Improved Differential-Linear Attacks with Applications to ARX Ciphers

Christof Beierle, Marek Broll, Federico Canale, Nicolas David, Antonio Flórez-Gutiérrez, Gregor Leander, María Naya-Plasencia, Yosuke Todo

▶ **To cite this version:**

Christof Beierle, Marek Broll, Federico Canale, Nicolas David, Antonio Flórez-Gutiérrez, et al.. Improved Differential-Linear Attacks with Applications to ARX Ciphers. Journal of Cryptology, 2022, 35 (4), pp.29. 10.1007/s00145-022-09437-z . hal-03947756

HAL Id: hal-03947756
https://inria.hal.science/hal-03947756

Submitted on 19 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Improved Differential-Linear Attacks with Applications to ARX Ciphers

Christof Beierle[1], Marek Broll[1], Federico Canale[1], Nicolas David[2], Antonio Flórez-Gutiérrez[2], Gregor Leander[1], María Naya-Plasencia[2] and Yosuke Todo[3*]

[1]Ruhr University Bochum, Bochum, Germany.
[2]Inria, Paris, France.
[3]NTT Social Informatics Laboratories, Tokyo, Japan.

*Corresponding author(s). E-mail(s):
yosuke.todo.xt@hco.ntt.co.jp;
Contributing authors: christof.beierle@rub.de;
marek.broll@rub.de; federico.canale@rub.de;
nicolas.david@inria.fr; antonio.florez-gutierrez@inria.fr;
gregor.leander@rub.de; maria.naya_plasencia@inria.fr;

Abstract

We present several improvements to the framework of differential-linear attacks with a special focus on ARX ciphers. As a demonstration of their impact, we apply them to Chaskey and ChaCha and we are able to significantly improve upon the best attacks published so far.

Keywords: Symmetric Cryptanalysis, ARX, Chaskey, ChaCha

1 Introduction

Symmetric cryptographic primitives play major roles in virtually any cryptographic scheme and security-related application. The main reason for this massive deployment of symmetric primitives, i.e., (tweakable) block ciphers,

This article is an extended version of the paper presented at CRYPTO 2020 [1]. Some further improvements introduced in [2] are included.

stream ciphers, hash functions, or cryptographic permutations, is their significant performance advantage. Symmetric primitives usually outperform other cryptographic schemes by up to several orders of magnitude.

One class of design of symmetric primitives that is inherently motivated by (software) efficiency is the *ARX-based* design. ARX is short for **a**ddition (modulo a power of two), word-wise **r**otation and **X**OR. Indeed, ciphers following this framework are composed of those operations and avoid the computation of smaller S-boxes through look-up tables. As CPUs may have these operations implemented on the hardware level, particularly an addition unit and a barrel shifter, executing them on such CPUs based on a suitable register size is inherently fast.

The block cipher FEAL [3] was probably the first ARX cipher presented in the literature, and by now, several state-of-the-art ciphers follow this approach. One of the most important (families of) ARX ciphers is certainly the one formed by Salsa20, ChaCha and their variants [4, 5]. Designed by Bernstein, these ciphers are now the default replacement for RC4 in TLS due to their high efficiency and the simplicity of their implementations, and are thus some of the most widely-used ciphers in practice. Besides being used in TLS, ChaCha is also deployed in several other products, and in particular, it is used as a building block in the popular hash functions Blake and Blake2 [6, 7].

The ARX-based design approach is not restricted to stream ciphers, as it can also be used in the design of efficient block ciphers (e.g., Sparx [8]), cryptographic permutations (e.g., Sparkle [9]), and message authentication codes (MACs). For the latter, Chaskey [10] is among the most prominent examples.

Besides the advantage of having efficient implementations, there are also good reasons for ARX-based designs regarding security. The algebraic degree of ARX ciphers is usually high after only a very few rounds, as the carry bit within one modular addition already has an almost maximal degree. Structural attacks like integral [11] or invariant attacks [12] are less of a concern and rotational cryptanalysis [13], originally invented for ARX ciphers, can be efficiently prevented in most cases by the XOR of constants.

When it comes to differential [14] and linear attacks [15], ARX-based designs often show a peculiar behavior. For a small number of rounds, i.e., only very few modular additions, the differential probabilities (resp., absolute linear correlations) are very high. In particular, for a single modular addition, those are equal to 1 due to the linear behavior of the least significant and, in the case of differentials, most significant bits. Moreover, for a single modular addition, the differential probabilities and linear correlations are well understood, and we have at hand nice and efficient formulas for their computation [16, 17]. In the case of (dependent) chains of modular additions, bitwise rotations, and XORs, the situation is different, and experimentally checking the probabilities is often the best way to evaluate the behavior.

Table 1 (Partial) Key-Recovery Attacks on Chaskey and ChaCha.

Target	Key size	Rounds	Time	Data	Ref.
Chaskey	**128**	6	$2^{28.6}$	2^{25}	[19]
		7	2^{67}	2^{48}	[19]
			2^{50}	$2^{40.21}$	Sect. 9.3
		7.5	2^{77}	2^{48}	Sect. 9.4
ChaCha	**256**	6	2^{139}	2^{30}	[22]
			2^{136}	2^{28}	[23]
			2^{116}	2^{116}	[20]
			2^{89}	2^{48}	Sect. 10.4
			$2^{77.4}$	2^{58}	Sect. 10.5
		7	2^{248}	2^{27}	[22]
			$2^{246.5}$	2^{27}	[23]
			$2^{238.9}$	2^{96}	[24]
			$2^{237.7}$	2^{96}	[20]
			$2^{235.22}$	–	[21]
			$2^{230.86}$	$2^{48.83}$	Sect. 10.6
		7.25	$2^{255.62}$	$2^{48.36}$	[25]

While a few rounds are thus very weak, for a well-crafted ARX scheme, the probabilities of differentials and the absolute correlations of linear approximations decrease very quickly as the number of rounds increases. Indeed, this property led to the *long-trail strategy* for designing ARX-based ciphers [8].

Now, for symmetric primitives, the existence of strong differentials and linear approximations for a few rounds with a rapid decrease of probabilities (resp. absolute correlations) is exactly the situation in which considering differential-linear attacks [18] is promising. In a nutshell, differential-linear attacks combine a differential with probability p for the first r rounds of the cipher and a linear approximation with correlation q for the next t rounds into a linear approximation for $r+t$ rounds with correlation pq^2 that can be turned into an attack with data complexity of roughly $p^{-2}q^{-4}$.

Indeed, it is not surprising that the best attacks against many ARX constructions, including ChaCha and Chaskey, are differential-linear [19–21]. Our work builds upon those ideas and improves differential-linear attacks on ARX ciphers along several dimensions.

1.1 Our Contribution

In this paper, we present the best known[1] attacks on Chaskey and ChaCha. Our improvements over prior work are based on improvements in the differential, the linear part, the LLR statistic, and the key-recovery part of differential-linear attacks.

