions. In this paper we push further this refinement, proposing several new improvements. In particular, we develop a better understanding of the related correlations, improve upon the statistics by using the LLR, and finally use ideas from conditional differentials for finding many right pairs. We illustrate the usefulness of these ideas by presenting the first 7.5-round attack on Chaskey. Finally, we present a new competitive attack on 12 rounds of Serpent, and as such the first cryptanalytic progress on Serpent in 10 years.

Keywords: cryptanalysis, differential-linear attack, partitions, LLR, Chaskey, Serpent, conditional-differential

1 Introduction

Symmetric ciphers are deployed in virtually any application using cryptography and are indeed used for encryption of the bulk of our private data. The security of symmetric primitives is evaluated as their resistance against known attacks, and the great success of symmetric cryptography is based on the community's effort to continually improve upon the best attacks by developing new general ideas and applying them to concrete ciphers to get concrete security levels and security margins. It is safe to say that the most studied attack families are differential and linear cryptanalysis and their variants.

Here, we are especially interested in their combination known as differential-linear cryptanalysis, initially introduced in [16]. The high-level idea is to split a cipher into two parts and to apply a differential attack to the first, and a linear attack to the second part. If successful, this results in a biased distribution when comparing (a linear combination of) output bits from pairs of ciphertexts stemming from plaintexts with a particular difference. Given a differential part with probability p and a linear part with correlation c this results in a correlation of roughly pc^2 and finally in an attack with (data) complexity of order $p^{-2}c^{-4}$.

Very recently we have seen a renewed attention to the classical differential-linear attack. There have been several improvements on estimating the involved biases using the differential-linear connectivity table [2] and general analysis of this tool and its relation to other criteria in Boolean functions [11]. Moreover, in a recent paper at CRYPTO 2020 [4], several technical improvements were made to the framework of differential-linear attacks with a focus on ARX ciphers.

ARX ciphers are of special interest in this context as (i) they are highly relevant as many recent designs follow this construction framework [19,5,3] and (ii) they are natural candidates for differential-linear attacks. Indeed, differential-linear attacks are especially useful if a good differential and linear property exist for a few rounds but degrades drastically with an increased number of rounds. This is indeed the case for many ARX designs.

The paper [4] proposed mainly three improvements upon previous work. First, the authors proposed to use that in many cases it is easy to construct many right pairs for a high probable differential from a single given right pair by simply flipping some of its bits. This idea is widely used in other contexts and in particular can be seen as a particular case of the more general framework on conditional differential attacks as introduced in [15]. In the optimal case, this allows to reduce the complexity of the attack by a factor of p^{-1}.

The second improvement is to use a more elaborate variant of the partitioning technique introduced in [17]. The basic idea is to select the linear approximation used depending on some specific bits of the state before the final key-addition. In a nutshell, the idea is that in some cases it is possible to deduce the value of the carry-bit at a certain bit-position which in turn allows to derive linear equations between the input and output of the modular addition. This can be used to significantly increase the correlation of the linear part (at the cost of not being able to use all the ciphertext originally available).

Finally, the third improvement, is the use of an FFT based technique on the key-recovery part that nicely allows to combine the value of the key-bits guessed during partitioning and making use of the fact that the remaining key-bits are reflected (only) in the sign of the involved correlations.

Putting those improvements together resulted in the best known attacks for ChaCha [5] and Chaskey [19]. However, several questions remained open. In particular, we realized that the strong link between the first improvement and conditional differential attacks might allow significantly more involved improvements. Therefore, it seems promising to check if further improvements are possible by using the full power of the conditional differential framework from [15]. Moreover, using a stronger statistical test would allow the use of additional partitioning and linear approximations with different correlations jointly.

Our Contribution. We tackle all the open ends mentioned above in this paper. We start by developing some useful formulas (in Section 3) to better understand the correlation of the differential-linear distinguishers when used with partition.

Table 1. Summary of results.

Target	Rounds	Source	Attack type	Time	Data	Memory
Chaskey	7	[4]	Differential-linear	$2^{51.21}$	$2^{40.21}$	practical
	7	This paper	Differential-linear	2^{50}	2^{39}	practical
	7.5	This paper	Differential-linear	2^{77}	2^{48}	practical
Serpent	12	[20,18][b]	Multidimensional linear	2^{242} MA	$2^{125.8}$	2^{108}
	11	[13][a]	Differential-linear	$2^{137.7}$	$2^{113.7}$	2^{94}
	12	This paper	Differential-linear	2^{241}	$2^{127.96}$	$2^{127.96}$

[a] In [13] a 12 round differential-linear attack was proposed, but as we will detail in its corresponding section, we found a flaw in this attack that has been acknowledged by the authors. [b] It should be noted that the original complexity estimates in [20] were too optimistic. We refer to [18] for the updated complexities and the fact that one of the two attacks from [20] is actually invalid.

This can be seen as the equivalent of the linear-hull theorems for classical linear approximations to the case of differential-linear attacks. Those formulas avoid unnecessary independence assumptions and are the basis for the approximation of the involved biases in our applications.

Next, in Section 4, we explain how and why using LLR statistics allows to further improve the attack framework and give an example of how this allows to use previously dismissed data in the partitioning step. We like to note that many previous LLR techniques are used when multiple linear approximations are applied to a single text like multiple/multidimensional linear attacks. On the contrary, our LLR is used to construct the best distinguisher in the case where correlations of every text are different.

The link to conditional differential attacks allows immediately for two improvements to the framework as discussed in Section 5. First, given that the attack from [4] is only successful if the basis pair is a right pair for the first part of the cipher, immediately allows to deduce additional key information. Second, a careful study of the involved conditions allows to identify ways to create more right pairs beyond the simple flipping of single bits as used before. This extends the ideas developed in [15] for stream ciphers to ARX ciphers (which turns out to be more challenging) and new applications to SPN ciphers. Moreover, in contrast to [8] we use conditions in the key-recovery part, and for the first time in the differential-linear cryptanalysis setting.

The impact of those improvements is demonstrated by resulting in the first attack on 7.5-round Chaskey. The details of those attacks are presented in Section 6, and these results are summarized in Table 1. For easy understanding, we first apply our techniques to the 7-round Chaskey. Considering a few bit improvements for the 7-round attack, at the first glance, our technique looks like a minor improvement. However, we emphasize these techniques are necessary to attack the 7.5-round Chaskey because correlation is lower. To distinguish the 7.5-round Chaskey with such low correlation, we need to collect more pairs, but only the technique shown in [4] is not sufficient to recover the secret key (see

Supplementary Materials F in detail). We remark that the original Chaskey was 8 rounds, which are still considered secure by the designers. Therefore, our additional half round for the attack against Chaskey significantly reduced the security margin.

Finally, in Section 7 we summarize our improvements for Serpent. In Table 1, we compare ours to the best published attacks until now, see [20].

Our new attack has a time complexity of 2^{241}, thus competitive with the best known attack. Moreover, it is the first differential-linear attack on Serpent to reach 12 rounds.

2 Preliminaries

Differential-linear cryptanalysis [16] starts by splitting a cipher E into two parts E_1 and E_2 such that $E = E_2 \circ E_1$. Supposing that we can apply a differential and linear distinguisher for E_1 and E_2 respectively, we get a distinguisher for the whole cipher E. We first recapture the general attack structure introduced in [4] and set the notation. Figure 7 shows the high-level description of the general structure.

Three main different techniques were introduced: 1) amplifying the probability of the differential part by carefully choosing the linear subspace $\mathcal{U}$, 2) choosing the linear masks dynamically dependent on ciphertexts, and 3) an FFT-based technique for improving the key recovery part when using partitions.

2.1 Differential Part

The first step of the attack is to collect many $(x, x \oplus \Delta_{in})$ called *right pairs* satisfying $E_1(x) \oplus E_1(x \oplus \Delta_{in}) = \Delta_m$. Let $\mathcal{X}$ be a set defined as

$$\mathcal{X} = \{x \in \mathbb{F}_2^n \mid E_1(x) \oplus E_1(x \oplus \Delta_{in}) = \Delta_m\}.$$

If pairs are used from $\mathcal{X}$, the probability of the differential part becomes 1 and the correlation of the differential-linear distinguisher also increases. To collect many such pairs efficiently, a linear subspace $\mathcal{U}$ is exploited in [4]. In the simple case, this subspace is chosen such that for any $x \in \mathcal{X}$ and any $u \in U$ it holds that $x \oplus u \in \mathcal{X}$, i.e. $x \oplus u$ is a right pair as well. However, this strict requirement restricts the size of $\mathcal{U}$ significantly and for the attack it is sufficient if this implication is true for most elements in $\mathcal{X}$. To capture this precisely, we define a subset $\mathcal{X}'$ of the set of right pairs $\mathcal{X}$ as

$$\mathcal{X}' = \{x \in \mathcal{X} \mid x \oplus u \in \mathcal{X} \text{ for all } u \in U\}.$$

Once we find a $x \in \mathcal{X}'$, we can find $2^{\dim U}$ right pairs for free. The differential probability denoted by p is defined as $p = |\mathcal{X}|/2^n$, and we can reduce the data complexity with the factor of p^{-1} when $\mathcal{X}' = \mathcal{X}$. Remark that this idea is already used before [4] in other contexts, e.g., the differential attack.

2.2 Linear Part

The idea introduced in [4] is to dynamically change linear masks dependent on ciphertexts, and approximations derived from these different linear masks are combined efficiently. In the following, without loss of generality, $c \in \mathbb{F}_2^n$ is divided into two parts $c_{\mathcal{P}} \in \mathbb{F}_2^{n_{\mathcal{P}}} = \mathcal{P}$ and $c_{\mathcal{R}} \in \mathbb{F}_2^{n-n_{\mathcal{P}}}$, and $c = c_{\mathcal{P}} \| c_{\mathcal{R}}$. Similarly, $k = k_{\mathcal{P}} \| k_{\mathcal{R}}$ and $y = y_{\mathcal{P}} \| y_R$. Then, for any $p_i \in \mathcal{P}$, the partition $\mathcal{T}_{p_i} \subset \mathbb{F}_2^n$ is defined as $\mathcal{T}_{p_i} = \{y \in \mathbb{F}_2^n \mid y_{\mathcal{P}} = p_i\}$. We first define two types of correlations:

- $q_{i,j}$: Correlation of the distinguisher in each partition $(y, \tilde{y}) \in (\mathcal{T}_{p_i}, \mathcal{T}_{p_j})$.

$$q_{i,j} \coloneqq \mathbf{Cor}_{\substack{x \in \mathcal{X} \text{ such that} \\ (y,\tilde{y}) \in \mathcal{T}_{p_i} \times \mathcal{T}_{p_j}}} \left[\langle \mathit{\Gamma}_{\text{out}}^{(p_i)}, z \rangle \oplus \langle \mathit{\Gamma}_{\text{out}}^{(p_j)}, \tilde{z} \rangle \right]$$

- ε_i: Correlation of the linear part for the function F.

$$\varepsilon_i \coloneqq \mathbf{Cor}_{y \in \mathcal{T}_{p_i}} \left[\langle \mathit{\Gamma}_{\text{out}}^{(p_i)}, z \rangle \oplus \langle \gamma^{(p_i)}, y \rangle \right]$$

We consider two assumptions to simplify the estimation of correlation.

Assumption 1 *The correlation $q_{i,j}$ is independent of $\mathcal{T}_{p_i} \times \mathcal{T}_{p_j}$. In other words, we assume*

$$q_{i,j} = \mathbf{Cor}_{x \in \mathcal{X}} \left[\langle \mathit{\Gamma}_{\text{out}}^{(p_i)}, z \rangle \oplus \langle \mathit{\Gamma}_{\text{out}}^{(p_j)}, \tilde{z} \rangle \right].$$

Correlations $q_{i,j}$ are estimated under Assumption 1. Without this assumption, correlations $q_{i,j}$ must be estimated for every partition, and it is practically difficult when the number of partitions is too large. Assuming Assumption 1 holds, the final correlation is estimated by using the following assumption.

Assumption 2 *The correlation $q_{i,j}$, ε_i, and ε_j are independent of each other. In other words, we assume*

$$q_{i,j}\varepsilon_i\varepsilon_j = \mathbf{Cor}_{\substack{x \in \mathcal{X} \text{ such that} \\ (y,\tilde{y}) \in \mathcal{T}_{p_i} \times \mathcal{T}_{p_j}}} \left[\langle \gamma^{(p_i)}, y \rangle \oplus \langle \gamma^{(p_j)}, \tilde{y} \rangle \right].$$

2.3 Key Recovery

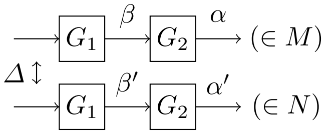
Which partition the text belongs to is determined by $y_{\mathcal{P}}$. If it belongs to the ith partition, the linear mask $\gamma^{(p_i)}$ is used. Since attackers can observe the ciphertext only, the key denoted by $k_{\mathcal{P}}$ is guessed to identify the partition and $y_{\mathcal{P}} = c_{\mathcal{P}} \oplus k_{\mathcal{P}}$. After guessing $k_{\mathcal{P}}$, assuming $y_{\mathcal{P}}$ and $\tilde{y}_{\mathcal{P}}$ belong to the i and jth partition, respectively, we get the following approximation finally.

$$\langle c \oplus k, \gamma^{(p_i)} \rangle \approx \langle \tilde{c} \oplus k, \gamma^{(p_j)} \rangle \Rightarrow \langle c, \gamma^{(p_i)} \rangle \oplus \langle \tilde{c}, \gamma^{(p_j)} \rangle \approx \langle k, \gamma^{(p_i)} \oplus \gamma^{(p_j)} \rangle$$

Since the left side is known, we get the parity of k with linear mask $\gamma^{(p_i)} \oplus \gamma^{(p_j)}$. For a linear subspace W defined by $W \coloneqq \mathrm{Span}\{\gamma^{(p_i)} \oplus \gamma^{(p_j)} | i, j \in \{1, \ldots, s\}\}$, approximations above involves $\dim W$ bits of information for k. Note that the Fast Walsh-Hadamard Transform (FWHT) is available in this step. Namely, we do not need to guess $\dim W$ bits for every pair of texts, and the time complexity is estimated as $2^{n_{\mathcal{P}}}(2N + \dim W \cdot 2^{\dim W})$, where $n_{\mathcal{P}}$ is the bit length of $k_{\mathcal{P}}$.

3 Differential-Linear Hull for Partitioning

In this section we want to clarify the Assumptions 1 and 2 used in [4]. For this, we will derive general formulas how to express the differential-linear correlation with restricted output of a function composed of two parts. Note that in the following we do not make any assumptions on independence of the parts.



We start by considering the unrestricted variant. Given two functions $G_1, G_2 : \mathbb{F}_2^n \to \mathbb{F}_2^n$, an input difference Δ and two output-masks α and α', we denote the differential-linear correlation (known as the auto-correlation in the theory of Boolean function) on $H := G_2 \circ G_1$ by

$$\mathrm{Aut}_H(\Delta, \alpha, \alpha') := 2^{-n} \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \alpha, H(x) \rangle + \langle \alpha', H(x+\Delta) \rangle}.$$

In differential-linear attacks we are normally interested in the case $\alpha = \alpha'$, but as we will see below, considering the more general case is important.

We are interested in how to compute the differential-linear correlation when considering intermediate masks β and β'. In a second step, the outputs will be restricted to M and N.

It is well known (see [12]), that this auto-correlation can be expressed as

$$\mathrm{Aut}_H(\Delta, \alpha, \alpha') = \sum_{u \in \mathbb{F}_2^n} \widehat{H}(u, \alpha) \widehat{H}(u, \alpha') (-1)^{\langle u, \Delta \rangle}. \tag{1}$$

where we denote by

$$\widehat{H}(u, \alpha) = 2^{-n} \sum_{x} (-1)^{\langle \alpha, H(x) \rangle + \langle u, x \rangle}$$

the correlation of the approximation of H with input and output masks u and α.

In the context of [4] and Assumptions 1 and 2, G_1 would correspond to E_m and G_2 to E_2, and we would experimentally estimate the auto-correlation and multiply this with the correlation of G_2 with input mask β and output mask α, i.e. we estimate

$$\mathrm{Aut}_H(\Delta, \alpha, \alpha) \approx \widehat{G}_2(\beta, \alpha)^2 \, \mathrm{Aut}_{G_1}(\Delta, \beta, \beta). \tag{2}$$

This is of course only an approximation and we now want to get an explicit expression of the hull effect, i.e. of all the parts we ignore in the above expression without making any assumptions.