Differential part

For the differential part, our observation is both simple and effective. Recall that for a differential-linear attack. One needs many (roughly q^{-4}) pairs to fulfill the difference in the first part of the cipher, that is many right pairs for the differential. Now, imagine that an attacker could construct many right pairs with probability (close to) one, given only a single right pair. It would immediately reduce the data complexity of the attack by a factor of p^{-1}. As we will see, this situation is rather likely to occur for a few rounds of many ARX ciphers, particularly for ChaCha and Chaskey. The details of those improvements are presented in Sect. 5.

Linear part

For the linear part, our first observation is that it is often beneficial to not restrict to a single mask but rather consider *multiple* linear approximations. As we detail in Sect. 6, this nicely combines with an improved version of the partitioning technique for ARX ciphers [19, 29], which splits the space of ciphertexts into subsets to increase the correlation of linear approximations. The starting point of our attacks is a new way of partitioning the ciphertexts, summarized in Sect. 3. Note that, although we use multiple linear masks in the attack, because of partitioning the ciphertexts, we basically use only a *single linear mask for each ciphertext*. This way, we avoid possible dependencies that would be hard to analyze otherwise.

LLR statistic

Our advanced partitioning technique for the linear part exploits linear approximations with different correlations for every ciphertext. One ciphertext pair causes high absolute correlation, but another might cause lower absolute correlation. It is not appropriate to treat these ciphertext pairs in the same manner. We use the log-likelihood ratio (LLR) statistic to solve this problem. According to the Neyman-Pearson lemma [30], the LLR test is the most powerful statistical test and, as such, has been used as a cryptanalytic tool (see e.g. [31, 32]). In our case, the use of the LLR statistic is beneficial because we can exploit all partitions which were discarded by the original partitioning technique in [1]. The details of the LLR-based technique are presented in Sect. 7.

[1] After presenting those results at CRYPTO 2020 [1], improved attacks on ChaCha have been proposed [26]. Later, [27] pointed out mistakes in some parts of [26], leading to an updated version that has been published on the Cryptology ePrint Archive [28]. Very recently, another improved differential-linear attack has been presented [25].

Key recovery

Related to the improvement in the linear part and LLR statistic, we present a significant speed-up in the key recovery part. Here, the main observation is that after considering multiple masks and the partitioning technique, several key bits appear only *linearly* in the approximations. In particular, their value does not affect the absolute value of the correlation but rather the sign only. Instead of guessing those keys individually as done in previous attacks, this observation allows us to recover them by applying the Fast Walsh-Hadamard Transform (FWHT). Similar ideas have already been described in [33]. Details of this approach are given in Sect. 8.

Putting these improvements into one framework and applying this framework to round-reduced variants of ChaCha and Chaskey results in significantly reduced attack complexities. Our attacks and their corresponding complexities are summarized in Table 1, together with a comparison to the best attacks published so far. It is important to note that, as these attacks are on *round-reduced* variants of the ciphers only, they do not pose any threat on the full-round versions of ChaCha or Chaskey. Rather, these attacks strengthen our trust in the design. We expect that our improvements have applications to other ciphers, especially ARX-based designs.

2 Preliminaries

By $\oplus$ we denote the XOR operation, i.e., addition in $\mathbb{F}_2^n$ and by $+$ we either denote the addition in $\mathbb{Z}$, or the modular addition mod 2^n (we identify elements of $\mathbb{F}_2^n$ with elements of $\mathbb{Z}$ by regarding them as binary representations), depending on the context. For $x \in \mathbb{F}_2^n$, we denote by $\bar{x}$ the bitwise complement of x. Note that we have $-x = \bar{x} + 1$. Given a non-empty set $\mathcal{S} \subseteq \mathbb{F}_2^n$ and a Boolean function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$, we define

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

If for (another) Boolean function $g\colon \mathbb{F}_2^n \to \mathbb{F}_2$ we have $\mathbf{Cor}_{x\in\mathcal{S}}\left[g(x)\oplus f(x)\right] = c$, we say that $g(x) \approx f(x)$ *holds with correlation* c *if* $x \in \mathcal{S}$.

We denote the ith unit vector of a binary vector space by $[i]$ and the sum of unit vectors $[i_1] \oplus [i_2] \oplus \cdots \oplus [i_t]$ by $[i_1, i_2, \ldots, i_t]$. Given a vector $x \in \mathbb{F}_2^n$, $x[i]$ denotes the ith bit of x, and $x[i_1, i_2, \ldots, i_t]$ denotes $\bigoplus_{j=1}^{t} x[i_j]$. For $\gamma, x \in \mathbb{F}_2^n$, we define the inner product by $\langle \gamma, x\rangle = \bigoplus_{i=0}^{n-1} \gamma[i]x[i]$. In particular, $x[i_1, i_2, \ldots, i_t] = \langle x, [i_1, i_2, \ldots, i_t]\rangle$.

In the remainder of this paper we assume that, when a randomly chosen sampling set $\mathcal{S} \subseteq \mathbb{F}_2^n$ is a (sufficiently large) subset of $\mathbb{F}_2^n$, $\mathbf{Cor}_{x\in\mathcal{S}}\left[f(x)\right]$ is a good approximation for $\mathbf{Cor}_{x\in\mathbb{F}_2^n}\left[f(x)\right]$. In other words, we assume that the empirical correlations obtained by sampling for a sufficiently large number of messages closely match the actual correlations.

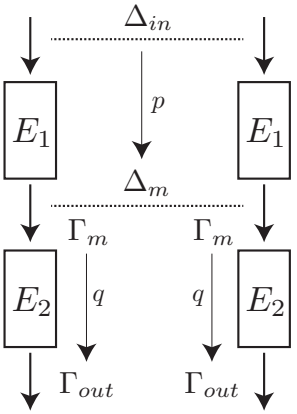


Fig. 1 The structure of a classical differential-linear distinguisher.

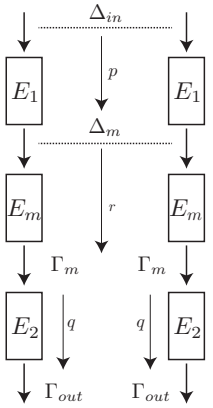


Fig. 2 A differential-linear distinguisher with experimental evaluation of the middle correlation r.

We denote by $\mathcal{N}(\mu, \sigma^2)$ the normal distribution with mean μ and variance σ^2. By Φ we denote the cumulative distribution function of the standard normal distribution $\mathcal{N}(0, 1)$. Thus if $X \sim \mathcal{N}(\mu, \sigma^2)$, it holds that

$$\mathbf{Pr}\left[X \leq \Theta\right] = \Phi\left(\frac{\Theta - \mu}{\sigma}\right) .$$

2.1 Differential-Linear Attacks

We first recall the basic variant of differential-linear cryptanalysis as introduced by Langford and Hellman [18], and the enhancement by Biham, Dunkelman, and Keller [34]. Figure 1 shows the overview of the distinguisher. An entire cipher E is divided into two sub ciphers E_1 and E_2, such that $E = E_2 \circ E_1$, and a differential distinguisher and a linear distinguisher is applied to the first and second part, respectively.

In particular, assume that the differential $\Delta_{\text{in}} \xrightarrow{E_1} \Delta_m$ holds with probability $\mathbf{Pr}_{x \in \mathbb{F}_2^n}\left[E_1(x) \oplus E_1(x \oplus \Delta_{\text{in}}) = \Delta_m\right] = p$.

Let us further assume that the linear approximation $\Gamma_m \xrightarrow{E_2} \Gamma_{\text{out}}$ has correlation $\mathbf{Cor}_{x \in \mathbb{F}_2^n}\left[\langle \Gamma_m, x\rangle \oplus \langle \Gamma_{\text{out}}, E_2(x)\rangle\right] = q$. The differential-linear distinguisher exploits the fact that, under the assumption that $E_1(x)$ and $E_2(x)$ are independent random variables, we have

$$\mathbf{Cor}_{x \in \mathbb{F}_2^n}\left[\langle \Gamma_{\text{out}}, E(x)\rangle \oplus \langle \Gamma_{\text{out}}, E(x \oplus \Delta_{\text{in}})\rangle\right] = pq^2 . \tag{1}$$

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

In practice, there might be a problem with the assumption that $E_1(x)$ and $E_2(x)$ are independent, resulting in wrong estimates for the correlation. To provide a better justification of this independence assumption (and in order to improve attack complexities), adding a middle part is a simple solution and

usually done in recent attacks (including ours). Here, the cipher E is divided into three sub ciphers E_1, E_m and E_2 such that $E = E_2 \circ E_m \circ E_1$ and the middle part E_m is experimentally evaluated. In particular, let

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

where $\mathcal{S}$ denotes the set of samples over which the correlation is computed. Then, the total correlation in Equation 1 can be estimated as prq^2. Recently, as a theoretical support for this approach, the Differential-Linear Connectivity Table (DLCT) [35], has been introduced. The overall attack framework is depicted in Fig. 2 and we will use this description in the remainder of the paper.

Finally, in order to better understand Equation (1), we denote the *differential-linear correlation* (known as the *auto-correlation* in the theory of Boolean functions) on E by

$$\mathrm{Aut}_E(\Delta_{\mathrm{in}}, \alpha, \alpha') := 2^{-n} \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \alpha, E(x) \rangle \oplus \langle \alpha', E(x \oplus \Delta_{\mathrm{in}}) \rangle} ,$$

where Equation (1) is special case such that $\alpha = \alpha' = \Gamma_{\mathrm{out}}$.

2.2 Partitioning Technique for ARX-based Designs

Partitioning allows increasing the correlation of the differential-linear distinguisher by deriving linear equations which hold conditioned on ciphertext and key bits. We recall the partitioning technique as used in [19]. Let $a, b \in \mathbb{F}_2^m$ and let $z = a + b$. When $i = 0$ (lsb), the modular addition for bit i becomes linear, i.e., $z[0] = a[0] \oplus b[0]$. Of course, for $i > 0$, the ith output bit of modular addition is not linear on the inputs. However, by restricting (a, b) to a specific subset, we might obtain other linear relations. In previous work, the following formula on $z[i]$ was derived.

Lemma 1 ([19]) *Let $a, b \in \mathbb{F}_2^m$ and $z = a + b$. For $i \geq 2$, we have*

$$z[i] = \begin{cases} a[i] \oplus b[i] \oplus a[i-1] & \text{if } a[i-1] = b[i-1] \\ a[i] \oplus b[i] \oplus a[i-2] & \text{if } a[i-1] \neq b[i-1] \text{ and } a[i-2] = b[i-2] . \end{cases}$$

3 New Partitioning Technique

Before introducing our new attack framework for differential-linear attacks, we first introduce some new partitioning techniques.

The original idea of the partitioning technique [29] 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 [19] also has the same feature, i.e., when a single modular addition is analyzed, all the data is divided into four partitions and one out of those four is discarded.

Our new partitioning techniques are twofold. First, we introduce linear masks for partitions that have originally been discarded. Our new FWHT-based key recovery using the LLR statistic allows us to efficiently use such

partitions. Second, we additionally introduce a partitioning technique to compute $z[i] \oplus z[i-1]$ with the same key guessing cost as the evaluation of $z[i]$. There are multiple linear trails with high correlation in the ARX ciphers. The new partition is useful when we dynamically change available linear trails for each partition.

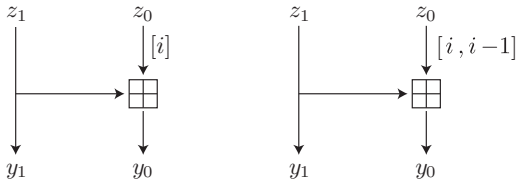


Fig. 3 Two examples of the partitioning technique. In the first case (left picture), we are interested in an approximation for $z_0[i]$. In the second case (right picture), we are interested in an approximation for $z_0[i] \oplus z_0[i-1]$. The corresponding partitions and approximations are given in Lemma 2 and Lemma 3, respectively.

Let us consider two m-bit words z_0 and z_1 and a modular addition operation

$$\mathbb{F}_2^{2m} \to \mathbb{F}_2^{2m}, \quad (z_1, z_0) \mapsto (y_1, y_0) = (z_1, z_0 + z_1)\ ,$$

as depicted in Fig. 3. In the attacks we present later, we are interested in the value $z_0[i] = (y_0 - y_1)[i] = (y_0 + \bar{y}_1 + 1)[i]$. Notice that for $i \leq 2$ there exist trivial relations for $z_0[i]$. The following lemma deals with the case $i \geq 3$, which is relevant for our applications.

Lemma 2 *Let $s = y_0 \oplus \bar{y}_1$ and let $i \geq 3$. Let $\mathcal{S}_{b_0 b_1} := \{(y_1, y_0) \in \mathbb{F}_2^{2m} \mid s[i-1] = b_0$ and $s[i-2] = b_1\}$. We have*

$$z_0[i] \approx \begin{cases} y_0[i] \oplus y_1[i] \oplus y_0[i-1], & \text{with cor. } -1, \quad \text{if } (y_1, y_0) \in \mathcal{S}_{0*}, \\ y_0[i] \oplus y_1[i] \oplus y_0[i-2], & \text{with cor. } -1, \quad \text{if } (y_1, y_0) \in \mathcal{S}_{10}, \\ y_0[i] \oplus y_1[i] \oplus y_0[i-3], & \text{with cor. } -2^{-1}, \quad \text{if } (y_1, y_0) \in \mathcal{S}_{11}, \end{cases} \tag{2}$$

where $\mathcal{S}_{0*} = \mathcal{S}_{00} \cup \mathcal{S}_{01}$.

Proof Figure 4 represents the computation of $z_0[i]$, where $z_0 = y_0 - y_1 = y_0 + \bar{y}_1 + 1$. In fact, if $c[i]$ denotes the carry occurring at bit position i, and assume that $c[-1] := 1$, we have that $z_0[i] = y_0[i] \oplus y_1[i] \oplus 1 \oplus c[i-1]$ for all $i \geq 0$.