For this, we use the fact (see [21]) that we can split the correlation of H into

$$\widehat{H}(u,\alpha) = \sum_{\beta} \widehat{G}_1(u,\beta)\widehat{G}_2(\beta,\alpha)$$

with an intermediate mask β, i.e. the linear hull. Putting this back in the definition of the auto-correlation, we get

Proposition 1 (Differential-Linear-Hull).

$$\mathrm{Aut}_{G_2\circ G_1}(\varDelta,\alpha,\alpha) = \sum_{\beta,\beta'} \widehat{G}_2(\beta,\alpha)\widehat{G}_2(\beta',\alpha)\,\mathrm{Aut}_{G_1}(\varDelta,\beta,\beta')$$

Proof. Let $H = G_2 \circ G_1$. Then

$$\begin{aligned}
\mathrm{Aut}_{G_2\circ G_1}(\varDelta,\alpha,\alpha) &= \sum_{u\in\mathbb{F}_2^n} \widehat{H}(u,\alpha)^2(-1)^{\langle u,\varDelta\rangle} \\
&= \sum_{u\in\mathbb{F}_2^n} \left(\sum_{\beta}\widehat{G}_1(u,\beta)\widehat{G}_2(\beta,\alpha)\right)^2 (-1)^{\langle u,\varDelta\rangle} \\
&= \sum_{\beta,\beta'} \widehat{G}_2(\beta,\alpha)\widehat{G}_2(\beta',\alpha) \sum_{u\in\mathbb{F}_2^n} \widehat{G}_1(u,\beta)\widehat{G}_1(u,\beta')(-1)^{\langle u,\varDelta\rangle} \\
&= \sum_{\beta,\beta'} \widehat{G}_2(\beta,\alpha)\widehat{G}_2(\beta',\alpha)\,\mathrm{Aut}_{G_1}(\varDelta,\beta,\beta').
\end{aligned}$$

□

So, as could be expected, the linear hull theorem has a natural extension as a differential-linear hull theorem and the approximation in Equation 2 corresponds to focusing on a single β while actually all pairs (β,β') have to be considered. It remains to see how restricting the input, i.e. partitioning, effects this expression.

3.1 Impact of Partitioning on the Correlation

We consider again a function $H : \mathbb{F}_2^n \to \mathbb{F}_2^n$, an input difference $\varDelta$, an output-mask α and not furthermore two sub-sets M, N of $\mathbb{F}_2^n$. We are interested in

$$\mathrm{Aut}_H^{(M,N)}(\varDelta,\alpha,\alpha) := \sum_{\substack{x\in\mathbb{F}_2^n \\ H(x)\in M,\, H(x+\varDelta)\in N}} (-1)^{\langle\alpha,H(x)\rangle+\langle\alpha',H(x+\varDelta)\rangle}.$$

One would hope that one can still use Equation 1 with minor modifications. That is basically by replacing the correlation of H by its restricted version.

To capture this, for a function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and a set $S \subseteq \mathbb{F}_2^n$ we denote by

$$\widehat{F|_S}(a,b) := \frac{1}{|S|}\sum_{x\in S}(-1)^{\langle b,F(x)\rangle+\langle a,x\rangle}$$

the correlation when *the input is restricted* to the set S. Later, it will actually be the output that is restricted, which will be handled by considering the inverse of the function (and swapping input and output-mask).

Using the Lemma 3 and 4 in Supplementary Material B we can state the main insight of this Section.

Proposition 2 (Differential-Linear-Hull with Restriction).

$$\mathrm{Aut}^{(N,M)}_{G_2 \circ G_1}(\Delta, \alpha, \alpha) = \sum_{\beta,\beta'} \widehat{G_2^{-1}|_N}(\alpha,\beta)\widehat{G_2^{-1}|_M}(\alpha,\beta')\,\mathrm{Aut}_{G_1}(\Delta,\beta,\beta')$$

The important observation is that we still can consider the unrestricted auto correlations of G_1 in this hull. Again, the work in [4] can be seen as focusing only one particular input mask. As we demonstrate in the application to Chaskey in Section 6, using Proposition 2 allows to improve upon the correlation by considering multiple intermediate masks.

4 LLR for distinguishing and key-recovery

According to the Neyman-Pearson lemma, the LLR test is the most powerful statistical test and as such has been used as a cryptanalysis tool (see e.g. [1,9]). We use it for the distinguisher as a replacement for the statistic used in [4].

Following the notation introduced in Section 2, we let

$$w_\ell = \langle c_l, \gamma^{(p_{i(\ell)})}\rangle \oplus \langle \tilde{c}_l, \gamma^{(p_{j(\ell)})}\rangle$$

for the ℓ^{th} pair $(c_\ell, \tilde{c}_\ell)$, and consider the probability $\pi_\ell = \Pr\left[w_\ell = 0\right]$.

Notice that in the applications, the computation of the correct estimate for the correspondent correlation $C_\ell = 2\pi_\ell - 1$ is the crucial point.

Let D_0 and D_1 be the distributions

$$D_0 : (\mathcal{B}(1,\pi_1),\ldots,\mathcal{B}(1,\pi_N)), \quad D_1 : (\mathcal{B}(1,1/2),\ldots,\mathcal{B}(1,1/2)).$$

where $\mathcal{B}(n,\pi_\ell)$ is the binomial distribution with n trials, each having probability π_ℓ of success, where $0 \leq \pi_\ell \leq 1$ are not necessarily distinct. If we consider samples from D_1, we denote the random variable of the LLR statistic $\mathcal{W}$. With samples from D_0, we denote the random variable $\mathcal{R}$ (see Supplementary Material C for the basics on LLR.) Next, we have to determine the means and variances of the LLR statistic under D_0 and D_1, i.e. of $\mathcal{R}$ and $\mathcal{W}$. As shown in Supplementary Material C we get

$$\mathbb{E}\left[\mathcal{W}\right] \approx -\frac{1}{2}\sum_{\ell=1}^{N} C_\ell^2 \text{ and } \mathbb{E}\left[\mathcal{R}\right] \approx \frac{1}{2}.\sum_{\ell=1}^{N} C_\ell^2,$$

where $C_\ell = 2\pi_\ell - 1$ corresponds to the correlation as before. For the variances we get

$$\mathrm{Var}\left[\mathcal{W}\right] \approx \mathrm{Var}\left[\mathcal{R}\right] \approx \sum_{\ell=1}^{N} C_\ell^2,$$

where we treat π_l as constant for simplicity.

4.1 Distinguishing between the Distributions

Our experiments indicate that the LLR statistic is normally distributed both in the random and in the real case. While this could potentially be treated theoretically, using e.g. some variant of the central limit theorem, we prefer to back this up by experiments in the applications. For now let $\mathcal{N}(\mu_0, \sigma_0^2)$ and $\mathcal{N}(\mu_1, \sigma_1^2)$ be the (assumed) normal distributions for the LLR statistics when the correct and wrong keys are guessed, respectively. The previous computation has shown that

$$\mu_0 = -\mu_1 = \frac{1}{2}\sum_{\ell=1}^{N} C_\ell^2 = \frac{N}{2}\bar{C}, \qquad \sigma_0^2 = \sigma_1^2 = \sum_{\ell=1}^{N} C_\ell^2 = N\bar{C},$$

where by $\bar{C}$ we denote the average of the squared correlation, i.e. $\bar{C} := \frac{1}{N}\sum_{\ell=1}^{N} C_\ell^2$.

In order to distinguish between the two distributions, we are interested in the gap between $\mu_0 - \mu_1$ and $\sigma_0 (= \sigma_1)$.

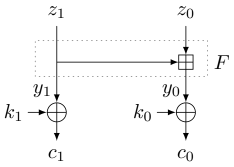
$$\frac{\mu_0 - \mu_1}{\sigma_1} = \frac{N\bar{C}}{\sqrt{N\bar{C}}} = \sqrt{N\bar{C}} = \sqrt{\sum_{\ell=1}^{N} C_\ell^2}. \tag{3}$$

Therefore, the larger $\sum_{\ell=1}^{N} C_\ell^2$, the bigger the gap is. This implies that in order to maximize this gap, no data should be discarded, that is there should be no partition with correlation zero. This is different from what happened in [4], where the same gap can approximately be represented as $\frac{\sum_{\ell=1}^{N} |C_\ell|}{\sqrt{N}} = \sqrt{N\bar{c}^2}$, where $\bar{c} = \frac{1}{N}\sum_{\ell=1}^{N} |C_\ell|$. In other words, this gap is proportional to the squared value of the average of the (absolute value of the) correlation ($\bar{c}$), while for the LLR statistic the same gap is proportional to the average of squared correlations ($\bar{C}$). We remark that the latter is always larger than the former, as expected when using the LLR.

4.2 New Partitioning strategy for ARX

The original idea of the partitioning technique [7] is to divide all the data into some partitions, and only using those partitions that can decrease the data complexity. The generalized partitioning technique in [17], which is also used in [4], also has the same feature. When the single modular addition is analyzed, all the data are divided into four partitions, and one out of the four is discarded.

Motivated by the LLR statistic and the feature of the FWHT-based key recovery, we propose a new partitioning strategy for ARX ciphers. Unlike the previous one, the LLR statistic shows that all partitions whose correlation is non-zero can contribute to decreasing the data complexity. Therefore, we want to use all partitions even if their correlations are relatively low. Recall that the time complexity of the FWHT-based key recovery is estimated as $2^{n_\mathcal{P}}(2N + \dim W \cdot 2^{\dim W})$. When $2N$ is significantly larger than $\dim W \cdot 2^{\dim W}$, the increase of



b_0b_1	$z_0[i]$		$z_0[i, i-1]$	
	mask γ	correlation ε	mask γ	correlation ε
00	11100	-1	11110	-1
01	11100	-1	11101	-2^{-1}
10	11010	-1	11000	1
11	11001	-2^{-1}	11000	1

Fig. 1. Partitions for a single modular addition.

$\dim W$ does not impact the whole of the time complexity. On the other hand, increasing $n_{\mathcal{P}}$ directly affects the whole of the time complexity. Therefore, we look for a new partitioning technique where the number of partitions is the same as the previous technique but linear masks γ involves more bits.

Single Modular Addition. Let us start with the most simple case of a single modular addition. To compute the parity $z_0[i]$ and $z_0[i] \oplus z_0[i-1]$ (shortly denoted by $z_0[i, i-1]$) from c_0 and c_1 (see Fig. 1), we represent each element of $\mathcal{P}$ as two-bit values b_0b_1, therefore dividing the whole set into four subsets

$$\mathcal{T}_{b_0b_1} = \{(y_1, y_0) \in (\mathbb{F}_2^n)^2 \mid b_0b_1 \cong s[i-1] \| s[i-2]\},$$

where $s = \bar{y}_1 \oplus y_0$. Note that these partition can be constructed by guessing two bits of key information, i.e., $(\bar{k}_1 \oplus k_0)[i-1]$ and $(\bar{k}_1 \oplus k_0)[i-2]$. Linear masks used in the previous partitioning technique involves 4 bits, i.e., $y_1[i]$, $y_0[i]$, $y_0[i-1]$, and $y_0[i-2]$. Our new partitioning technique additionally involves $y_0[i-3]$, and parities $z_0[i]$ and $z_0[i, i-1]$ are approximated to

$$\langle \gamma, y_1[i] \| y_0[i] \| y_0[i-1] \| y_0[i-2] \| y_0[i-3] \rangle,$$

where γ and the corresponding correlations are summarized in Fig. 1. As anticipated, the partition with $b_0b_1 \cong 11$ is available for $z_0[i]$ and the partition with $b_0b_1 \cong 01$ is available for $z_0[i, i-1]$, unlike the previous partitioning technique.

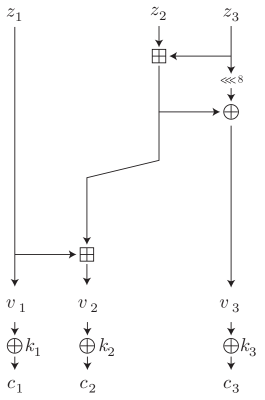
More Complicated Case. In a similar way, we can extend the technique for the case of two consecutive modular additions. A concrete example, which is used to attack the 7-round Chaskey, is shown in Fig. 2.

The goal is to compute the parity $z_2[11]$ and $z_2[11, 10]$ from c_1, c_2, and c_3 (see Fig. 2). We split the ciphertext into 2^5 partitions (this time indexed by five-bit values $b_0b_1b_2b_3b_4$ representing the generic element of $\mathcal{P}$) in the following way:

$$\mathcal{T}_{b_0b_1b_2b_3b_4} = \{(v_1, v_2, v_3) \in (\mathbb{F}_2^n)^3 \mid b_0b_1b_2b_3b_4 \cong (v_3[18] \oplus v_2[17] \oplus v_2[9]) \| \\ s[10] \| s[9] \| s[18] \| s[17]\},$$

where $s = \bar{v}_1 \oplus v_2$. In order for previously discarded partition to be available, our new partitioning technique additionally involves $v_2[8]$ and $v_2[16]$, and parities $z_2[11]$ and $z_2[11, 10]$ are approximated to

$$\langle \gamma, v_3[19] \| v_1[11] \| v_2[11] \| v_2[10] \| v_2[9] \| v_2[8] \| v_1[19] \| v_2[19] \| v_2[18] \| v_2[17] \| v_2[16] \rangle,$$



$b_0b_1b_2b_3b_4$	$z_2[11]$		$z_2[11,10]$	
	mask γ	correlation ε	mask γ	correlation ε
00000	11110111100	-2^{-2}	11110011100	1
00001	11111011100	-2^{-1}	11110011100	2^{-1}
00010	11111011010	-1	11111111010	-2^{-2}
00011	11111011010	-2^{-1}	11110011010	-2^{-1}
00100	11111011100	$2^{-0.263}$	11110011100	$2^{-0.263}$
00101	11111011111	$2^{-1.263}$	11110011100	$2^{-0.263}$
00110	11111011010	$-2^{-0.263}$	11110011010	$2^{-0.263}$
00111	11111011010	$-2^{-0.263}$	11110011001	$2^{-1.263}$
01000	11100011100	1	11100111100	2^{-2}
01001	11100011100	2^{-1}	11101011100	2^{-1}
01010	11101111010	-2^{-2}	11101011010	1
01011	11100011010	-2^{-1}	11101011010	2^{-1}
01100	11100011100	$2^{-0.263}$	11101011100	$-2^{-0.263}$
01101	11100011100	$2^{-0.263}$	11101011111	$-2^{-1.263}$
01110	11100011010	$2^{-0.263}$	11101011010	$2^{-0.263}$
01111	11100011001	$2^{-1.263}$	11101011010	$2^{-0.263}$
10000	11111011100	-1	11111111100	-2^{-2}
10001	11111011100	-2^{-1}	11110011100	2^{-1}
10010	11110111010	-2^{-2}	11110011010	1
10011	11111011010	2^{-1}	11110011010	2^{-1}
10100	11111011100	$-2^{-0.263}$	11110011100	$2^{-0.263}$
10101	11111011111	$-2^{-1.263}$	11110011100	$2^{-0.263}$
10110	11111011010	$2^{-0.263}$	11110011010	$2^{-0.263}$
10111	11111011010	$2^{-0.263}$	11110011001	$2^{-1.263}$
11000	11101111100	-2^{-2}	11101011100	1
11001	11100011100	2^{-1}	11101011100	2^{-1}
11010	11100011010	1	11100111010	2^{-2}
11011	11100011010	2^{-1}	11101011010	-2^{-1}
11100	11100011100	$2^{-0.263}$	11101011100	$2^{-0.263}$
11101	11100011100	$2^{-0.263}$	11101011111	$2^{-1.263}$
11110	11100011010	$2^{-0.263}$	11101011010	$-2^{-0.263}$
11111	11100011001	$2^{-1.263}$	11101011010	$-2^{-0.263}$

Fig. 2. Partition for two consecutive modular additions.

where γ is appropriately chosen following Fig. 2. We remark again that this new way of partitoning the ciphertexts allows us to find high-correlation masks for all the 32 partitions, up from the 24 used with the previous technique.

4.3 Using the LLR statistic for Key Recovery

Let $\tilde{y}_\ell$ be the intermediate state before the last key addition as in Section 2. After guessing a part k_P of the key, we know enough about these intermediate states to calculate $i(l), j(l)$ such that $(y_\ell, \tilde{y}_\ell) \in T_{p_{i(l)}} \times T_{p_{j(l)}}$. In order to ease notations, in the following we will simply write i and j, without making the dependency of the indices on ℓ explicit.