Let us first assume that $(y_1, y_0) \in \mathcal{S}_{0*}$, i.e., $y_0[i-1] \oplus \bar{y}_1[i-1] = 0$. Then, $c[i-1] = y_0[i-1]$ if $i \geq 1$. Thus,

$$z_0[i] = y_0[i] \oplus y_1[i] \oplus 1 \oplus y_0[i-1],$$

for all $i \geq 1$, and we obtain the first equality.

Next, let us assume that $(y_1, y_0) \in \mathcal{S}_{10}$, i.e., $y_0[i-1] \oplus \bar{y}_1[i-1] = 1$ and $y_0[i-2] \oplus \bar{y}_1[i-2] = 0$. Then, $c[i-1] = c[i-2]$, and $c[i-2] = y_0[i-2]$ if $i \geq 2$. Thus,

$$z_0[i] = y_0[i] \oplus y_1[i] \oplus 1 \oplus y_0[i-2],$$

for all $i \geq 2$, and we obtain the second equality.

Finally, let us assume that $(y_1, y_0) \in \mathcal{S}_{11}$, i.e., $y_0[i-1] \oplus \bar{y}_1[i-1] = 1$ and $y_0[i-2] \oplus \bar{y}_1[i-2] = 1$. Then, $c[i-1] = c[i-2]$, and $c[i-2] = c[i-3]$ if $i \geq 3$. Thus,

$$z_0[i] = y_0[i] \oplus y_1[i] \oplus 1 \oplus c[i-3]$$

holds for all $i \geq 3$. The carry $c[i-3]$ is the output of the majority function as $c[i-3] = maj(c[i-4], y_0[i-3], \bar{y}_1[i-3])$, and a linear approximation, $c[i-3] \approx y_0[i-3]$, holds with correlation 2^{-1}. Thus, we have

$$z_0[i] \approx y_0[i] \oplus y_1[i] \oplus y_0[i-3]$$

with correlation -2^{-1}, and we obtain the final approximation. □

$$\begin{array}{cccccc} & c[i-1] & c[i-2] & c[i-3] & c[i-4] & \\ & y_0[i] & y_0[i-1] & y_0[i-2] & y_0[i-3] & \cdots \\ + & \bar{y}_1[i] & \bar{y}_1[i-1] & \bar{y}_1[i-2] & \bar{y}_1[i-3] & \cdots \\ \hline & z_0[i] & z_0[i-1] & z_0[i-2] & z_0[i-3] & \cdots \end{array}$$

Fig. 4 Representation for $z_0 = y_0 + \bar{y}_1 + 1$.

The representations for the partitions $\mathcal{S}_{0*}$ and $\mathcal{S}_{10}$ are the same as in Lemma 1. We additionally introduce a linear approximation for the partition $\mathcal{S}_{11}$, which was discarded in the original partitioning technique. Note that the cost to determine partitions is not increased compared to the previous partitioning technique because we simply use the discarded partition. The cost increase only involves the new bit $y_0[i-3]$, but thanks to our FWHT-based key recovery technique, the cost increase will be negligible. We discuss it in detail in Sect. 8.

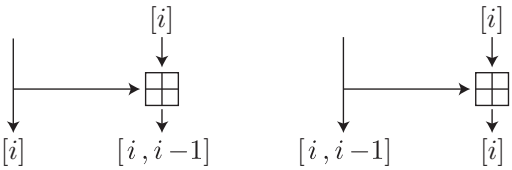


Fig. 5 Two linear trails with correlation 2^{-1}.

Due to the propagation rules for linear trails over modular addition, we may end up with multiple linear trails that are closely related to each other. As an example, Fig. 5 shows two possible trails, where $[i]$ and $[i, i-1]$ denote the corresponding linear masks. The partitioning technique described above evaluates $z_0[i]$, but we can expect that there is a highly-biased linear trail in

which $z_0[i] \oplus z_0[i-1]$ needs to be evaluated instead of $z_0[i]$. In the trivial method, we would apply the partitioning technique of Lemma 2 for $z_0[i]$ and $z_0[i-1]$ separately, which requires the knowledge of 3 bits of information from y in total. Our new partitioning method allows us to determine the partition with the knowledge of only the same 2 bits of information as needed for evaluating the case of $z_0[i]$, namely $(y_0[i-1] \oplus y_1[i-1])$ and $(y_0[i-2] \oplus y_1[i-2])$. This is especially helpful if y consists of the ciphertext XORed with the key, so we need to guess less key bits to evaluate the partition.

Lemma 3 *Let $s = y_0 \oplus \bar{y}_1$ and let $i \geq 3$. Let $\mathcal{S}_{b_0 b_1} \coloneqq \{(y_1, y_0) \in \mathbb{F}_2^{2m} \mid s[i-1] = b_0$ and $s[i-2] = b_1\}$. We have*

$$z_0[i] \oplus z_0[i-1] \approx \begin{cases} y_0[i] \oplus y_1[i], & \text{with cor. } 1, \quad \text{if } (y_1, y_0) \in \mathcal{S}_{1*}, \\ y_0[i] \oplus y_1[i] \oplus y_0[i-1] \oplus y_0[i-2], & \text{with cor. } -1, \quad \text{if } (y_1, y_0) \in \mathcal{S}_{00}, \\ y_0[i] \oplus y_1[i] \oplus y_0[i-1] \oplus y_0[i-3], & \text{with cor. } -2^{-1}, \quad \text{if } (y_1, y_0) \in \mathcal{S}_{01}, \end{cases}$$

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

Proof By evaluating the modular addition $z_0 = y_0 + \bar{y}_1 + 1$, we have

$$z_0[i] \oplus z_0[i-1] = y_0[i] \oplus \bar{y}_1[i] \oplus c[i-1] \oplus y_0[i-1] \oplus \bar{y}_1[i-1] \oplus c[i-2] \ ,$$

where $c[i-1]$, resp., $c[i-2]$ denotes the carry occurring at bit position $i-1$, resp., $i-2$. As before, we define $c[-1] \coloneqq 1$.

Let us first assume that $(y_1, y_0) \in \mathcal{S}_{1*}$, i.e., $y_0[i-1] \oplus \bar{y}_1[i-1] = 1$. Then, clearly $c[i-1] = c[i-2]$. Thus,

$$z_0[i] \oplus z_0[i-1] = y_0[i] \oplus y_1[i],$$

and we obtain the first equality.

Next, let us assume that $(y_1, y_0) \in \mathcal{S}_{00}$, i.e., $y_0[i-1] \oplus \bar{y}_1[i-1] = 0$ and $y_0[i-2] \oplus \bar{y}_1[i-2] = 0$. Since $y_0[i-1] \oplus \bar{y}_1[i-1] = 0$, we have $c[i-1] = \bar{y}_1[i-1]$. Since $y_0[i-2] \oplus \bar{y}_1[i-2] = 0$, we have $c[i-2] = y_0[i-2]$. Thus,

$$z_0[i] \oplus z_0[i-1] = y_0[i] \oplus y_1[i] \oplus 1 \oplus y_0[i-1] \oplus y_0[i-2],$$

and we obtain the second equality.