The original attack algorithm in [4] used

$$\alpha(i,j) = \sum_{\ell:(y_\ell,\tilde{y}_\ell)\in T_{p_i}\times T_{p_j}} (-1)^{\operatorname{sgn}(C_\ell)+w_\ell}$$

in order to build up counters $C(k_P, k_\ell)$, which allow identifying candidates for the correct key. We now formulate a replacement for α in terms of the LLR statistic which leads to an efficient implementation.

$$\mathrm{LLR} = N\ln(2) + \sum_{\ell=1}^{N} \ln(1-\pi_\ell) + \sum_{\ell=1}^{N} \ln\left(\frac{\pi_\ell}{1-\pi_\ell}\right) w_\ell$$

$$
\begin{aligned}
&= N\ln(2) + \sum_{\ell=1}^{N} \ln\left(\frac{1-C_\ell}{2}\right) + \sum_{\ell=1}^{N} \ln\left(\frac{1+C_\ell}{1-C_\ell}\right) \frac{1-(-1)^{w_\ell}}{2} \\
&= \sum_{\ell=1}^{N} \underbrace{\ln\left(\sqrt{1-C_\ell}\sqrt{1+C_\ell}\right)}_{0.5\ln\left(1-C_\ell^2\right)} + \frac{1}{2}\sum_{\ell=1}^{N} \ln\left(\frac{1-C_\ell}{1+C_\ell}\right)(-1)^{w_\ell}
\end{aligned}
$$

where $\pi_\ell = \Pr[w_\ell = 0] = \frac{1+C_\ell}{2}$. This leads to the new definition

$$
\alpha'(i,j) := \frac{1}{2}\left(\sum_{\ell:(c_\ell,\tilde{c}_\ell)\in T_{p_i}\times T_{p_j}} \ln\left(1-C_\ell^2\right) + \sum_{\ell:(c_\ell,\tilde{c}_\ell)\in T_{p_i}\times T_{p_j}} \ln\left(\frac{1-C_\ell}{1+C_\ell}\right)(-1)^{w_\ell} \right)
$$

The new counters are then defined analogously to the ones in [4]:

$$
\beta'(\gamma) := \sum_{(i,j):\gamma=\gamma^{(p_i)}+\gamma^{(p_j)}} \alpha'(i,j) \text{ and } C'(k_P,k_\ell) := \sum_{\gamma\in W} (-1)^{\langle\gamma,k_\ell\rangle}\beta'(\gamma).
$$

5 Improving Differential-Linear Attacks using Conditions

The new idea introduced in [4] for producing a set of pairs of data all verifying a certain differential path reflects a similar scenario as the one from conditional differential attacks. In a study presented in [15] in a context of NLFSR-based primitives, the basis of $\mathcal{U}$ from [4] are called *freebits* and involved more complex relations derived from the differential paths than just single bits. In Section 3.2 from [15] three types of conditions are presented: type zero that involve uniquely public bits (which is common in NLFSR initialization, but not in ARX or SPN networks); type one, involving both secret and public bits and these conditions define the free bits as the ones not affecting them; and type 2 which are conditions directly on the keybits. Using these definitions we can improve previous attacks in two ways: increasing for free the number of keybits recovered by the differential-linear attack thanks to type 2 conditions, and increasing the size of $\mathcal{U}$ by using the freebits as defined in [15].

We are going to briefly provide in this section some hints and general ideas on how to use this framework for improving the analysis in ARX constructions. In Supplementary Material E we provide a detailed example on how to determine additional keybits with type 2 conditions and on how to increase the number of freebits with evolved relations. The same ideas can be applied to SPN networks by considering Sboxes conditions. In Section 7.1 we develop this application on SPN networks following the example of the improved attack on Serpent.

Conditional differential framework for differential-linear attacks. Using the definitions from [15, §3.2] it is easy to see that we could improve some attacks (on ARX quite straight forward) exploiting the differential part in two main ways:

1. We can increase the size of $\mathcal{U}$ by using the non-freebits by exhausting the input values that keep the conditions of type-1 fixed to a certain value (as was done in the applications in that paper).
2. When trying a particular set of plaintexts to check if it is the one verifying the differential path, this will also provide some information on the value of some associated keybits or keybit relations (given directly by conditions of type-2 and indirectly by conditions of type-1). This means that we can suppose some known information on the key, that might be used to recover more bits and more importantly, could reduce the complexity of the key-search part in the final rounds.
3. Combination of both: guessing some keys, that might be useful for the linear part, that will allow to detect sampling bit relations that follow the path with probability 1.

Main ideas for exploiting the conditions on ARX. We present here some general ideas for exploiting conditions on ARX constructions. Though some might seem trivial, it is good to set them as rules to follow. We will provide technical details in Supplementary Material E about the example of application on Chaskey.

We can define some rules that apply when flipping the parity of differences. Instead of using only single non-active bit flipping for defining the freebits, we can study the effect of flipping the parity of the differences as additional sampling bits when possible. We can identify several relevant cases, we present here the 4 more representative ones: (i) If a pair of differences is going to be erased after a modular addition (which implies they have a different parity), changing the parity of one will need changing the parity of the other. (ii) If a bit-difference is staying at the same position (and not propagating further) after a modular transition, changing its parity won't affect the transition. (iii) If two active bits at position i will produce a difference after the modular addition at position $i+1$ (move the difference), flipping both active bits at the same time will change the parity of the output at $i+1$. (iv) If two words are added with a difference in position i and in positions i and $i+1$ respectively, and we want to absorb the differences after the modular additions, the carries of the previous positions won't affect the bits after position i. We can also change the parity of the three bits simultaneously, and the differences will still be absorbed, and the values will stay the same. Of course, all this might have an effect on further rounds, which will have, in turn, to be taken into account.

It is also useful to keep in mind that when we identify several input bits that only influence the differential transitions by a xor, swapping a pair number of these will not alter the verification of the path.

When dealing with carries, they might affect transitions with low probability. It is interesting to keep in mind that, when there is a sum of two zeros at position i, the value of all the bits at lower positions won't affect the carries at any higher positions. That might imply that a small guess (for instance 2 keybits for fixing two bits to zero) can generate many more bits for the sampling part with probability one if their only affected the differential path through these carries.

Table 2. Probability that adding one basis affects the output difference.

probability	basis	number of indices
$\gamma_j = 1$	v_2 : 16,17,18,19,20,22,23,24,25,30,31 v_3 : 16,17,18,19,20,22,23	18
$0.93 \leq \gamma_j < 1$	v_0 : 19,20,31 v_3 : 24,25,30 v_1 : 19,20	8
$\gamma_j = 1$	$v_0[8] \oplus v_1[8, 13] \oplus v_2[29]$ $v_2[21, 29] \oplus v_3[21]$ $v_0[18, 21, 30] \oplus v_1[21, 26, 30] \oplus v_2[3, 26] \oplus v_3[26, 27]$ $v_2[15] \oplus v_3[15]$	4

6 Improved Attacks against Chaskey

We improve differential-linear attacks against Chaskey (briefly described in supplementary material D) by using the LLR statistics and provide the first key-recovery attack reaching 7.5-round Chaskey.

To mount our improved 7-round and 7.5-round attacks, we also exploit the conditional differential techniques described in the previous section in order to introduce an improved basis of the linear subspace $\mathcal{U}$. Recall that the necessary condition for the attack to be effective is $2^{|\dim \mathcal{U}|} > \epsilon r^{-2} q^{-4}$. The new basis of $\mathcal{U}$ allows us to exploit lower correlations of the differential-linear distinguishers.

Another difficulty to mount our improved attacks is the estimation of the theoretical correlation, which is necessary for the LLR-based attack. While the theoretical correlation was already estimated in [4], a non-negligible gap was observed and the experimental correlation was much higher than the theoretical one. The previous estimation takes only the differential-linear trail with the highest correlation into account, but as we showed in the previous section, we should include the impact on the linear hull effect shown in Proposition 2.

6.1 New Basis of $\mathcal{U}$ for the Differential Part

Before we discuss the improved basis we have found using the conditional differential technique, we first recall the basis of $\mathcal{U}$ introduced in [4] (see the first two row blocks in Table 2). Here, the threshold of the probability is relaxed from 0.95 to 0.93, and $v_3[30]$ is newly added in the basis. The conditional differential techniques provide us additional four bases with probability 1, which cannot be found by Algorithm 1 [4] (see the third row block in Table 2). Linear subspace $\mathcal{U}$ whose dimension is 21 and 30 is finally used to attack 7-round and 7.5-round Chaskey, respectively. In particular, all, i.e., $18 + 8 + 4 = 30$, basis are used to attack 7.5-round Chaskey.

In part E from the Supplementary Material, we provide the details on how to obtain these relations. We use the conditional differential framework and Figure 9 in order to recover for free the value of some keybits and also to find additional bits of information for sampling and increasing the size of $\mathcal{U}$ from 18 as given in [4] (and involving exclusively one-bit relations) to 22, or 23 if one-bit relation on the key is known. The new proposed set of freebits (or relations with probability 1) is optimal and no more such relations exist.

Keybits that are obtained for free. If we find a set of inputs that verifies the differential path, we can directly deduce the following linear relations on the keybits, due to the conditions where differences are absorbed during the first modular additions (or the other way round, for each guess of these values, build sets of inputs that verify the 6 related conditions: $k_1[8] \oplus k_0[8]$, $k_1[21] \oplus k_0[21]$, $k_1[30] \oplus k_0[30]$ and $k_2[26] \oplus k_3[26]$, $k_2[21] \oplus k_3[21]$, $k_2[26] \oplus k_3[27]$.

Additional space for sampling. Compared with the linear subspace $\mathcal{U}$ shown in [4], the dimension of $\mathcal{U}$ increases by 4 by adding vectors listed in the third row block in Table 2 to the basis. In order to find these relations, we have used the rules presented in Section 5, and some more detailed explanations can be found in Supplementary Material 9 for the interested reader using figure 9. We summarize this in the following lemma:

Lemma 1. *There is a set $\mathcal{X}' \subseteq \mathbb{F}_2^{128}$ of size 2^{128-17} and a 22-dimensional linear subspace $\mathcal{U}$, such that for any element $x \in \mathcal{X}'$ and any $u \in \mathcal{U}$ it holds that $E_1(x \oplus u) \oplus E_1(x \oplus u \oplus \Delta_{in}) = \Delta_m$, where E_1 denotes 1.5 rounds of Chaskey*

Our improved 7-round attack uses this linear subspace.

One additional probability-one relation can be obtained if we flip the bit $v_2[27]$ and at the same time $v_2[29] = v_2[29] \oplus v_2[28] \oplus v_3[28]$. The issue with this one is that it depends on the relation of $k_2[28] \oplus k_3[28]$ (guessing this bit of information for instance would allow us to have an extra sampling bit) and it won't be used in the attack.

In addition to the probability one relations, we can consider larger linear subspace by adding relations with very high probabilities.

Lemma 2. *There is a set $\mathcal{X}' \subseteq \mathbb{F}_2^{128}$ whose size is about $2^{128-17.28}$ and a 30-dimension linear subspace $\mathcal{U}$, such that for any element $x \in \mathcal{X}'$ and any $u \in \mathcal{U}$ it holds that $E_1(x \oplus u) \oplus E_1(x \oplus u \oplus \Delta_{in}) = \Delta_m$ where E_1 denotes 1.5 rounds of Chaskey*

We can build a 30-dimension linear subspace such that all its elements verify simultaneously the differential with probability $2^{-17.28}$. For this, we consider the 22-dimension linear subspace of Lemma 1 and add to it the 7 vectors from [4] and $v_3[30]$. Our 7.5-round attack uses this linear subspace.

6.2 Improved 7-Round Attack

List of Differential-Linear Distinguishers. In [4], two differential-linear distinguishers with correlations $2^{-5.1}$ are shown.

$$\mathrm{Aut}_{E_m}(\Delta_m, ([], [], [20], []), ([], [], [20], [])) \approx 2^{-5.1},$$
$$\mathrm{Aut}_{E_m}(\Delta_m, ([], [], [20, 19], []), ([], [], [20, 19], [])) \approx 2^{-5.1},$$

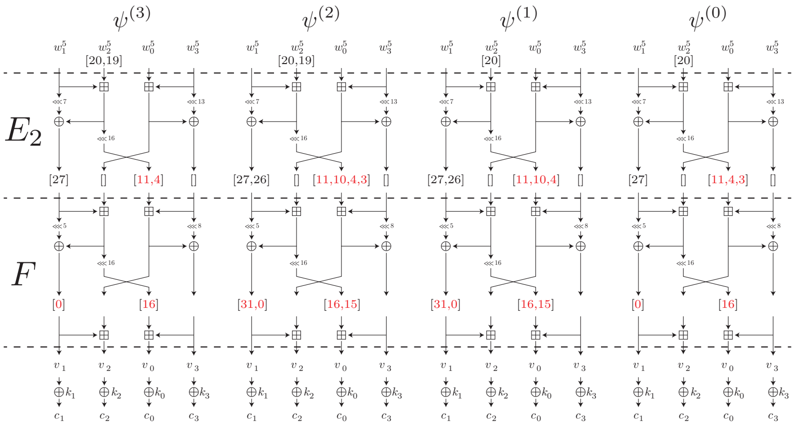


Fig. 3. Four 0.5-round linear trails for the 7-round attack.

Table 3. List of output linear masks after 6 rounds.

type	linear mask	δ	type	linear mask	δ
$\psi^{(0)}$	$\beta_0^{(0)} = ([27], [], [11,4,3], [])$	1	$\psi^{(2)}$	$\beta_0^{(2)} = ([27,26], [], [11,10,4,3], [])$	1
	$\beta_1^{(0)} = ([27], [], [11,4,2], [])$	1		$\beta_1^{(2)} = ([27,26], [], [11,10,4,2], [])$	1
	$\beta_2^{(0)} = ([27], [], [11,4,1], [])$	1		$\beta_2^{(2)} = ([27,26], [], [11,10,4,1], [])$	1
$\psi^{(1)}$	$\beta_0^{(1)} = ([27,26], [], [11,10,4], [])$	-1	$\psi^{(3)}$	$\beta_0^{(3)} = ([27], [], [11,4], [])$	1
	$\beta_1^{(1)} = ([27,26], [], [11,10,4,3,2], [])$	1		$\beta_1^{(3)} = ([27], [], [11,4,3,2], [])$	-1
	$\beta_2^{(1)} = ([27,26], [], [11,10,4,3,1], [])$	1		$\beta_2^{(3)} = ([27], [], [11,4,3,1], [])$	-1

where $\varDelta_m = ([31], [], [], [])$. By extending $([], [], [20], [])$ and $([], [], [20,19], [])$ by 0.5 rounds, respectively, we can get four linear masks (see Fig. 3). When both texts in pairs use either of $\psi^{(0)}$ or $\psi^{(1)}$, the correlation is $\pm 2^{-6.42}$. Moreover, when both texts in pairs use either of $\psi^{(2)}$ or $\psi^{(3)}$, the correlation is $\pm 2^{-6.42}$.

We have other linear masks whose correlation is relatively high but lower than $\pm 2^{-6.43}$. Table 3 summarizes 12 output masks. For any $(i,j) \in \{0,1\} \times \{0,1\}$ and $(i,j) \in \{2,3\} \times \{2,3\}$, correlations of the differential-linear distinguishers are estimated by the combination of two output masks as follows:

$$\begin{aligned} \mathrm{Aut}(\beta_0^{(i)}, \beta_0^{(j)}) &= \delta_0^{(i)} \cdot \delta_0^{(j)} \cdot 2^{-6.42}, & \mathrm{Aut}(\beta_0^{(i)}, \beta_1^{(j)}) &= \delta_0^{(i)} \cdot \delta_1^{(j)} \cdot 2^{-7.70}, \\ \mathrm{Aut}(\beta_0^{(i)}, \beta_2^{(j)}) &= \delta_0^{(i)} \cdot \delta_2^{(j)} \cdot 2^{-8.76}, & \mathrm{Aut}(\beta_1^{(i)}, \beta_1^{(j)}) &= \delta_1^{(i)} \cdot \delta_1^{(j)} \cdot 2^{-8.95}, \\ \mathrm{Aut}(\beta_1^{(i)}, \beta_2^{(j)}) &= \delta_1^{(i)} \cdot \delta_2^{(j)} \cdot 2^{-10.01}, & \mathrm{Aut}(\beta_2^{(i)}, \beta_2^{(j)}) &= \delta_2^{(i)} \cdot \delta_2^{(j)} \cdot 2^{-11.06}, \end{aligned}$$

where $\mathrm{Aut}(\beta^{(i)}, \beta^{(j)}) = \mathrm{Aut}_{E_2 \circ E_m}(\varDelta_m, \beta^{(i)}, \beta^{(j)})$ and $\delta_h^{(i)} \in \{1, -1\}$ is defined by the δ column in Table 3. Each correlation is estimated by using 1024 2^{25} pairs, i.e., 2^{35} pairs in total. Considering the lowest correlation is $2^{-11.06}$, estimation with 2^{35} pairs is reliable enough. These differential-linear distinguishers are finally used to estimate the theoretical correlation by considering the linear-hull effect.

Table 4. List of partition points for the attack against 7-round Chaskey.

ζ_1	choice : $(w_0^6[16], w_0^6[16, 15])$ $\mathcal{P}_1 \ni p_i \cong (s^R[15], s^R[14])$ linear : $v_3[16], v_0[16], v_0[15], v_0[14], v_0[13]$
ζ_2	choice : $(v_2^6[11], v_2^6[11, 10])$ $\mathcal{P}_2 \ni p_i \cong (v_3[18] \oplus v_2[9, 17], s^L[10], s^L[9], s^L[18], s^L[17])$ linear : $v_3[19], v_1[11], v_2[11], v_2[10], v_2[9], v_2[8], v_1[19], v_2[19], v_2[18], v_2[17], v_2[16]$
ζ_3	choice : $(v_2^6[4], v_2^6[4, 3])$ $\mathcal{P}_3 \ni p_i \cong (v_3[11] \oplus v_2[2, 10], s^L[3], s^L[2], s^L[11], s^L[10])$ linear : $v_3[12], v_1[4], v_2[4], v_2[3], v_2[2], v_2[1], v_1[12], v_2[12], v_2[11], v_2[10], v_2[9]$

Theoretical Correlations with Linear Hull Effect. To understand how to estimate the theoretical correlation, we provide an example. We observe a pair of ciphertexts $(c, \tilde{c})$ and guess key bits to identify the partition.

Table 4 summarizes the partition points for the 7-round attack. To identify the partition, we need to know

$$s^R[15], s^R[14], v_3[18] \oplus v_2[9, 17], s^L[10], s^L[9], s^L[18], s^L[17],$$
$$v_3[11] \oplus v_2[2, 10], s^L[3], s^L[2], s^L[11], (s^L[10]),$$

and 11-bit key guessing is enough, where $s^L = k_1 \oplus k_2$ and $s^R = k_0 \oplus k_3$. After we guess the 11-bit key, we assume that $\zeta_1 \ni p_i \cong (0, 0)$, $\zeta_2 \ni p_i \cong (0, 0, 0, 1, 0)$, and $\zeta_3 \ni p_i \cong (0, 0, 0, 1, 0)$ for c. We now consider the case that linear trail $\psi^{(3)}$ is used for both texts in a pair. When $\beta_0^{(3)}$ is used, available linear masks and corresponding correlation is computed as follows:

- To compute $w_0^6[16]$, $\gamma = 11100$ is used with correlation -1.
- To compute $v_2^6[11]$, $\gamma = 11111011010$ is used with correlation -1.
- To compute $v_2^6[4]$, $\gamma = 11111011010$ is used with correlation -1.

Note that the partition shown in Fig. 2 is directly available to evaluate $v_2^6[11]$. For other bits, e.g., $v_2^6[4]$, corresponding correlations must be reevaluated because the 11th bit and 4th bit provide slightly different correlations. Assuming all partition points are independent, correlation is

$$\widehat{F^{-1}|_{\mathcal{T}_{p_i}}}(\gamma^{p_i}, \beta_0^{(3)}) = -1 \times -1 \times -1 = -1$$

due to the piling-up lemma.

We also assume that $\zeta_1 \ni p_i \cong (0, 0)$, $\zeta_2 \ni p_i \cong (0, 0, 1, 0, 0)$, and $\zeta_3 \ni p_i \cong (0, 0, 0, 1, 0)$ for $\tilde{c}$. When $\beta_0^{(3)}$ is used, available linear masks and corresponding correlation is computed as follows:

- To compute $\tilde{w}_0^6[16]$, $\gamma = 11100$ is used with correlation -1.
- To compute $\tilde{v}_2^6[11]$, $\gamma = 11111011100$ is used with correlation $2^{-0.263}$.
- To compute $\tilde{v}_2^6[4]$, $\gamma = 11111011100$ is used with correlation -1.

Again, assuming all partition points are independent, correlation is

$$\widehat{F^{-1}|_{\mathcal{T}_{p_j}}}(\gamma^{p_j}, \beta_0^{(3)}) = -1 \times 2^{-0.263} \times -1 = 2^{-0.263}.$$

Thus, when $\beta_0^{(3)}$ and $\beta_0^{(3)}$ are used for c and $\tilde{c}$, respectively, the correlation (with one trail) is estimated as

$$\begin{aligned}&\widehat{F^{-1}|_{\mathcal{T}_{p_i}}}(\gamma^{p_i}, \beta_0^{(3)}) \times \widehat{F^{-1}|_{\mathcal{T}_{p_j}}}(\gamma^{p_j}, \beta_0^{(3)}) \times \mathrm{Aut}_{E_2 \circ E_m}(\varDelta_m, \beta_0^{(3)}, \beta_0^{(3)}) \\ &= -1 \times 2^{-0.263} \times (\delta_0^{(3)} \times \delta_0^{(3)} \times 2^{-6.42}) = -2^{-6.683}.\end{aligned}$$

We now take the linear-hull effect into account. Instead of $\beta_0^{(3)}$ for c, we use $\beta_1^{(3)}$ and compute the correlation when the same linear mask γ is used.

- To compute $v_2^6[4, 3, 2]$, $\gamma = 11111011010$ is used with correlation $2^{-0.677}$.

Therefore,

$$\widehat{F^{-1}|_{\mathcal{T}_{p_i}}}(\gamma^{p_i}, \beta_1^{(3)}) = -1 \times -1 \times 2^{-0.677} = 2^{-0.677}.$$

Therefore, when $\psi_1^{(3)}$ and $\psi_0^{(3)}$ are used for c and $\tilde{c}$, respectively, the correlation (with one trail) is estimated as

$$\begin{aligned}&\widehat{F^{-1}|_{\mathcal{T}_{p_i}}}(\gamma^{p_i}, \beta_1^{(3)}) \times \widehat{F^{-1}|_{\mathcal{T}_{p_j}}}(\gamma^{p_j}, \beta_0^{(3)}) \times \mathrm{Aut}_{E_2 \circ E_m}(\varDelta_m, \beta_1^{(3)}, \beta_0^{(3)}) \\ &2^{-0.677} \times 2^{-0.263} \times (\delta_1^{(3)} \times \delta_0^{(3)} \times 2^{-7.70}) = -2^{-8.64}.\end{aligned}$$

We estimate $3 \times 3 = 9$ correlations and sum up these correlations (considering sign). As a result, when $\psi_1^{(3)}$ and $\psi_0^{(3)}$ are used, the correlation increases to $-2^{-5.90893}$. We similarly estimate correlations when different linear trails are used, but as the fact, using $\psi_1^{(3)}$ and $\psi_0^{(3)}$ causes the highest correlation on this partition. Remark that once the indicator is given, the best linear mask and corresponding correlation are computed with offline. The complexity is about $2^{2k_{\mathcal{P}}}$, which is negligible to consider the time complexity for the whole of attacks.

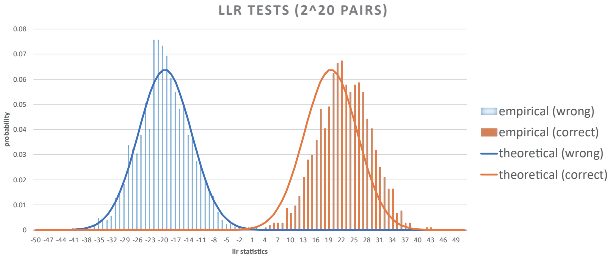
Experimental Reports. Correlation of each partition is high enough that we experimentally verify our attack procedure. In our experiments, we used the right pair and the correct key to observe the LLR statistics for the correct case. On the other hand, the right pair is not used for the wrong case.

The LLR statistics depends on the sum of the squared correlation $N\bar{C} = \sum_{\ell=1}^{N} c_\ell^2$. We estimated $\bar{C} \approx 2^{-14.711}$, and $N\bar{C} \approx 39.1$ when $N = 2^{20}$ pairs are used. The following shows the comparison of the LLR statistics, where the theoretical distribution is drawn by the normal distribution with mean $N\bar{C}/2$ (for a correct case) and $-N\bar{C}/2$ (for wrong case) and the standard deviation $\sqrt{N\bar{C}}$.

Table 5. List of output linear masks after 6.5 rounds.

type	linear mask	δ
$\psi^{(0)}$	$\beta_0^{(0)} = ([16, 15], [31, 0], [19, 12, 11, 4, 3], [19, 12])$	1
	$\beta_1^{(0)} = ([16, 15], [31, 0], [19, 12, 11, 4, 2], [19, 12])$	1
	$\beta_2^{(0)} = ([16, 15], [31, 0], [19, 12, 11, 4, 1], [19, 12])$	1
$\psi^{(1)}$	$\beta_0^{(1)} = ([16, 15], [31, 0], [19, 12, 4], [19, 12, 11])$	-1
	$\beta_1^{(1)} = ([16, 15], [31, 0], [19, 12, 4, 3, 2], [19, 12, 11])$	1
	$\beta_2^{(1)} = ([16, 15], [31, 0], [19, 12, 4, 3, 1], [19, 12, 11])$	1
$\psi^{(2)}$	$\beta_0^{(2)} = ([16], [0], [19, 18, 12, 4, 3], [19, 18, 12, 11])$	1
	$\beta_1^{(2)} = ([16], [0], [19, 18, 12, 4, 2], [19, 18, 12, 11])$	1
	$\beta_2^{(2)} = ([16], [0], [19, 18, 12, 4, 1], [19, 18, 12, 11])$	1
$\psi^{(3)}$	$\beta_0^{(3)} = ([16], [0], [19, 18, 12, 11, 4], [19, 18, 12])$	1
	$\beta_1^{(3)} = ([16], [0], [19, 18, 12, 11, 4, 3, 2], [19, 18, 12])$	-1
	$\beta_2^{(3)} = ([16], [0], [19, 18, 12, 11, 4, 3, 1], [19, 18, 12])$	-1

type	linear mask	δ
$\psi^{(4)}$	$\beta_0^{(4)} = ([16, 15], [31, 0], [19, 12, 11, 4], [19, 12])$	1
	$\beta_1^{(4)} = ([16, 15], [31, 0], [19, 12, 11, 4, 3, 2], [19, 12])$	-1
	$\beta_2^{(4)} = ([16, 15], [31, 0], [19, 12, 11, 4, 3, 1], [19, 12])$	-1
$\psi^{(5)}$	$\beta_0^{(5)} = ([16, 15], [31, 0], [19, 12, 4, 3], [19, 12, 11])$	1
	$\beta_1^{(5)} = ([16, 15], [31, 0], [19, 12, 4, 2], [19, 12, 11])$	1
	$\beta_2^{(5)} = ([16, 15], [31, 0], [19, 12, 4, 1], [19, 12, 11])$	1
$\psi^{(6)}$	$\beta_0^{(6)} = ([16], [0], [19, 18, 12, 11, 4, 3], [19, 18, 12])$	-1
	$\beta_1^{(6)} = ([16], [0], [19, 18, 12, 11, 4, 2], [19, 18, 12])$	-1
	$\beta_2^{(6)} = ([16], [0], [19, 18, 12, 11, 4, 1], [19, 18, 12])$	-1
$\psi^{(7)}$	$\beta_0^{(7)} = ([16], [0], [19, 18, 12, 4], [19, 18, 12, 11])$	1
	$\beta_1^{(7)} = ([16], [0], [19, 18, 12, 4, 3, 2], [19, 18, 12, 11])$	-1
	$\beta_2^{(7)} = ([16], [0], [19, 18, 12, 4, 3, 1], [19, 18, 12, 11])$	-1



By repeating our attack procedure 1024 times, two experimental histograms are drawn. A slight gap is observed between the theoretical distribution and experimental histogram in the correct case. Note that the experimental one is more biased than the theoretical estimation. We expect that the reason comes from the additional linear hull effect that we do not take into account.

We finally estimate the data and time complexities. To identify the partition, we need to guess the 11-bit secret key. We also enumerated elements of the linear subspace W and computed the basis by using Gaussian elimination. As a result, the dimension of W is 10. Because of Lemma 1, 2^{17} iterations are required to find the right pair. Thus, we need to remove $2^{11+10+17} = 2^{38}$ wrong cases. When 2^{21} pairs are used, $N\bar{C} \approx 78.2$. With a success probability of 90%, we can construct a 45.5-bit filter, which is enough to remove 2^{38} wrong cases. We finally estimate the time complexity by using the formula as follows.

$$T = p^{-1} \times 2^{n_{\mathcal{P}}} \times \left(2N + \dim W 2^{\dim W}\right) = 2^{17} \times 2^{11} \times \left(2 \times 2^{21} + 10 \times 2^{10}\right) \approx 2^{50.00}.$$

6.3 The 7.5-Round Attack

We further extend four 0.5-round linear trails to eight 1-round linear trails. For every linear trail, we have two different trails whose correlation is slightly low. Table 5 shows 24 such output masks. For any $(i, j) \in \{0, 1, 2, 3\} \times \{0, 1, 2, 3\}$ and $(i, j) \in \{4, 5, 6, 7\} \times \{4, 5, 6, 7\}$, correlations of the differential-linear distinguish-

Table 6. List of partition points for the attack against 7.5-round Chaskey.

ζ_1 $(v_2^7[23], v_0^7[23, 22])$	$\mathcal{P}_1 \ni p_i \cong (s^R[22], s^R[21], s^R[20], s^R[19], s^R[18])$ linear : $v_3[23], v_2[23], v_2[22], v_2[21], v_2[20], v_2[19], v_2[18], v_2[17]$
ζ_2 $(v_0^7[25], v_0^7[25, 24])$	$\mathcal{P}_2 \ni p_i \cong (s^L[24], s^L[23])$ linear : $v_1[25], v_0[25], v_0[24], v_0[23], v_0[22]$
ζ_3 $(w_0^6[16], v_0^7[16, 15])$	$\mathcal{P}_3 \ni p_i \cong (v_3[28] \oplus v_0[27, 14], s^L[15], s^L[14], s^L[28], s^L[27])$ linear : $v_3[29], v_1[16], v_0[16], v_0[15], v_0[14], v_0[13], v_1[29], v_0[29], v_0[28], v_0[27], v_0[26]$
ζ_4 $(w_2^6[19], v_0^7[19, 18])$	$\mathcal{P}_4 \ni p_i \cong (v_1[25] \oplus v_2[8, 1], s^R[2], s^R[1], s^R[9], s^R[8])$ linear : $v_1[26], v_3[3], v_2[3], v_2[2], v_2[1], v_3[10], v_2[10], v_2[9], v_2[8], v_2[7]$
ζ_5 $(w_2^6[12], v_0^7[12, 11])$	$\mathcal{P}_5 \ni p_i \cong (v_1[18] \oplus v_2[26, 1], s^R[27], s^R[26], s^R[2], s^R[1])$ linear : $v_1[19], v_3[28], v_2[28], v_2[27], v_2[26], v_2[25], v_3[3], v_2[3], v_2[2], v_2[1])$
ζ_6 $(w_2^6[4], v_0^7[4, 3])$	$\mathcal{P}_6 \ni p_i \cong (v_1[10] \oplus v_2[25, 18], s^R[19], s^R[18], s^R[26], s^R[25])$ linear : $v_1[11], v_3[20], v_2[20], v_2[19], v_2[18], v_2[17], v_3[27], v_2[27], v_2[26], v_2[25], v_2[24]$
ζ_7 $(v_0^7[31])$	$\mathcal{P}_7 \ni p_i \cong (s^L[30], s^L[29], s^L[28], s^L[27])$ linear : $v_1[31], v_0[31], v_0[30], v_0[29], v_0[28], v_0[27], v_0[26]$

ers are estimated by the combination of two output masks as follows:

$$\begin{aligned}
\mathrm{Aut}(\beta_0^{(i)}, \beta_0^{(j)}) &= \delta_0^{(i)} \cdot \delta_0^{(j)} \cdot 2^{-9.72}, & \mathrm{Aut}(\beta_0^{(i)}, \beta_1^{(j)}) &= \delta_0^{(i)} \cdot \delta_1^{(j)} \cdot 2^{-10.99}, \\
\mathrm{Aut}(\beta_0^{(i)}, \beta_2^{(j)}) &= \delta_0^{(i)} \cdot \delta_2^{(j)} \cdot 2^{-12.04}, & \mathrm{Aut}(\beta_1^{(i)}, \beta_1^{(j)}) &= \delta_1^{(i)} \cdot \delta_1^{(j)} \cdot 2^{-12.26}, \\
\mathrm{Aut}(\beta_1^{(i)}, \beta_2^{(j)}) &= \delta_1^{(i)} \cdot \delta_2^{(j)} \cdot 2^{-13.32}, & \mathrm{Aut}(\beta_2^{(i)}, \beta_2^{(j)}) &= \delta_2^{(i)} \cdot \delta_2^{(j)} \cdot 2^{-14.40},
\end{aligned}$$

where $\mathrm{Aut}(\beta^{(i)}, \beta^{(j)}) = \mathrm{Aut}_{E_2 \circ E_m}(\varDelta_m, \beta^{(i)}, \beta^{(j)})$ and $\delta_h^{(i)} \in \{1, -1\}$ is defined by the δ column in Table 5. These correlations are estimated by using 106384 2^{26} pairs, i.e., 2^{40} pairs in total. Considering the lowest correlation is $2^{-14.6}$, estimation with 2^{40} pairs is reliable enough. We use the same method as the attack against 7-round Chaskey to determine a linear mask and estimate the corresponding correlation.