Finally, let us assume that $(y_1, y_0) \in \mathcal{S}_{01}$, i.e., $y_0[i-1] \oplus \bar{y}_1[i-1] = 0$ and $y_0[i-2] \oplus \bar{y}_1[i-2] = 1$. Since $y_0[i-1] \oplus \bar{y}_1[i-1] = 0$, we have $c[i-1] = \bar{y}_1[i-1]$. Since $y_0[i-2] \oplus \bar{y}_1[i-2] = 1$, we have $c[i-2] = c[i-3]$. Thus,

$$z_0[i] \oplus z_0[i-1] = y_0[i] \oplus y_1[i] \oplus 1 \oplus y_0[i-1] \oplus c[i-3].$$

The carry $c[i-3]$ is the output of the majority function as $c[i-3] = maj(c[i-4], y_0[i-3], \bar{y}_1[i-3])$, and a linear approximation, $c[i-3] \approx y_0[i-3]$, holds with correlation 2^{-1}. Thus,

$$z_0[i] \oplus z_0[i-1] \approx y_0[i] \oplus y_1[i] \oplus y_0[i-1] \oplus y_0[i-3]$$

holds with correlation -2^{-1}, and we obtain the final approximation. □

In practice, we need to consider a more complicated partition. For example, we sometimes consider the case that the approximated function consists of multiple modular subtractions and rotation. A summary of our partitioning is given in Appendix A.

4 High-Level Overview of the New Attack Framework

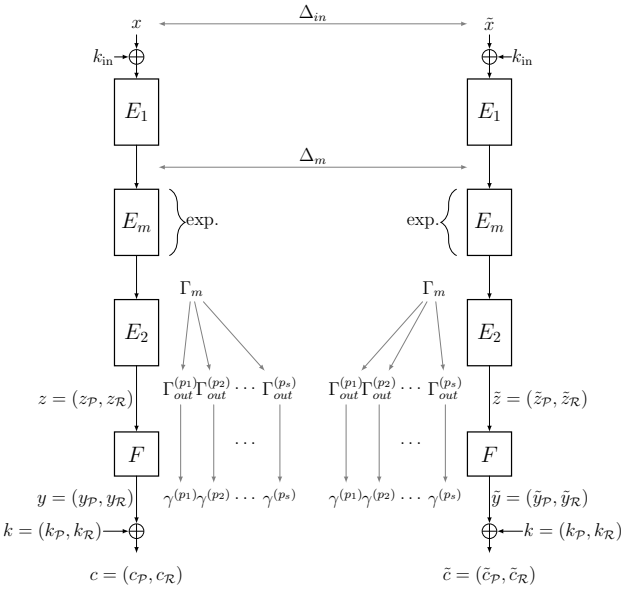


Fig. 6 The new attack framework.

In this section, we introduce a high-level overview of our new attack framework for differential-linear attacks with the partitioning technique. The framework consists of several novel techniques, which are: 1) amplifying the probability of the differential part by carefully choosing an appropriate linear subspace $\mathcal{U}$ for generating good pairs, 2) choosing the linear masks dynamically depending on each partition, and 3) an FWHT-based technique for improving the key recovery part when using partitions.

Figure 6 shows the high-level description of the general structure. Here F corresponds to the part of the cipher that we are going to cover using our improved key-guessing strategy. Our aim is to recover parts of the last whitening key k by using a differential-linear distinguisher given by s (multiple) linear approximations $\langle \Gamma_{\text{out}}^{(p_i)}, z\rangle \oplus \langle \Gamma_{\text{out}}^{(p_j)}, \tilde{z}\rangle$. In the following, we assume that the ciphertext space $\mathbb{F}_2^n$ is split into a direct sum $\mathcal{P} \oplus \mathcal{R}$ with $n_{\mathcal{P}} := \dim \mathcal{P}$ and $n_{\mathcal{R}} := \dim \mathcal{R} = n - n_{\mathcal{P}}$, so that the partitions will be given by the cosets $\mathcal{T}_{p_i} = p_i \oplus \mathcal{R}$ for any $p_i \in \mathcal{P}$ (i.e. p_i represents a set of the partition). Notice

that, without loss of generality, we can also assume that $c \in \mathbb{F}_2^n$ is divided into two parts $c_{\mathcal{P}} \in \mathbb{F}_2^{n_{\mathcal{P}}} = \mathcal{P}$ and $c_{\mathcal{R}} \in \mathbb{F}_2^{n-n_{\mathcal{P}}} = \mathcal{R}$, and $c = (c_{\mathcal{P}}, c_{\mathcal{R}})$; similarly, we can write $k = (k_{\mathcal{P}}, k_{\mathcal{R}})$ and $y = (y_{\mathcal{P}}, y_{\mathcal{R}})$. Then, for any $p_i \in \mathcal{P}$, the partition $\mathcal{T}_{p_i} \subset \mathbb{F}_2^n$ is defined as $\mathcal{T}_{p_i} = \{y \in \mathbb{F}_2^n \mid y_{\mathcal{P}} = p_i\}$. Since both formalizations are equivalent, we will use either interchangeably depending on the context.

4.1 The Differential Part

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

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

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, we use a linear subspace $\mathcal{U}$. In the simplest case, this subspace is chosen such that for any $x \in \mathcal{X}$ and any $u \in \mathcal{U}$ it holds that $x \oplus u \in \mathcal{X}$, i.e. $(x \oplus u, x \oplus u \oplus \Delta_{\text{in}})$ 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

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

Once we find an $x \in \mathcal{X}'$, we can find $2^{\dim \mathcal{U}}$ right pairs for free. The differential probability p is in fact defined as $p = |\mathcal{X}|/2^n$, which means we can reduce the data complexity by the factor p^{-1} when $\mathcal{X}' = \mathcal{X}$. We would like to remark that this idea has already been used in other contexts, e.g., the differential attack [36], but to best of our knowledge, it has not been applied to differential-linear attacks. We discuss the differential part in detail in Sect. 5.

4.2 The Linear Part

The idea is to identify several tuples $(\mathcal{T}_{p_i}, \Gamma_{\text{out}}^{(p_i)}, \gamma^{(p_i)})$, $i \in \{1, \ldots, s\}$, where $\gamma^{(p_i)} \in \mathcal{R}$, for which we can observe a high absolute correlation

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

In the simplest case, we would have $\varepsilon_i = 1$, i.e.,

$$y \in \mathcal{T}_{p_i} \quad \Rightarrow \quad \left(\langle \Gamma_{\text{out}}^{(p_i)}, z \rangle = \langle \gamma^{(p_i)}, y \rangle = \langle \gamma^{(p_i)}, c \rangle \oplus \langle \gamma^{(p_i)}, k \rangle \right) .$$

In other words, by considering only a specific subset of the ciphertexts (defined by $\mathcal{T}_{p_i}$) we obtain *linear relations* in the key with a high correlation.

The correlation of the differential-linear distinguisher for $E_2 \circ E_m \circ E_1$, which is denoted by $q_{i,j}$, is defined as

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

In practice, it is not feasible to compute $q_{i,j}$ for all partitions indexed by (i, j) when each correlation is low and the number of partitions is large. Therefore, we introduce the following assumption.

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

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

We finally observe the following correlation for $(y, \tilde{y})$ by guessing the secret key, and the final correlation is defined as

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

In addition to Assumption 1, we use the following assumption in order to estimate the final correlation.

Assumption 2 *The correlations $q_{i,j}$, ε_i, and ε_j are independent of each other, and $\rho_{i,j}$ can be estimated as*

$$\rho_{i,j} = \varepsilon_i \varepsilon_j q_{i,j}. \tag{3}$$

Unfortunately, Assumption 2 does not hold in general because it ignores the impact of the *auto-correlation-linear hull effect*. Namely, for a more precise evaluation, we need to consider multiple differential-linear trails with tuples $(\mathcal{T}_{p_i}, *, \gamma^{(p_i)})$, where $*$ represents arbitrary linear masks, and $\rho_{i,j}$ can be computed as the sum of these correlations. In other words, we need to consider multiple $\Gamma_{\text{out}}^{(p_i)}$ for each fixed $(\mathcal{T}_{p_i}, \gamma^{(p_i)})$. We later discuss the auto-correlation-linear hull in Sect. 6. There it is also shown that, when considering the auto-correlation-linear hull, Assumptions 1 and 2 are replaced by the assumption that the hull is dominated by a single trail in order to justify Equation (3).

How to identify belonging partitions is very important. It highly depends on the specification of the target cipher. Applications to Chaskey and ChaCha are too complicated to understand this behavior. We provide some simple cases for a more easy understanding of this behavior in Appendix B.

4.3 LLR-Based Statistical Test

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. [31, 32]). Considering the use of multiple linear trails with different correlations, the LLR test is more appropriate than the simple sum without considering the different correlations of each linear approximation.

Let us consider our differential-linear attacks using N pairs. An important remark is that each of the N pairs contributes differently to the correlations. Therefore, we need to consider the contribution to the theoretical correlation of each of them. Let $(y, \tilde{y})$ be the ℓth pair and let us assume that y and $\tilde{y}$ belong to the ith and jth partitions respectively, i.e., $y_{\mathcal{P}} = p_i$ and $\tilde{y}_{\mathcal{P}} = p_j$ (for

ease of notation, we do not make the dependency of $y, \tilde{y}, i, j$ on ℓ explicit). We then get the 1-bit representation

$$w_\ell = \langle y, \gamma^{(p_i)} \rangle \oplus \langle \tilde{y}, \gamma^{(p_j)} \rangle$$

and consider the probability

$$\pi_\ell = \mathbf{Pr}_{y,\tilde{y}} [w_\ell = 0] \ .$$

We refer to the theoretical correlation as $\rho_{i,j}$ when the ith and jth partitions are used. Namely, $\pi_\ell = \frac{\rho_{i,j}+1}{2}$. For simplicity, let C_ℓ denote this correlation for the ℓth pair (and the ith and jth partitions are used for this pair), i.e., $C_\ell = \rho_{i,j} = 2\pi_\ell - 1$.

Let D_0 and D_1 be the random vector 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 $\mathcal{B}(n, \pi_\ell)$ are independent binomial distributions with n trials and success probability π_ℓ, and where the π_ℓ are not necessarily distinct.

Our goal is to distinguish whether $w := (w_1, \ldots, w_N)$ is the result of sampling from D_0 (i.e., the real distribution) or D_1 (i.e., the random distribution). Let q_0 and q_1 be the probability that $w := (w_1, \ldots, w_N)$ is the result of sampling from D_0 or D_1 respectively. Thus,

$$q_0 = \mathbf{Pr} [X = w \mid X \sim D_0] , \qquad q_1 = \mathbf{Pr} [X = w \mid X \sim D_1] .$$

The LLR statistic is defined as $\ln(q_0/q_1)$, and it is computed as

$$\ln \left(\frac{q_0}{q_1} \right) = \frac{1}{2} \sum_{\ell=1}^{N} \ln \left(1 - C_\ell^2\right) + \frac{1}{2} \sum_{\ell=1}^{N} \ln \left(\frac{1 - C_\ell}{1 + C_\ell} \right) (-1)^{w_\ell} .$$

Let us assume that the LLR statistic follows normal distributions $\mathcal{N}(\mu_0, \sigma_0^2)$ and $\mathcal{N}(\mu_1, \sigma_1^2)$ when the correct and wrong keys are guessed, respectively. We have

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

where by $\overline{C}$ we denote the average of the squared correlation, i.e. $\overline{C} := \frac{1}{N} \sum_{\ell=1}^{N} C_\ell^2$. We later show the formula described above in Sect. 7.

4.4 WHT-Based Key Recovery Technique

We use $(y, \tilde{y})$ to identify partitions and compute the LLR statistic. Note that $y \in \mathcal{T}_{p_i} \Leftrightarrow c \in \mathcal{T}_{p_i} \oplus k_\mathcal{P}$, so we need to guess $n_\mathcal{P}$ bits of k to partition the ciphertexts into the corresponding $\mathcal{T}_{p_i}$. Note that, since we require $\gamma^{(p_i)} \in \mathcal{R}$, we obtain linear relations only on $k_\mathcal{R}$.

Let $(c, \tilde{c})$ be the pair of ciphertexts. The partition the pair belongs to is determined by $y_\mathcal{P} = c_\mathcal{P} \oplus k_\mathcal{P}$. Therefore, part of the key $k_\mathcal{P}$ is guessed to

identify the partition. After guessing $k_{\mathcal{P}}$, we get the following approximation.

$$\langle y, \gamma^{(y_{\mathcal{P}})} \rangle \approx \langle \tilde{y}, \gamma^{(\tilde{y}_{\mathcal{P}})} \rangle \Rightarrow \langle c \oplus k, \gamma^{(y_{\mathcal{P}})} \rangle \approx \langle \tilde{c} \oplus k, \gamma^{(\tilde{y}_{\mathcal{P}})} \rangle$$
$$\Rightarrow \langle c, \gamma^{(y_{\mathcal{P}})} \rangle \oplus \langle \tilde{c}, \gamma^{(\tilde{y}_{\mathcal{P}})} \rangle \approx \langle k, \gamma^{(y_{\mathcal{P}})} \oplus \gamma^{(\tilde{y}_{\mathcal{P}})} \rangle$$

Since the left side is known, we get the parity of k with respect to the linear mask $\gamma^{(y_{\mathcal{P}})} \oplus \gamma^{(\tilde{y}_{\mathcal{P}})}$. For a linear subspace W defined by $W := \mathrm{Span}\{\gamma^{(p_i)} \oplus \gamma^{(p_j)} \mid i, j \in \{1, \ldots, s\}\}$, the approximations above involve $\dim W$ bits of information for k. By using the Fast Walsh-Hadamard Transform (FWHT), 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 $k_{\mathcal{P}}$. We discuss this procedure in detail in Sect. 8.