Table 6 summarizes the partition points for the 7.5-round attack. To identify the partition, we need to know

$$\begin{aligned}
&s^R[22], s^R[21], s^R[20], s^R[19], s^R[18], s^L[24], s^L[23], \\
&v_3[28] \oplus v_0[27, 14], s^L[15], s^L[14], s^L[28], s^L[27], \\
&v_1[25] \oplus v_2[8, 1], s^R[2], s^R[1], s^R[9], s^R[8], \\
&v_1[18] \oplus v_2[26, 1], s^R[27], s^R[26], (s^R[2]), (s^R[1]), \\
&v_1[10] \oplus v_2[25, 18], (s^R[19]), (s^R[18]), (s^R[26]), s^R[25], \\
&s^L[30], s^L[29], (s^L[28]), (s^L[27])
\end{aligned}$$

and 24-bit key guessing is enough, where $s^L = k_0 \oplus k_1$ and $s^R = k_2 \oplus k_3$.

Experimental Reports. Each correlation is relatively lower than that for 7-round attack, but it is still possible to verify our attack procedure experimentally by using about 2^{28} pairs. Similarly to the 7-round attack, we used a right pair and correct key to observe the LLR statistics for the correct case, and a right pair is not used for the wrong case.

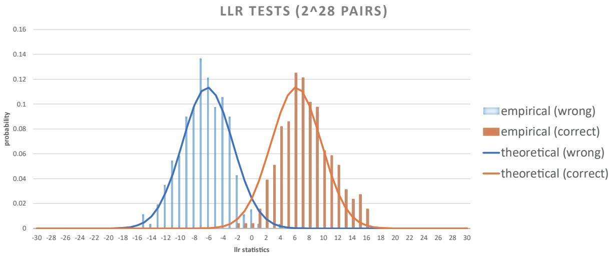


Fig. 4. Comparison with LLR statistics to attack 7.5-Round Chaskey.

We estimated $\bar{C} \approx 2^{-24.37}$, and $N\bar{C} \approx 12.38$ when $N = 2^{28}$ pairs are used. The above Figure shows the comparison of the LLR statistics, where the theoretical distribution is drawn by the normal distribution with mean $N\bar{C}/2$ (for a correct case) and $-N\bar{C}/2$ (for wrong case) and the standard deviation $\sqrt{N\bar{C}}$. By repeating our attack procedure 256 times, two experimental histograms are drawn. Similarly to the 7-round attack, a slight gap is observed between the theoretical distribution and experimental histogram in the correct case, and we again expect that the reason comes from the additional linear hull effect that we do not take into account.

We estimate the data and time complexities. To identify the partition, we need to guess the 25-bit secret key. We also enumerated elements of the linear subspace W and computed the basis by using Gaussian elimination. As a result, the dimension of W is 21. To find a right pair, we need $2^{17.28}$ iterations because of Lemma 2. Thus, we need to remove $2^{24+21+17.28} = 2^{62.28}$ wrong cases. Chaskey outputs at most 2^{48} data, the number of available pairs is at most $2^{48-17.28-1} = 2^{29.72}$. Then, $N\bar{C} \approx 40.78$. With success probability 90%, we can construct 22.5-bit filter, which is not enough to remove all wrong cases. Considering $2^{17.28}$ iterations to find a right pair, the performance to filter wrong keys decreases to 5.22 bits. We finally estimate the time complexity:

$$T = p^{-1} \cdot 2^{n_{\mathcal{P}}} \cdot \left(2N + \dim W 2^{\dim W}\right) = 2^{17.28} \cdot 2^{24} \cdot \left(2 \cdot 2^{30} + 21 \cdot 2^{21}\right) \approx 2^{72.28}$$

Using Multiple Linear Approximations Every Partition. Only filtering $2^{5.22}$ wrong keys is not always enough to attack 7.5-round Chaskey. To recover the unique key under the restriction of 2^{48} data, we use an extended attack, where multiple linear approximations are used for every partition. In 7.5-round attack, there are $2 \times 4 \times 4 = 32$ linear approximations and we choose only one approximation with highest correlation. However, why we do not use the other 31 approximations? The use of these approximations allows us to reduce the data complexity significantly. Of course, this is a little controversial technique because we are unlikely to be able to assume that each approximation is independent. Fortunately, since our attack can be verified experimentally, we simply implement our attack under this controversial assumption.

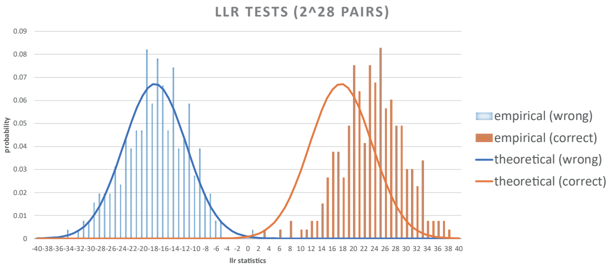


Fig. 5. Comparison with LLR statistics to attack 7.5-Round Chaskey when multiple linear masks are used every partition.

We estimated $\bar{C} \approx 2^{-22.86}$. When $N = 2^{28}$ pairs are used, $N\bar{C} \approx 35.26$, which increases from 12.38. Figure 5 shows the comparison of the LLR statistics, where the theoretical distribution is drawn by the normal distribution with mean $N\bar{C}/2$ (for a correct case) and $-N\bar{C}/2$ (for wrong case) and the standard deviation $\sqrt{N\bar{C}}$. By repeating our attack procedure 256 times, two experimental histograms are drawn. In spite of the controversial assumption, our theoretical estimation can simulate the experimental result nicely. Therefore, for the application to 7.5-round Chaskey, we conclude that the use of multiple linear approximations independently does not have any issue.

We estimate the data and time complexities. Again, since only $2^{29.72}$ pairs are available, $N\bar{C} \approx 116.16$. With a success probability of 90%, we can construct 69.6-bit filter, enough to remove $2^{62.28}$ wrong cases. The number of approximations, 32, is multiplied. We finally estimate the time complexity as follows

$$\begin{aligned} T &= p^{-1} \times 2^{n_{\mathcal{P}}} \times \left(2N \times 32 + \dim W 2^{\dim W}\right) \\ &= 2^{17.28} \times 2^{24} \times \left(2 \times 2^{29.72} \times 32 + 21 \times 2^{21}\right) \approx 2^{77.00}, \end{aligned}$$

7 Improved Attacks on Serpent

In this section we wanted to adapt some of the improved differential-linear techniques originally intended for ARX, like using conditions or using many pairs verifying the differential part, to propose attacks on SPN constructions. One of the most popular such constructions is Serpent, finalist of the AES competition, described in Supplementary Material H, and has been the target of many cryptanalysis papers. The previous best known attack on Serpent reaches 12 rounds and is a multidimensional linear attack presented in [20] and corrected in [18] (where the second variant was declared invalid): it has $2^{125.8}$ data complexity, 2^{242} MA time complexity and 2^{108} memory complexity. Additionally, some 12-round differential-linear attacks have been proposed, but during our work we noticed that these attacks, based on [13] are flawed (which has been confirmed by the authors). Any evident options for countering this would either increase the data complexity beyond the whole code book or the time complexity over the exhaustive search. We now show modifications which make it work, and we propose an optimized attack using partitions/conditions which provides, arguably, the best known attack on Serpent, as can be seen in Table 1.

Before describing the attacks in detail, we discuss some useful (differential and linear) conditional properties of the Serpent Sboxes. After these properties we will present a first modification of the flawed attack from [13] using some ideas which ensure the partial verification of the differential or linear parts, which provides an attack very close to exhaustive search. Next we show how using conditional differentials and the properties of the Sboxes we are able to improve the time complexity by a factor of around 2^{14}.

This constitutes the first improvement in the time complexity of attacks on Serpent in 10 years.

7.1 Conditions in (Small) Sboxes

Let us take the first Serpent Sbox S_0. If we denote the input by $x = (x_3, x_2, x_1, x_0)$ and the output by $y = (y_3, y_2, y_1, y_0)$ then the following set of relations always holds, which we will use in the attack below.

$$x_2 \oplus x_1 \oplus x_0 = 1 \Rightarrow y_0 = x_0 \oplus x_3 \oplus x_2 \oplus 1 \tag{4}$$

$$x_1 \oplus x_2 \oplus x_3 = 0 \Rightarrow y_1 = x_0 \oplus x_3 \oplus x_2 \oplus 1 \tag{5}$$

$$x_2 = 1 \Rightarrow y_2 = x_0 \oplus x_1 \oplus x_3 \tag{6}$$

$$x_3 = 0 \Rightarrow y_3 = x_0 \oplus x_1 \oplus x_2 \tag{7}$$

$$x_3 = 1 \Rightarrow y_3 = x_1 \oplus x_2 \oplus x_3 \tag{8}$$

Thanks to these relations, we can compute one bit of the output of the Sbox without guessing the full input. In order to determine y_3, we only need two bits of information of the input (we first query x_3 and then either $x_0 \oplus x_1 \oplus x_2$ or $x_1 \oplus x_2 \oplus x_3$). This decreases the amount of key material that needs to be guessed in an attack. To compute y_0, y_1 and y_2 we also need the following relations:

$$x_1 \oplus x_0 = 0 \text{ and } x_2 = 0 \Rightarrow y_0 = x_3 \oplus 1 \tag{9}$$

$$x_1 \oplus x_0 = 1 \text{ and } x_2 = 1 \Rightarrow y_0 = x_0 \oplus 1 \tag{10}$$

$$x_1 \oplus x_3 = 0 \text{ and } x_2 = 1 \Rightarrow y_1 = x_3 \oplus 1 \tag{11}$$

$$x_1 \oplus x_3 = 1 \text{ and } x_2 = 0 \Rightarrow y_1 = x_0 \oplus 1 \tag{12}$$

$$x_1 = 0 \text{ and } x_2 = 0 \Rightarrow y_2 = x_3 \tag{13}$$

$$x_1 = 1 \text{ and } x_2 = 0 \Rightarrow y_2 = x_0 \oplus 1 \tag{14}$$

With these relations we can determine any one bit of the output of the Sbox with either two or three parity bits of the input.

7.2 Attack on 12-R Serpent without conditions

We reuse the path from the flawed 12-round attack from [13], but consider the correct diffusion for the key recovery rounds and use the idea from [4] of generating many pairs which verify the differential part of the distinguisher.

Main Distinguisher. The path from the previous attack starts with round S_0, and the here *main distinguisher* will be the part starting after S_1 with Δ_{out} and until the input of S_2 eight rounds later with the mask Γ_{in}. We have $p = 2^{-6}$ and $q = 2^{-22}$, providing a total theoretical correlation of 2^{-50} for the differential-linear distinguisher. We expect it to be closer to $2^{-48.75}$ based on experiments on the transition between the differential and the linear trail.

We now focus on the first two rounds (Δ_{in} to Δ_{out}) and the last two rounds (Γ_{in} to Γ_{out}), which need to be studied in order to improve the attack:

Δ_{in}:	`XXXX`	`0XX0`	`00X0`	`0XX0`	`X0XX`	`X0XX`	`XXXX`	`00X0`	#0
After S_0,	`118A`	`0250`	`0080`	`0880`	`C012`	`4024`	`8618`	`0010`	#1
After LT	`5000`	`0000`	`0000`	`0C10`	`0800`	`9000`	`0000`	`0000`	#2
After S_1, 2^{-12}	`2000`	`0000`	`0000`	`01A0`	`0E00`	`4000`	`0000`	`0000`	#3
After LT, Δ_{out}	`0000`	`0000`	`0000`	`0000`	`0000`	`0000`	`4005`	`0000`	#4
8 Middle rounds:				$2^{-48.75}$					
Γ_{in}	`0000`	`A000`	`0000`	`0000`	`1000`	`0B00`	`00B0`	`000B`	#5
After S_2, , 2^{-5}	`0000`	`1000`	`0000`	`0000`	`5000`	`0100`	`0010`	`0001`	#6
After LT	`000B`	`0000`	`B000`	`0B00`	`00B0`	`200E`	`0000`	`0010`	#7
After S_3	`000X`	`0000`	`X000`	`0X00`	`00X0`	`X00X`	`0000`	`00X0`	#8
Γ_{out},									

If the active Sboxes after state #2 and before state #7 were paid probabilistically in full (that is, if we used a straightforward differential-linear distinguisher between those states), the data needed would be $O\left(2^{2\cdot(48.75+12+2\cdot 5)}\right) = O\left(2^{141.5}\right)$ which surpasses the size of the codebook.

Our first improvement is inspired by the idea proposed in [4] in the context of ARX, and we show for the first time how can it also be used in SPN constructions. The idea is to ensure some (not all) transitions in the first round of the distinguisher so that they do not contribute to the data. This can be done by guessing all the keybits related to these Sboxes, so that for each plaintext, we can deduce (partial) associated plaintexts so that the pair always verifies these transitions. Note that we are actually guessing more keybits than needed if we would only aim at computing the difference after S_0.

First rounds. Extending the differential at the top makes five Sboxes active (11,14,17,18 and 31) in the S_1 layer. These lead to 20 active Sboxes in the first round, which means that we require 80 bits of the first round subkey in order to guarantee the right difference enters the S_1 layer.

Input values of the active Sboxes in #2. If we ignore the first-round columns which are already active, computing the input values at S_1 for some precise columns will not activate all the Sboxes in the previous round. Among other things, a detailed drawing can be found in figure 6. The following table lists these independence relations:

Column at #2	"Free" columns at #1
31	$\{0, 10, 14, 23\}$
18	$\{2, 10, 19, 23, 24\}$
17	$\{0, 14, 22, 23, 27\}$

Improving the differential probability. We can guess additional keybits in the first and second round so that the two Sbox transitions of probability 2^{-3} from #2 to #3 (Sboxes 17 and 31) and the 2^{-2} transition of Sbox 18 are always satisfied.

If we fixed the input value of the three Sboxes to one which satisfies the transition indicated in the path, the amount of available data would be reduced by a factor of 2^8. In order to counter this, we can consider three undetermined differences `X` in state #2, which increases the number of "useful" pairs. We have to be careful as the target difference in #1 is not fixed now. In order to activate as few columns as possible, we only consider differences which do not include bit x_3 in column 17, which will reject half of the plaintexts, but also makes column 29 inactive in the previous round. In column 31 we only consider differences in which x_0 is not active, which is possible for 1/4 of the plaintexts. We thus keep 2^{-3} of the plaintexts, which means we can generate up to 2^{124} pairs from the whole codebook which verify these Sbox transitions.

As shown in Figure 6, in addition to the 80 keybits corresponding to the difference, computing the input values to these three Sboxes requires guessing all the other keybits except for Sboxes 23 and 29, which are inactive. This implies a key guess in the first round of 120 bits plus 12 bits in the second round corresponding to the three active Sboxes in #2 whose input values we want to control. In return the amount of pairs required by the attack is reduced by a factor of $2^{2\cdot(3+3+2)} = 2^{16}$.

Last Rounds. As before, we don't want to pay in full the correlation of the last S_2 transition from #5 to #6. Similarily, for each active Sbox (0,5,10,15,27), we determine which bits are required to compute the full output values.

Relationships between Sboxes in the last two rounds. We compute, for each active column in #6, which columns in state #7 do not influence the values of the column on the previous round. The results are condensed in the following table (the columns at #7 which are already active have been ignored):

Column at #6	"Free" columns at #7
0	$\{2,4,6,9,10,12,14,15,16,17,19,20,21,24,26,27,29,30,31\}$
5	$\{0,3,4,9,14,15,16,17,19,20,21,22,24,25,26,29,31\}$
10	$\{2,4,5,9,14,16,19,20,21,22,24,25,26,27,29,30,31\}$
15	$\{0,2,3,4,6,7,9,10,14,19,21,24,25,26,27,29,30,31\}$
27	$\{3,4,5,6,7,9,10,12,14,15,16,19,21,22,25,26,31\}$

If we determine the full output (instead of just the desired parity bit) of Sbox 0, there are 13 active Sboxes in the last round: we use a linear output mask spread between the inputs and the outputs of the S_2 layer. The number of recovery keybits at the last round is $4 \cdot 13 = 52$, together with 4 bits of the previous subkey. The remaining four Sboxes are part of the linear distinguisher.