5 The Differential Part – Finding Many Right Pairs

Let us be given a permutation $E_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and a differential $\Delta_{\mathrm{in}} \xrightarrow{E_1} \Delta_m$ that holds with probability p. In other words,

$$|\{x \in \mathbb{F}_2^n \mid E_1(x) \oplus E_1(x \oplus \Delta_{\mathrm{in}}) = \Delta_m\}| = p \cdot 2^n \ .$$

In a usual differential-linear attack on a permutation $E = E_2 \circ E_m \circ E_1$ as explained in Sect. 2.1, the internal structure of E_1 could be in general arbitrary and we would consider randomly chosen $x \in \mathbb{F}_2^n$ to observe the ciphertexts of the plaintext pairs $(x, x \oplus \Delta_{\mathrm{in}})$. For each of those pairs, the differential over E_1 is fulfilled with probability p, which results in a data complexity of roughly $\epsilon p^{-2} r^{-2} q^{-4}$ for the differential-linear attack. In other words, we did not exploit the particular structure of E_1. In particular, it would be helpful to know something about the distribution of right pairs $(x, x \oplus \Delta_{\mathrm{in}}) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ which fulfill the above differential.

Let us denote by $\mathcal{X}$ the set of all values that define right pairs for the differential, i.e.,

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

To amplify the correlation of a differential-linear distinguisher, instead of choosing random plaintexts from $\mathbb{F}_2^n$, we could consider only those that are in $\mathcal{X}$. In particular, we have[2]

$$\mathbf{Cor}_{x \in \mathcal{X}} \left[\langle \Gamma_{\mathrm{out}}, E(x) \rangle \oplus \langle \Gamma_{\mathrm{out}}, E(x \oplus \Delta_{\mathrm{in}}) \rangle \right] = rq^2 \ .$$

Since the set $\mathcal{X}$ might have a rather complicated structure and is key-dependent, we cannot use this directly for an arbitrary permutation E_1. However, if $\mathcal{X}$ presents a special structure such that, given one element $x \in \mathcal{X}$, we can generate many other elements in $\mathcal{X}$ for free,[3] independently of the secret key, we can use this to reduce the data complexity in a differential-linear attack. For example, if $\mathcal{X}$ contains a large affine subspace $\mathcal{A} = \mathcal{U} \oplus a$,

[2] under the assumption that the sets $\{\langle \Gamma_{\mathrm{out}}, E(x) \rangle \oplus \langle \Gamma_{\mathrm{out}}, E(x \oplus \Delta_{\mathrm{in}}) \rangle \mid x \in \mathcal{X}\}$ and $\{\langle \Gamma_{\mathrm{out}}, E(x) \rangle \oplus \langle \Gamma_{\mathrm{out}}, E(x \oplus \Delta_{\mathrm{in}}) \rangle \mid x \in \mathcal{S}\}$ are indistinguishable, where $\mathcal{S}$ denotes a set of uniformly chosen samples of the same size as $\mathcal{X}$.

[3] Or at least with a cost much lower than p^{-1}, see Sect. 5.2.

given $x \in \mathcal{A}$, we can generate (roughly) $2^{\dim \mathcal{U}}$ elements in $\mathcal{X}$ for free, namely all elements $x \oplus u$, for $u \in \mathcal{U}$. In order to obtain an effective distinguisher, we must be able to generate enough plaintext pairs to observe the correlation of the differential-linear approximation. In particular, we require $|\mathcal{U}| > \epsilon r^{-2} q^{-4}$.

This will be exactly the situation we find in ChaCha. Here the number of rounds covered in the differential part is so small that it can be described by the independent application of two functions (see Sect. 5.1).

If $|\mathcal{U}|$ is smaller than the threshold of $\epsilon r^{-2} q^{-4}$, we cannot generate enough right pairs for free to obtain a distinguisher by this method and we might use a probabilistic approach, see Sect. 5.2.

In Sect. 5.3 we show how the conditional differential framework can be used to efficiently find bigger sets $\mathcal{U}$, as well as to recover some information on the key, and provide some ideas to adapt it to the ARX scenario.

5.1 Fully Independent Parts

Let $E_1 \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ with $n = 2m$ be a parallel application of two block ciphers $E_1^{(i)} \colon \mathbb{F}_2^m \to \mathbb{F}_2^m$, $i \in \{0, 1\}$ (for a fixed key), i.e.,

$$E_1 \colon (x^{(1)}, x^{(0)}) \mapsto (E_1^{(1)}(x^{(1)}), E_1^{(0)}(x^{(0)})) \ .$$

Suppose that, $E_1^{(0)}$ presents a differential $\alpha \overset{E_1^{(0)}}{\to} \beta$ with probability p. We consider the differential $\Delta_{\text{in}} \overset{E_1}{\to} \Delta_m$ with $\Delta_{\text{in}} = (0, \alpha)$ and $\Delta_m = (0, \beta)$, which also holds with probability p. Given one element $(x^{(1)}, x^{(0)}) \in \mathcal{X}$, any $(x^{(1)} \oplus u, x^{(0)})$ for $u \in \mathbb{F}_2^m$ is also contained in $\mathcal{X}$, and we can thus generate 2^m right pairs for free.

If $2^m > \epsilon r^{-2} q^{-4}$, a differential-linear distinguisher on $E = E_2 \circ E_m \circ E_1$ would work as follows:

1. Choose $a = (a^{(1)}, a^{(0)}) \in \mathbb{F}_2^n$ uniformly at random.
2. Empirically compute
$$\mathbf{Cor}_{x \in a \oplus (\mathbb{F}_2^m \times \{0\})} \left[\langle \Gamma_{\text{out}}, E(x) \rangle \oplus \langle \Gamma_{\text{out}}, E(x \oplus \Delta_{\text{in}}) \rangle \right] \ .$$
3. If we observe a correlation of rq^2 using $\epsilon r^{-2} q^{-4}$ many x, the distinguisher succeeded. If not, start over with Step 1.

With probability p, we choose an element $a \in \mathcal{X}$ in Step 1. In that case, the distinguisher succeeds in Step 3. Therefore, the data complexity of the distinguisher is $\epsilon p^{-1} r^{-2} q^{-4}$, compared to $\epsilon p^{-2} r^{-2} q^{-4}$ as in the classical differential-linear attack.