The key recovery in the linear part can be performed efficiently by using the FFT technique detailed in [14], with a cost of $2 \cdot 52 \cdot 2^{2\cdot 52}$ for each guess of the key in the top and each of the 2^4 key guesses of the second-to-last round.

Cost of the operations with respect to a whole encryption. In order to properly evaluate the time complexity, we need to compare the cost of the operations in the attack against 12-round Serpent encryptions. For each plaintext, we perform a two round encryption at the beginning and a one round decryption at the end. However, the partial encryption can be performed for one plaintext x and the result can be reused for different values of columns 23 and 29. The filtering requires us to process 2^3 plaintexts to find one which satisfies the desired conditions. The cost of processing each plaintext is thus $(2^{-8} \cdot 2^3 \cdot 2/12 + 1/12) = 2^{-3.50}$ encryptions.

Complexity. Given the distinguisher's correlation of $2^{-48.75-4-2\cdot 4} = 2^{-60.75}$, and using the model from [10], we obtain that with $2^{123.96}$ pairs, ($2^{127.96}$ data complexity before sieving), an advantage of 15 is obtained with probability 0.1.

The time complexity of the attack is as follows:

$$\underbrace{2^{120} \cdot 2^{12}}_{\text{Top key guess}} \cdot \left(2^{123.5} \cdot 2^{-3.50} + \underbrace{O\left(2^4 \cdot (2 \cdot 52) \cdot 2^{2\cdot 52}\right)}_{\text{Bottom key guess}}\right) + 2^{256-15} \simeq 2^{252}$$

encryptions. We need $2^{127.96}$ registers for the data, as well as 2^{104} for the distillation tables of the FFT. The overall memory complexity is thus around $2^{127.96}$.

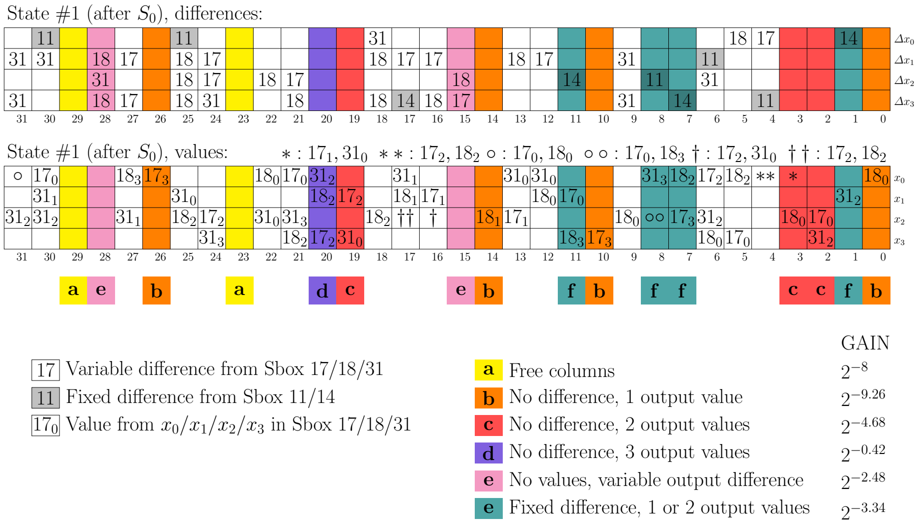
7.3 Improved 12-round Attack on Serpent thanks to Conditions

In this section, we apply some of this paper's new ideas to the previous 12-R attack which had a time complexity only barely better than exhaustive search, and improve it using the properties of the Sboxes. This allows us to propose the attack shown in Table 1.

A more detailed analysis of the key recovery at the top. We can reduce the time complexity by exploiting the fact that we might not need to guess all the keybits at the input of an Sbox: if we only need one or two bits of the output, the knowledge of a few input bits might be enough, as we showed earlier. We start by looking at the configuration of required differences and values at the output of S_0 (state #1), represented in Figure 6. A more detailed explanation of the figure than the legend can be found in s Supplementary material I.

Merging keybit guesses from rounds 0 and 1. As the relations for the Sboxes that we have introduced describe the desired output bits as linear combinations of the inputs, we'll sometimes be able to XOR them into a key guess for the next round. This can only be done once for each input bit of S_1.

Columns of type a (yellow). There are 2 columns (23,29) which are inactive and need no key guessing at all (the same as in the previous attack). These provide gain factor of 2^{-8} with respect to a full subkey guess.



17 Variable difference from Sbox 17/18/31

11 Fixed difference from Sbox 11/14

17_0 Value from $x_0/x_1/x_2/x_3$ in Sbox 17/18/31

a	Free columns	2^{-8}
b	No difference, 1 output value	$2^{-9.26}$
c	No difference, 2 output values	$2^{-4.68}$
d	No difference, 3 output values	$2^{-0.42}$
e	No values, variable output difference	$2^{-2.48}$
e	Fixed difference, 1 or 2 output values	$2^{-3.34}$

Fig. 6. Reducing the key guessing to determine parts of #1.

Columns of type b (orange). In columns 0, 10, 14 and 26 there is no difference and we only need the value of one output bit from each. Instead of guessing sixteen keybits, we can make use of the properties of S_0 described in Section 7.1.

- Columns 0 and 26: We only need y_0, which we can determine as follows:
  - If $x_0 \oplus x_1 \oplus x_2 = 0$ then:
    - If $x_0 \oplus x_1 = 0$ then $y_0 = x_3 \oplus 1$.
    - If $x_0 \oplus x_1 = 1$ then $y_0 = x_0 \oplus 1$.
  - If $x_0 \oplus x_1 \oplus x_2 = 1$ then $y_0 = x_0 \oplus x_2 \oplus x_3 \oplus 1$.

  For a given plaintext, we consider six possible guesses of the key instead of sixteen on each column, thus resulting in a 6/16 gain factor for each.
- Column 14: We only need y_2, and the gain factor is 6/16 again.
- Column 10: We only need y_3 for column 10. In this case, we only need to guess two bits instead of four. Hence the gain factor is 4/16.

Over the four columns, we obtain a gain factor of $\frac{6}{16} \cdot \frac{6}{16} \cdot \frac{6}{16} \cdot \frac{4}{16} \cdot \simeq 2^{-6.25}$. In addition, we can absorb the last bit guess for some of the columns into into the next round. As bit x_3 of column 17 in S_1 is associated to Sboxes 10 and 26 in S_0, we can only do this 3 times (x_3 in column 17 and x_0, x_1 in column 18 at #2). The total gain factor with respect to a full subkey guess is $2^{-6.25} \cdot 2^{-3} = 2^{-9.26}$.

Columns of type c (red). In columns 2, 3 and 19, there is no output difference and we require two output bits. We proceed as in the previous case, but considering both bits at the same time.

- Column 19: We need y_1 and y_3. Determining y_3 needs two bits, and determining y_1 requires an additional bit. We also absorb one keybit into the next round (bit x_2 from column 17). The gain for this column is thus 2^{-2}.

- Column 3: We need y_0 and y_2. We can obtain an average guessing cost of 10, which gives a gain factor of $2^{-0.68}$.
- Column 2: We need 3 input bits to determine the two desired output bits. Half of the time we absorb one keybit into x_0 of column 17 in #2 and into x_2 of column 31 the rest of the time. We thus obtain a gain factor of 2^{-2}.

Combining the gains from these Sboxes we obtain a total factor of $2^{-4.68}$.

Columns of type d (purple). In column 20 there is no output difference, but we want to determine three output bits. By carefully combining the relations, we can obtain an average guessing cost of 12 instead of 16, which provides a gain factor of $2^{-0.42}$. Key absorption might be too complex.

Columns of type e (violet). Columns 15 and 28 only require the output difference, but this difference is not fixed.

There are four possible differences in column 28: `0`, `4`, `A` or `E` (as both Δx_1 and Δx_3 come from difference Δx_2 in column 18), which appear with probability 1/4 (this can be deduced from the DDT of S_1 and the output differences for columns 18 and 31). We first guess the parts of the first round subkey which determine the values at the inputs of columns 18 and 31, so that we know which difference we want at the output of column 28. This is only possible because we only need the output differences from these two columns, and no actual bit values.

As an example, we compute the cost when the desired output difference in column 28 is `4`. Looking at the DDT for S_0, there are only 4 possible input differences: `5`, `7`, `C` or `E`. We come up with the following guessing strategy:

- If $x_0 = 0$:
  - If $x_1 = 0$ then the good input difference is `C`.
  - If $x_1 = 1$:
    - If $x_3 = 0$ then the good input difference is `5`.
    - If $x_3 = 1$ then the good input difference is `7`.
- If $x_0 = 1$:
  - If $x_3 \oplus x_1 = 0$ then the good input difference is `E`.
  - If $x_3 \oplus x_1 = 1$:
    - If $x_3 = 0$ then the good input difference is `5`.
    - If $x_3 = 1$ then the good input difference is `7`.

This gives 6 possible guesses. The costs for the other non-zero differences are obtained in a similar way and are 2^3 for both. The overall cost becomes:

$$\frac{1}{4}6 + \frac{1}{4}2^3 + \frac{1}{4}2^3 + \frac{1}{4} \cdot 1 \simeq 2^{2.52}$$

instead of 2^4, and implies a gain factor of $2^{-1.48}$.

For column 15 we have two possible output differences: `4` and `C` (as input Δx_3 in column 18 is always 1). For difference `4` the average cost is 6, and for `C` the average cost is 10, which gives a gain factor of 2^{-1}.

Both gain factors multiply to $2^{-2.48}$.

Columns of type f (turquoise). In columns 1,7, 8 and 11 we have a fixed difference and we also need to determine some output values. Columns 1, 7 and 8 have average costs of 12 instead of 16. In column 7 we can also absorb one keybit into bit x_2 of column 18 in half of the cases, so it has an average cost of 18 (compared to 2^{4+1}). Column 11 can sometimes absorb one bit into x_0 in column 17 and sometimes into x_3 in column 18. Including this it has an average cost of $2^{4.32}$ (compared to 2^6). In total these columns generate a gain factor of $2^{-3.34}$.

On the last rounds. Even though similar properties exist for S_3 to the ones we have used for S_0, exploiting them when they depend on different guessed bits is much more complicated in the final rounds, as we are performing the last round key recovery with the Walsh transform. We leave as an open problem to find a way to use these properties in the last rounds.

Complexity. We now compute the time complexity of this improved version of the attack, as the data complexity is still $2^{127.96}$. If we consider all the gain factors we have accumulated, the equivalent number of encryptions is

$$2^{140-8-9.26-4.68-0.42-2.48-3.34} \cdot \left(\left(2^{123.5-3.5} + O\left(2^4 \cdot 104 \cdot 2^{104}\right)\right) + 2^{256-15} \simeq 2^{241}.$$

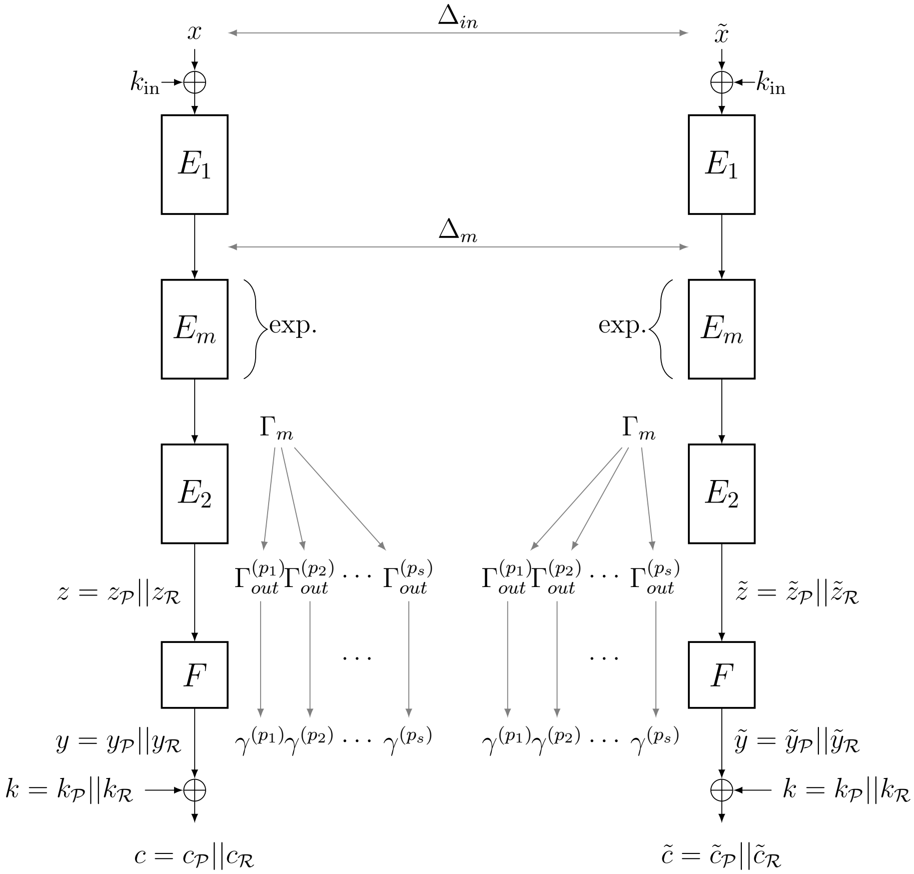
References

1. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004, Proceedings. LNCS, vol. 3329, pp. 432–450. Springer (2004)
2. Bar-On, A., Dunkelman, O., Keller, N., Weizman, A.: DLCT: A new tool for differential-linear cryptanalysis. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Proceedings, Part I. LNCS, vol. 11476, pp. 313–342. Springer (2019)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptol. ePrint Arch. **2013**, 404 (2013)
4. Beierle, C., Leander, G., Todo, Y.: Improved differential-linear attacks with applications to ARX ciphers. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Proceedings, Part III. LNCS, vol. 12172, pp. 329–358. Springer (2020)
5. Bernstein, D.J.: ChaCha, a variant of Salsa20 (2008)
6. Biham, E., Anderson, R.J., Knudsen, L.R.: Serpent: A new block cipher proposal. In: Vaudenay, S. (ed.) FSE '98, Proceedings. LNCS, vol. 1372, pp. 222–238. Springer (1998)
7. Biham, E., Carmeli, Y.: An improvement of linear cryptanalysis with addition operations with applications to FEAL-8X. In: Joux, A., Youssef, A.M. (eds.) SAC 2014, Revised Selected Papers. LNCS, vol. 8781, pp. 59–76. Springer (2014)
8. Biham, E., Perle, S.: Conditional linear cryptanalysis - cryptanalysis of DES with less than 242 complexity. IACR Trans. Symmetric Cryptol. **2018**(3), 215–264 (2018)
9. Blondeau, C., Gérard, B., Nyberg, K.: Multiple differential cryptanalysis using LLR and χ 2 statistics. In: Visconti, I., Prisco, R.D. (eds.) SCN 2012, Proceedings. LNCS, vol. 7485, pp. 343–360. Springer (2012)
10. Blondeau, C., Nyberg, K.: Improved parameter estimates for correlation and capacity deviates in linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2016**(2), 162–191 (2016)

11. Canteaut, A., Kölsch, L., Li, C., Li, C., Li, K., Qu, L., Wiemer, F.: On the differential-linear connectivity table of vectorial boolean functions. CoRR **abs/1908.07445** (2019)
12. Carlet, C.: Partially-bent functions. In: Brickell, E.F. (ed.) CRYPTO '92. LNCS, vol. 740, pp. 280–291. Springer (1992)
13. Dunkelman, O., Indesteege, S., Keller, N.: A differential-linear attack on 12-round serpent. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008, Proceedings. LNCS, vol. 5365, pp. 308–321. Springer (2008)
14. Flórez-Gutiérrez, A., Naya-Plasencia, M.: Improving key-recovery in linear attacks: Application to 28-round PRESENT. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Proceedings, Part I. LNCS, vol. 12105, pp. 221–249. Springer (2020)
15. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of nlfsr-based cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010, Proceedings. LNCS, vol. 6477, pp. 130–145. Springer (2010)
16. Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Desmedt, Y. (ed.) CRYPTO '94, Proceedings. LNCS, vol. 839, pp. 17–25. Springer (1994)
17. Leurent, G.: Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In: Fischlin, M., Coron, J. (eds.) EUROCRYPT 2016, Proceedings, Part I. LNCS, vol. 9665, pp. 344–371. Springer (2016)
18. McLaughlin, J., Clark, J.A.: Filtered nonlinear cryptanalysis of reduced-round serpent, and the wrong-key randomization hypothesis. In: Stam, M. (ed.) IMACC 2013, Proceedings. LNCS, vol. 8308, pp. 120–140. Springer (2013)
19. Mouha, N., Mennink, B., Herrewege, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In: Joux, A., Youssef, A.M. (eds.) SAC 2014, Revised Selected Papers. LNCS, vol. 8781, pp. 306–323. Springer (2014)
20. Nguyen, P.H., Wu, H., Wang, H.: Improving the algorithm 2 in multidimensional linear cryptanalysis. In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. Proceedings. LNCS, vol. 6812, pp. 61–74. Springer (2011)
21. Nyberg, K.: Linear approximation of block ciphers. In: Santis, A.D. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer (1994)

Supplementary Material

A General Structure



x
Δ_{in}
$\tilde{x}$
k_{in}
k_{in}
E_1
E_1
Δ_m
E_m
exp.
exp.
E_m
E_2
Γ_m
Γ_m
E_2
$z = z_{\mathcal{P}}||z_{\mathcal{R}}$
$\Gamma_{out}^{(p_1)}\Gamma_{out}^{(p_2)} \cdots \Gamma_{out}^{(p_s)}$
$\Gamma_{out}^{(p_1)}\Gamma_{out}^{(p_2)} \cdots \Gamma_{out}^{(p_s)}$
$\tilde{z} = \tilde{z}_{\mathcal{P}}||\tilde{z}_{\mathcal{R}}$
F
$\cdots$
$\cdots$
F
$y = y_{\mathcal{P}}||y_{\mathcal{R}}$
$\gamma^{(p_1)}\gamma^{(p_2)} \cdots \gamma^{(p_s)}$
$\gamma^{(p_1)}\gamma^{(p_2)} \cdots \gamma^{(p_s)}$
$\tilde{y} = \tilde{y}_{\mathcal{P}}||\tilde{y}_{\mathcal{R}}$
$k = k_{\mathcal{P}}||k_{\mathcal{R}}$
$k = k_{\mathcal{P}}||k_{\mathcal{R}}$
$c = c_{\mathcal{P}}||c_{\mathcal{R}}$
$\tilde{c} = \tilde{c}_{\mathcal{P}}||\tilde{c}_{\mathcal{R}}$

Fig. 7. The general structure of the attack.

B Lemmas towards the Restricted Differential-Linear Hull

The following lemma are needed in the proof of Proposition 2.

Lemma 3.

$$\mathrm{Aut}_H^{(M,N)}(\varDelta, \alpha, \alpha) = \sum_{u \in \mathbb{F}_2^n} \widehat{H^{-1}|_M}(\alpha, u)\widehat{H^{-1}|_N}(\alpha', u)(-1)^{\langle u, \varDelta \rangle}.$$

Proof. We start by expanding the left side of the equation, denoted by L, as follows

$$\begin{aligned}
L &= \sum_{u \in \mathbb{F}_2^n} \widehat{H^{-1}|_U}(\alpha, u)\widehat{H^{-1}|_V}(\alpha', u)(-1)^{\langle u, \mathit{\Delta} \rangle} \\
&= \sum_{u \in \mathbb{F}_2^n} \left(\sum_{y \in M} (-1)^{\langle u, H^{-1}(y)\rangle + \langle \alpha, y \rangle} \right) \left(\sum_{y' \in N} (-1)^{\langle u, H^{-1}(y')\rangle + \langle \alpha', y' \rangle} \right) (-1)^{\langle u, \mathit{\Delta} \rangle} \\
&= \sum_{y \in M, y' \in N} (-1)^{\langle \alpha, y \rangle + \langle \alpha', y' \rangle} \sum_{u \in \mathbb{F}_2^n} (-1)^{\langle u, H^{-1}(y) + H^{-1}(y') + \mathit{\Delta} \rangle} \\
&= \sum_{\substack{y \in M, y' \in N \\ H^{-1}(y') = H^{-1}(y) + \mathit{\Delta}}} (-1)^{\langle \alpha, y \rangle + \langle \alpha', y' \rangle}
\end{aligned}$$

We now define x as $H^{-1}(y)$ and x' as $H^{-1}(y')$. Then $x + x' = \mathit{\Delta}$ and we get

$$L = \sum_{\substack{x \in \mathbb{F}_2^n \\ H(x) \in M, H(x + \mathit{\Delta}) \in N}} (-1)^{\langle \alpha, H(x) \rangle + \langle \alpha', H(x + \mathit{\Delta}) \rangle}$$

which is equal to $\mathrm{Aut}_H^{(M,N)}(\mathit{\Delta}, \alpha, \alpha)$ by definition. □

In order to get the restricted version of the differential-linear-hull, we first have to understand the linear hull of a restriction first.

Lemma 4. *Let $H = G_2 \circ G_2$ and $S \subseteq \mathbb{F}_2^n$ then*

$$\widehat{H|_S}(\alpha, \beta) = \sum_{\gamma} \widehat{G}_2(\gamma, \beta)\widehat{G_1|_S}(\alpha, \gamma)$$

Proof.

$$\begin{aligned}
\sum_{\gamma} \widehat{G}_2(\gamma, \beta)\widehat{G_1|_S}(\alpha, \gamma) &= \sum_{\gamma} \left(\sum_{y} (-1)^{\langle \beta, G_2(y) \rangle + \langle \gamma, y \rangle} \right) \left(\sum_{x \in S} (-1)^{\langle \gamma, G_1(x) \rangle + \langle \alpha, x \rangle} \right) \\
&= \sum_{y, x \in S} (-1)^{\langle \beta, G_2(y) \rangle + \langle \alpha, x \rangle} \sum_{\gamma} (-1)^{\langle \gamma, y + G_1(x) \rangle} \\
&= \sum_{x \in S, y = G_1(x)} (-1)^{\langle \beta, G_2(y) \rangle + \langle \alpha, x \rangle} \\
&= \sum_{x \in S} (-1)^{\langle \beta, G_2(G_1(x)) \rangle + \langle \alpha, x \rangle} \\
&= \widehat{H|_S}(\alpha, \beta)
\end{aligned}$$

C Analyzing the parameters of the LLR statistics

C.1 LLR Basics

Following the notation introduced in Section 2, we let

$$w_\ell = \langle c_l, \gamma^{(p_{i(\ell)})} \rangle \oplus \langle \tilde{c}_l, \gamma^{(p_{j(\ell)})} \rangle$$

for the ℓ^{th} pair $(c_\ell, \tilde{c}_\ell)$, and consider the probability

$$\pi_\ell = \Pr\left[w_\ell = 0\right].$$

Notice that in the applications, the computation of the correct estimate for the correspondent correlation $C_\ell = 2\pi_\ell - 1$ is the crucial point.

Let D_0 and D_1 be the distributions

$$\begin{aligned} D_0 &: \mathcal{B}(1, \pi_1), \ldots, \mathcal{B}(1, \pi_N), \\ D_1 &: \mathcal{B}(1, 1/2), \ldots, \mathcal{B}(1, 1/2). \end{aligned}$$

where $\mathcal{B}(n, \pi_\ell)$ is the binomial distribution with n trials, each having probability π_ℓ of success, where $0 \leq \pi_\ell \leq 1$ are not necessarily distinct. Let q_0 and q_1 be the probability that $w \coloneqq (w_1, \ldots, w_N)$ is the result of sampling from D_0 (i.e. from the real distribution) or D_1 (i.e. the random distribution) respectively. Thus

$$\begin{aligned} q_0 &= \Pr[w = X | X \sim D_0] = \prod_{\ell=1}^{N} \pi_\ell^{w_\ell} (1 - \pi_\ell)^{1 - w_\ell}, \\ q_1 &= \Pr[w = X | X \sim D_1] = \prod_{\ell=1}^{N} 2^{-1} = 2^{-N}. \end{aligned}$$

The LLR statistic is defined as $\ln(q_0/q_1)$. The likelihood of D_0 is larger than that of D_1 when $\ln(q_0/q_1) > 0$. Then the LLR statistic can be rewritten as

$$\begin{aligned} \ln\left(\frac{q_0}{q_1}\right) &= \ln\left(2^N \times \prod_{\ell=1}^{N} \pi_\ell^{w_\ell} (1 - \pi_\ell)^{1 - w_\ell}\right) \\ &= N \ln(2) + \sum_{\ell=1}^{N} w_\ell \ln(\pi_\ell) + \sum_{\ell=1}^{N} (1 - w_\ell) \ln(1 - \pi_\ell) \\ &= N \ln(2) + \sum_{\ell=1}^{N} \ln(1 - \pi_\ell) + \sum_{\ell=1}^{N} w_\ell \ln\left(\frac{\pi_\ell}{1 - \pi_\ell}\right). \end{aligned}$$

Note that the first term is constant, but the second and third term depend on the value of the guessed key bits, which affects the partition of the pairs. With a slight abuse of notation, we treat q_i as a random variable.

Estimating the Mean Assuming that w is chosen from D_1, we the average value of each w_ℓ is $1/2$.

$$\begin{aligned} \mathbb{E}\left[\mathcal{W}\right] = \mathbb{E}\left[\ln\left(\frac{q_0}{q_1}\right)\right] &= N \ln(2) + \sum_{\ell=1}^{N} \ln(1 - \pi_\ell) + \sum_{\ell=1}^{N} 2^{-1} \ln\left(\frac{\pi_\ell}{1 - \pi_\ell}\right) \\ &= N \ln(2) + \sum_{\ell=1}^{N} \ln(1 - \pi_\ell) + \sum_{\ell=1}^{N} 2^{-1} \ln(\pi_\ell) - \sum_{\ell=1}^{N} 2^{-1} \ln(1 - \pi_\ell) \\ &= N \ln(2) + \sum_{\ell=1}^{N} 2^{-1} \ln(\pi_\ell (1 - \pi_\ell)). \end{aligned}$$

Let $c_\ell = 2\pi_\ell - 1$, and $\pi_\ell(1-\pi_\ell) = \frac{1+c_\ell}{2} \times \frac{1-c_\ell}{2} = \frac{1-c_\ell^2}{4}$. Therefore,

$$\begin{aligned}\mathbb{E}\left[\mathcal{W}\right] &= N\ln(2) + \sum_{\ell=1}^{N} 2^{-1} \ln\left(\frac{1-c_\ell^2}{4}\right) \\ &= N\ln(2) + \sum_{\ell=1}^{N} 2^{-1}\ln(1-c_\ell^2) - \sum_{\ell=1}^{N}\ln(2) \\ &= \frac{1}{2}\sum_{\ell=1}^{N}\ln(1-c_\ell^2)\end{aligned}$$

Using the Taylor series of $\ln(1-c_\ell^2)$ we can approximate this expression with $-c_\ell^2$ when c_ℓ^2 is close to 0. Therefore

$$\mathbb{E}\left[\mathcal{W}\right] \approx -\frac{1}{2}\sum_{\ell=1}^{N} c_\ell^2$$

Next, assuming that w_ℓ is chosen from D_0, the average value of w_ℓ is $1/2 + c_\ell/2$. Therefore

$$\mathbb{E}\left[\mathcal{R}\right] = -\frac{1}{2}\sum_{\ell=1}^{N} c_\ell^2 + \sum_{\ell=1}^{N}\frac{c_\ell}{2}\ln\left(\frac{\pi_\ell}{1-\pi_\ell}\right)$$

Since $\pi_\ell/(1-\pi_\ell) = \frac{1+c_\ell}{1-c_\ell}$, we can rewrite the second term:

$$\begin{aligned}\sum_{\ell=1}^{N}\frac{c_\ell}{2}\ln\left(\frac{\pi_\ell}{1-\pi_\ell}\right) &= \sum_{\ell=1}^{N}\frac{c_\ell}{2}\ln\left(\frac{1+c_\ell}{1-c_\ell}\right) \\ &= \sum_{\ell=1}^{N}\frac{c_\ell}{2}\left(\ln(1+c_\ell) - \ln(1-c_\ell)\right) \approx \sum_{\ell=1}^{N} c_\ell^2\end{aligned}$$

where we have use again a Taylor approximation of $\ln(1+z)$ for the last step. We conclude that

$$\mathbb{E}\left[\mathcal{R}\right] \approx \frac{1}{2}\sum_{\ell=1}^{N} c_\ell^2$$

Estimating the Variance In order to compute the variance of $\ln\left(\frac{q_0}{q_1}\right)$, for simplicity we treat the term π_ℓ as a constant and we have experimentally verified that this is reasonable. With this in mind we can write

$$\operatorname{Var}\left[\ln\left(\frac{q_0}{q_1}\right)\right] \approx \operatorname{Var}\left[\sum_{\ell=1}^{N} w_\ell \ln\left(\frac{\pi_\ell}{1-\pi_\ell}\right)\right].$$

Assuming that w is chosen from D_1, we know that the variance of w_ℓ is $1/4$. As before, since $\pi_\ell/(1-\pi_\ell) = \frac{1+c_\ell}{1-c_\ell}$, we obtain

$$\sum_{\ell=1}^{N} \frac{1}{4}\left[\ln\left(\frac{1+c_\ell}{1-c_\ell}\right)\right]^2 = \sum_{\ell=1}^{N} \frac{1}{4}\left(\ln(1+c_\ell) - \ln(1-c_\ell)\right)^2$$

and using Taylor approximation:

$$\operatorname{Var}\left[\mathcal{W}\right] \approx \sum_{\ell=1}^{N} \frac{1}{4}\left(\ln(1+c_\ell) - \ln(1-c_\ell)\right)^2 \approx \sum_{\ell=1}^{N} c_\ell^2$$

Similarly, assuming that w is chosen from D_0, the variance of w_ℓ is $1/4 + c_\ell^2/4$. Therefore, we want to compute

$$\operatorname{Var}\left[\mathcal{R}\right] \approx \sum_{\ell=1}^{N} \frac{1}{4}\left(1-c_\ell^2\right)\left[\ln\left(\frac{1+c_\ell}{1-c_\ell}\right)\right]^2$$

which is approximated with Taylor to

$$\sum_{\ell=1}^{N} \frac{1}{4}\left(1-c_\ell^2\right)\left[\ln\left(\frac{1+c_\ell}{1-c_\ell}\right)\right]^2 \approx \sum_{\ell=1}^{N} c_\ell^2 - c_\ell^4.$$

Therefore, the variance in both cases is approximately

$$\sum_{\ell=1}^{N} c_\ell^2.$$

D Chaskey specification

Chaskey was first introduced in [19] by Mouha and al. It is a lightweight MAC algorithm that uses a ARX permutation in a Even-Mansour construction. The underlying permutation operates on 128 bits split into four 32-bits words v_0, v_1, v_2, v_3, it consists in 12 applications of a round function described in figure 8. The designers claim security up to 2^{80} computations as long as the data is limited to 2^{48} blocks.

E Exploiting the conditions for finding Chaskey Relations

In figure 9 we have depicted the relations and the influence of the input bits on the conditions of the differential path. The bits that stay white (and have no pink colour beneath, coming from the carries of the furthest additions) are the bits that do not affect the differential transitions.

It is easy to see how the bits provided in [4] as available for sampling with probability one are the only white ones, and therefore not needed for the differential conditions: [31,30,25,24,23,22,20,19,18,17,16] from v2 and [23,22,20,19,18,17,16]

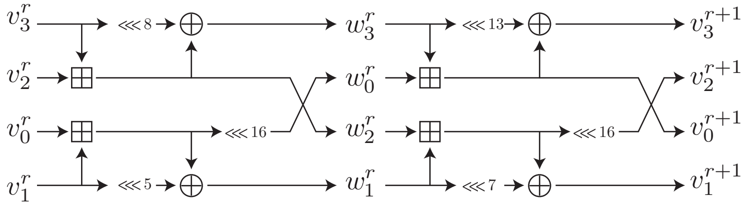


Fig. 8. Chaskey round function

from v3. The differences are represented in grey. Dependencies in colours. A 'g' in the position of a difference means that this difference will go away(be absorbed) after the next addition. An 's'means that the difference stays where it is, while 'm'means that it moves one position to the left. The colour of the bits with difference in each transition will be applied to all the bits that might affect this transition. Carries are not directly applied to the involved bits but to the upper row to report the difference this implies.

Please note that for instance bits 28 and 27 from v2 cannot be included as the carry of the position 29 is needed by the orange bit relations, i.e. the differences after one round at position 29 of v2 and v3, but as said in Section 5, the bits of previous positions to 26 and 27 won't affect this orange carry anymore due to the particular configuration of 26 and 27. The bits provided in [4] that are neutral with very high probability are 20 and 19 from v1 and 31, 20 and 19 from v0 and 25 and 24 of v3.

Lets now see how can we use the conditional differential ideas and Figure 9 in order to recover for free the value of some keybits and also to find additional bits of information for sampling and increasing the size of $\mathcal{U}$ from 18 as given in [4] (and involving exclusively one bit relations) to 22, or 23 if one bit relation on the key is known.

Additional space for sampling Using Figure 9 we can try to exploit the conditions to find more evolved relations for increasing the size of $\mathcal{U}$. Let us provide an example: Let us imagine we flip the bit from $v_0[8]$. The corresponding difference, marked with a 'g', will have a change of parity. In order for this difference to be absorbed, we need to also flip the other blue difference that will be used for absorbing this one: $v_1[8]$. But if we flip this one, the value of the bit $v_1[13]$ after one round, that does not contain a difference, will be flipped also, as to produce it, $v_1[8]$ is shifted of 5 positions and xored with the sum of v_0 and v_1, that has a difference in position 13, marked with an 's': these differences cancel out in both cases, but the value of the resulting bit will change with the parity of $v_1[8]$, and the value of this pink will affect the final light-pink transition in the third round, as can be seen in the picture. In order to avoid this, we have to also flip $v_1[13]$: the state v1 after 1 round will be know the same, but the orange bit $v_2[29]$ after one round, that contains a difference and a 'g' will have the parity changed. In order to make the related transition be satisfied, we need to also change the

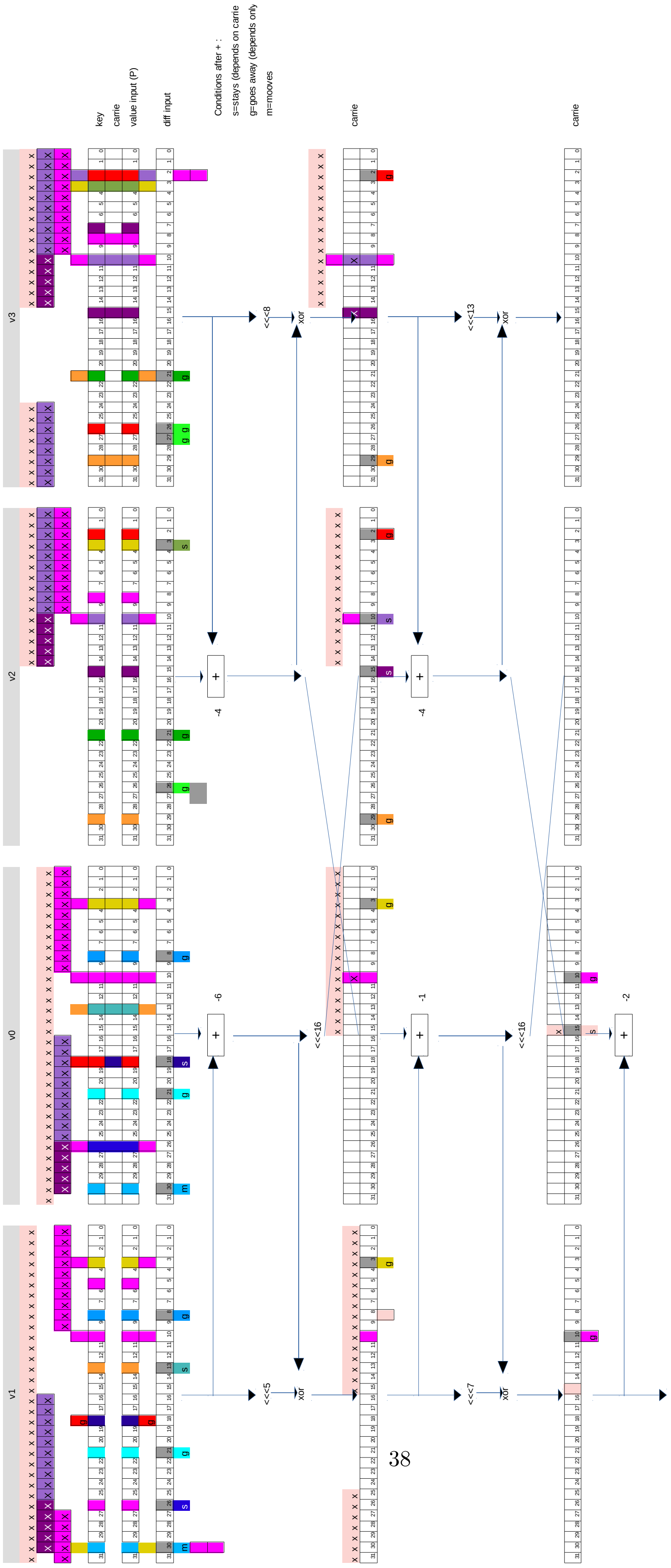


Fig. 9. Conditions on the differential transitions for the 3 first rounds of Chaskey

parity of the other orange bit with a ‘g’: we flip $v_2[29]$ from the first round, that doesn’t have a difference, but that will change the parity of $v_3[29]$ after the xor. This bit will not have any more influence in the remaining transitions, so we have found our close relation. In total we found four new probability-one relations by hand using this same technique. We have verified this relations as well as exhaustively searched all the ones with weight at most 3, verifying that there doesn’t exist any new ones.

F Impact on Conditional Differential and LLR Statistics to Attack the 7.5-Round Chaskey

In this paper, we mainly propose the two techniques: generating subspace $\mathcal{U}$ with larger dimension by using the conditional differential technique and the LLR statistics for the differential-linear attack with the partitioning technique. Here, we discuss the significant impact of these techniques to attack the 7.5-round Chaskey.

F.1 Impact of Conditional Differential

A cipher E is decomposed into three sub ciphers as $E = E_2 \circ E_m \circ E_1$, and p, r, and q denote

$$\begin{aligned}\mathbf{Pr}_{x \in \mathbb{F}_2^n}\left[E_1(x) \oplus E_1(x \oplus \varDelta_{\text{in}}) = \varDelta_m\right] &= p,\\ \mathbf{Cor}_{x \in \mathbb{F}_2^n}\left[\langle \varGamma_m, x\rangle \oplus \langle \varGamma_{\text{out}}, E_2(x)\rangle\right] &= q,\\ \mathbf{Cor}_{x \in \mathcal{S}}\left[\langle \varGamma_m, E_m(x)\rangle \oplus \langle \varGamma_m, E_m(x \oplus \varDelta_m)\rangle\right] &= r.\end{aligned}$$

Then, the correlation of the corresponding differential-linear distinguisher is pqr^2. The straightforward technique requires us to collect $\epsilon p^{-2}q^{-2}r^{-4}$ to distinguish the cipher, where ϵ is a constant.

In [4], the authors introduced a linear subspace $\mathcal{U}$, where for any $u \in \mathcal{U}$,

$$E_1(x \oplus u) \oplus E_1(x \oplus u \oplus \varDelta_{in}) = \varDelta_m$$

holds with high probability when $E_1(x) \oplus E_1(x \oplus \varDelta_{in}) = \varDelta_m$ holds. Then, they showed an attack procedure to reduce the data complexity from $\epsilon p^{-2}r^{-2}q^{-4}$ to $\epsilon p^{-1}r^{-2}q^{-4}$. However, this technique cannot be applied unless the condition $|\mathcal{U}| > \epsilon r^{-2}q^{-4}$ holds, and this necessary condition can be the difficulty when we attack the 7.5-round Chaskey.

The dimension of the linear subspace $\mathcal{U}$ in [4] was 25. This implies that $2^{25} > \epsilon r^{-2}q^{-4}$, and $rq^2 > 2^{-12.5}$ even if $\epsilon = 1$. We estimated correlation rq^2 experimentally. As a result, $rq^2 \approx 2^{-13}$ for the attack against the 7.5-round Chaskey. The size of the linear subspace does not hold the necessary condition.

Our conditional differential technique reveals a new linear subspace $\mathcal{U}$ with dimension 30. Since $2^{30} > \epsilon 2^{26}$ holds with $\epsilon < 16$. Unfortunately, $\epsilon < 16$ is not sufficient to recover the secret key. The technique shown in [4] imposes p^{-1}

iterations. In other words, unless at least p^{-1} wrong keys can be discarded with one statistical test, we cannot recover any bit of information. Supposing the normal distribution, the real roughly follows $\mathcal{N}(2^{30} \times 2^{-13}, 2^{30})$, but the ideal follows $\mathcal{N}(0, 2^{30})$. When we use a threshold $th = 2^{17}$, the probability such that the real is larger than the threshold is 50%. On the other hand, the probability such that the ideal is larger than the threshold is about $2^{-14.9}$. This implies that we cannot recover any bit of information because $p^{-1} \approx 2^{17}$.

F.2 Impact of LLR Statistics

The discussion above showed that only the conditional differential technique is not enough to attack the 7.5-round Chaskey. The reason is an insufficient statistical advantage. Therefore, we apply the LLR statistics to enhance the statistical advantage.

Let c_i be the correlation of the ith partition. Then, intuitively, the data complexity of the previous technique depends on $(\sum c_i)^2$ because the mean of the normal distribution is computed by the multiplication of the number of pairs and average correlation. On the other hand, when we use the LLR statistics, as shown in Sect. 4, the data complexity depends on $\sum c_i^2$. It is well-known that $\sum c_i^2 > (\sum c_i)^2$, which increases the advantage of the differential-linear distinguisher. The LLR statistics allow us to use more than a 17-bit filter (compared to a 14.9-bit filter) and to reduce the wrong-key space.

G Sbox Conditions

Here we show that linear conditions are not avoidable, if the Sbox is small compared to the number of (linear) restrictions in the input.

Lemma 5. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a permutation.If*

$$n - k + 2^k \leq 2n, \tag{15}$$

then there is a subspace U of dimension k and a vector $u \in \mathbb{F}_2^n$ such that $f|_{U+u}$ has maximal linearity.

Proof. Choose any subspace V of dimension $n - k$ and set $U = V^\perp$. Let L_V be the matrix with $n - k$ rows, whose row subspace is V and let M_U be a matrix, whose rows consist of all elements of U. Note that $x \in U$ iff $L_V x = 0$.

$f(M_U)$ is the matrix which is generated by applying f to every row of M_U. $M_U + u$ is defined analogously for any arbitrary vector.

Let u be an arbitrary vector. $[\alpha, \beta]$ is a mask from $U + u$ with correlation 1 if

$$\underbrace{\begin{bmatrix} L_V & 0 \\ M_U + u & f(M_U + u) \end{bmatrix}}_{W} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} L_V \alpha \\ c \\ \vdots \\ c \end{bmatrix} \tag{16}$$

for some $c \in \{0, 1\}$.

We are only interested in solutions where $\alpha \in U + u$. We set $u = 0$ for convenience. Then $L_V \alpha$ must be equal to zero.

W has $2n$ columns and $n-k+2^k$ rows. So if $n-k+2^k < 2n$ the system must be under-defined and all elements of the kernel of W are masks with correlation 1.

If $n - k + 2^k = 2n$ and the kernel contains only the trivial mask (**i.e. the square matrix W has full rank**), then $W^{-1}[0, 1, \dots, 1]^t$ is a non-trivial mask with correlation 1.

This lemma shows that for $0 \le k <= n \le 2$, $0 \le k < n = 3$ and $0 \le k \le 2$ for $n = 4$ there is always a restriction of dimension k which has maximal correlation:

n	k	$n-k+2^k$	$2n$
1	0	2	2
1	1	2	2
2	0	3	4
2	1	3	4
2	2	4	4
3	0	4	6
3	1	4	6
3	2	5	6
3	3	8	6
4	0	5	8
4	1	5	8
4	2	6	8
4	3	9	8
4	4	16	8

Of the highlighted cases in this table one can only hope to prove a comparable statement for $k = 3, n = 4$ (the others are obviously impossible).

Lemma 6. *For every permutation $f : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ there is a subspace U of dimension 3 and a vector $u \in \mathbb{F}_2^4$ such that $f|_{U+u}$ has linearity 8.*

Proof. Without loss of generality we assume that $f(0) = 0$ (via affine equivalence). We remove the corresponding line from W and the right-hand side of the equation for $c = 0$ (for $c = 1$ we immediately get a contradiction, so it is no use considering it). Again, we set $u = 0$. We choose V as above but in such a way that U contains certain vectors a, b, c, d.

W is then a square matrix and either has $rank < 8$, in which case we are done (the kernel of W contains at least one non-trivial solution), or it might have rank 8. Then the only solution is trivial. We show that the second case, rank 8, cannot occur if we choose a, b, c, d appropriately.

f cannot be APN because it is a permutation defined on 4 bits. Therefore there is a vanishing 2-flat for f:

$$a + b + c + d = 0 = f(a) + f(b) + f(c) + f(d). \qquad (17)$$

with a, b, c, d distinct. We distinguish two cases.

(Case 1) If none of these vectors is 0, then we can write

$$a = b + c + d, \; f(a) = f(b) + f(c) + f(d), \tag{18}$$

i.e. one further row of the remaining lines of W can be eliminated (i.e. annihilated by row transformations).

(Case 2) If one of the vectors is 0, assume wlog $d = 0$. Then

$$a = b + c, \; f(a) = f(b) + f(c) \tag{19}$$

since $d = 0 = f(0) = f(d)$ and we can still eliminate one further row of W.

H Description of Serpent Cipher

Serpent in a cipher introduced in [6] by Anderson, Biham and Knudsen. It is a SPN block cipher that has an internal state of 128 bits and keys can be up to 256 bits. To cipher, a round function is applied 32 times, it consists first in a step of key mixing, then in a layer of Sbox, finally in a linear transformation step. This cipher uses the following 8 Sboxes.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_0	3	8	F	1	A	6	5	B	E	D	4	2	7	0	9	C
S_1	F	C	2	7	9	0	5	A	1	B	E	8	6	D	3	4
S_2	8	6	7	9	3	C	A	F	D	1	E	4	0	B	5	2
S_3	0	F	B	8	C	9	6	3	D	1	2	4	A	7	5	E
S_4	1	F	8	3	C	0	B	6	2	5	4	A	9	E	7	D
S_5	F	5	2	B	4	A	9	C	0	3	E	8	D	6	7	1
S_6	7	2	C	5	8	4	6	B	E	9	1	F	D	3	A	0
S_7	1	D	F	0	E	8	2	B	7	4	C	A	9	3	5	6

To give a precise description of the round function, we will adopt the bit-sliced notations of [6]. We represent the internal state by four 32 bits vectors X_0, X_1, X_2, X_3.

- **Key mixing**. This step consist in Xoring a 128 bit subkey to the internal state. At round i, we will denote by $\hat{K}_i$ this subkey.
- **Sbox Layer**. This step will use a different Sbox depending on the round : at round number i, the Sbox $S_{i \mod 8}$. The Sbox Layer operation consists in the application of 32 copies the Sbox to the internal state. More precisely for each $i \in \{0, \cdots, 31\}$ we perform a Sbox transformation to the 4-bitstring composed by $X_0[i], X_1[i], X_2[i], X_3[i]$. We will denote by $\hat{S}_i$ the parallel applications of Sboxes done in round i.

- **Linear**. The linear operation denoted by LT, consists in applying the following operations:

$$\begin{aligned}
X_0 &\leftarrow X_0 <<< 13 \\
X_2 &\leftarrow X_2 <<< 3 \\
X_1 &\leftarrow X_1 \oplus X_0 \oplus X_2 \\
X_3 &\leftarrow X_3 \oplus X_2 \oplus (X_0 << 3) \\
X_1 &\leftarrow X_1 <<< 1 \\
X_3 &\leftarrow X_3 <<< 7 \\
X_0 &\leftarrow X_0 \oplus X_1 \oplus X_3 \\
X_2 &\leftarrow X_2 \oplus X_3 \oplus (X_1 << 7) \\
X_0 &\leftarrow X_0 <<< 5 \\
X_2 &\leftarrow X_2 <<< 22
\end{aligned}$$

  For the very last round the linear layer is omitted and an additional subkey is Xored to the internal state ($\hat{K}_{32}$).

If we design by $\hat{B}_i$ the 128-bits value at round i, the cipher specifies as follows:

$$\begin{aligned}
\hat{B}_0 &\leftarrow P \\
\hat{B}_{i+1} &\leftarrow R_i(\hat{B}_i) \\
C &\leftarrow \hat{B}_{32}
\end{aligned}$$

with

$$\begin{aligned}
R_i(X) &= LT(\hat{S}_i(X \oplus \hat{K}_i)) \text{ For } \ i = 0, \cdots, 30 \\
R_i(X) &= \hat{S}_i(X \oplus \hat{K}_i) \oplus \hat{K}_{32} \ \text{ For } i = 31
\end{aligned}$$

I Explanation of figure 6

Figure 6 describes both the differences (top half) and the values (bottom half) which are required at the output of the S_0 layer (state #1). On the top half, if a number is written in a cell, then the bit difference on this cell will influence the difference in the column indicated by the number written inside at the input of S_1 in state #2. The color of the background of such a cell indicates whether the difference is fixed (grey) or variable (white). On the bottom half, if an indexed number a_b is written inside a cell, then the bit associated to this cell will influence the value of x_b at column a at the input of S_1 in state #2. Symbols are used when more than one bit is influenced, there is a legend above the figure for these. We only need the values associated to columns $31, 18$ and 17 because the other two are treated probabilistically. The colors of the columns (also indexed with lower case letters from a to f) indicate those where we can reduce the amount of keybits we guess below four.