5.2 Probabilistic Independent Parts

Since the previous decomposition is not always possible or 2^m might not be big enough, we are also interested in the situations in which the differential part cannot be simply written as the parallel application of two functions. Again, the goal is, given one element $x \in \mathcal{X}$, to be able to generate $\epsilon r^{-2} q^{-4}$ other elements in $\mathcal{X}$, each one with a much lower cost than

p^{-1}. Suppose that $\mathcal{U} \subseteq \mathbb{F}_2^n$ is a subspace with $|\mathcal{U}| > \epsilon r^{-2} q^{-4}$ and suppose that $\mathbf{Pr}_{u \in \mathcal{U}} [x \oplus u \in \mathcal{X} \mid x \in \mathcal{X}] = p_1$, where p_1 is much larger than p. The data complexity of the improved differential-linear distinguisher would then be $\epsilon p^{-1} p_1^{-2} r^{-2} q^{-4}$. Note that the probability p_1 may also depend on x. In particular, there might be $x \in \mathcal{X}' \subseteq \mathcal{X}$ for which p_1 is (almost) 1, but the probability to draw such an initial element x from $\mathbb{F}_2^n$ is p', which is smaller than p. Then, the data complexity would be $\epsilon p'^{-1} p_1^{-2} r^{-2} q^{-4}$. For instance, this will be the case for the attack on Chaskey (Sect. 9), where we have $p_1 \approx 1$ and $p' = p \times 222/256$.

In such situations, we propose an algorithmic way to experimentally detect suitable structures in the set of right pairs. The idea, see Algorithm 1 for the pseudo code, is to detect canonical basis vectors within the subspace $\mathcal{U}$. Running this algorithm for enough samples will return estimates of the probability γ_j that a right pair $x \in \mathcal{X}$ stays a right pair when the jth bit is flipped, i.e.,

$$\gamma_i = \mathbf{Pr} [x \oplus [i] \in \mathcal{X} \mid x \in \mathcal{X}] \ .$$

When applied to a few rounds of ARX ciphers it can be expected that there are some bits that will always turn a right pair into a right pair, i.e. $\gamma_i = 1$. Moreover, due to the property of the modular addition that the influence of bits on distant bits degrades quickly, high values of $\gamma_j \neq 1$ can also be expected. As we will detail in Sect. 9 this will be the case for the application to Chaskey.

Algorithm 1 Computing probabilistic independent bits

Require: Number of samples T, input difference Δ_{in}, output difference Δ_m
Ensure: Probabilities $\gamma_0, \gamma_1, \ldots, \gamma_{n-1}$

```
Let s = 0 and c_j = 0 for j ∈ {0, ..., n − 1}.
for i = 1 to T do
    Pick a random X and compute E_1(X) and E_1(X ⊕ Δ_in)
    if E_1(X) ⊕ E_1(X ⊕ Δ_in) = Δ_m then
        Increment s
        for j ∈ {0, ..., n − 1} do
            Prepare X̂ where the jth bit of X is flipped.
            if E_1(X̂) ⊕ E_1(X̂ ⊕ Δ_in) = Δ_m then
                Increment c_j
            end if
        end for
    end if
end for
for j ∈ {0, ..., n − 1} do
    γ_j = c_j / s
end for
```

5.3 Using the Conditional Differential Framework for Finding Better Subspaces

A practical way for producing a set of pairs of data whose elements all verify a certain differential path reflects a similar scenario to the one which is considered in conditional differential attacks. In [36], in the context of NLFSR-based primitives, the elements of the basis of $\mathcal{U}$ are called *freebits* and involve more complex relations derived from the differential paths than just simple single-bit relations. In Sect. 3.2 of [36], three types of conditions are presented: type zero, which only involve public bits (which is common in NLFSR initialization, but not in ARX or SPN constructions); type one, which involve both secret and public bits; and type two, which are conditions directly on the keybits. The *freebits* are actually the ones that do not affect type one conditions. 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 [36].

In this section we provide some hints and general ideas on how to use this framework for improving the analysis in ARX constructions. In Appendix C 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 for Chaskey.

Conditional differential framework for differential-linear attacks

Using the definitions from [36, Sect. 3.2], it is easy to see how to improve the differential part of some attacks (quite straightforwardly for ARX) in three main ways:

1. We can increase the size of $\mathcal{U}$ 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), as if these conditions were true, they would stay true for all the sampling set. These exhausted bits might not be completely free as type one conditions need to remain constant.
2. When considering a particular set of plaintexts to check if it is the one verifying the differential path, some information on the value of some associated keybits or keybit relations can be presupposed (given directly by conditions of type-2 and indirectly by conditions of type-1). This means that, for all the cases, we can suppose some information on the key as known. This information 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 keybits, that might be useful for the linear part, and that simultaneously might allow to detect sampling bit relations that follow the path with probability 1.

Main ideas for exploiting the conditions on ARX

We now present some general ideas for exploiting conditions on ARX constructions. Even though some of them might seem trivial, it is nonetheless helpful to set them as rules to follow.

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, and we present here four cases of particular interest: (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 will not 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 will not 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.

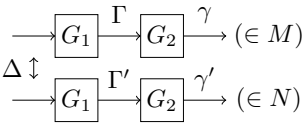
When dealing with carries, they might affect transitions with low probability. It is interesting to keep in mind that, when there is a sum of two zeros at position i, the value of all the bits at lower positions will not affect the carries at any higher positions. That might imply that a small guess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

6 Auto-Correlation-Linear Hulls and Partitioning

In this section, we want to better understand how to compute the correlations $\rho_{i,j}$ when the ith and jth partitions are used. This will allow us to shed light on Assumptions 1 and 2.

For this, we will derive general formulas which 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 the independence of

these parts. Furthermore, the notion of an *auto-correlation-linear hull* developed below will allow to improve upon the attacks by considering multiple intermediate masks.



We start by considering the unrestricted variant. Given two functions $G_1, G_2 \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$, an input difference Δ and two output-masks γ and γ', let $H := G_2 \circ G_1$ and

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

be the differential-linear correlation on H.

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

Note that this approach is different from the considerations in [37] as there it was about how to compute the auto-correlation by connecting the differential and the linear part correctly, while here we extend a differential-linear correlation using a second linear approximation of the parts.

This auto-correlation can be expressed as

$$\mathrm{Aut}_H(\Delta, \gamma, \gamma') = \sum_{u \in \mathbb{F}_2^n} \widehat{H}(u, \gamma) \widehat{H}(u, \gamma') (-1)^{\langle u, \Delta \rangle}, \tag{4}$$

where we denote by

$$\widehat{H}(u, \gamma) = 2^{-n} \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \gamma, H(x) \rangle + \langle u, x \rangle}$$

the correlation of the approximation of H with input and output masks u and γ. This follows from the connection of the Walsh-Hadamard transform and the convolution of functions, see e.g. [38, Proposition 11], but can also be verified directly.

In our attack framework, G_1 would correspond to $E_2 \circ E_m \circ E_1$ and G_2 to F and we would experimentally estimate the auto-correlation and multiply it with the correlation of G_2 with input mask Γ and output mask γ, i.e., we estimate

$$\mathrm{Aut}_H(\Delta, \gamma, \gamma') \approx \widehat{G}_2(\Gamma, \gamma) \widehat{G}_2(\Gamma', \gamma') \, \mathrm{Aut}_{G_1}(\Delta, \Gamma, \Gamma') \, . \tag{5}$$

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. Furthermore, we have to take the partitioning of the outputs into account.

For this, we use the fact (see [39]) that we can split the correlation of H into

$$\widehat{H}(u,\gamma) = \sum_{\Gamma\in\mathbb{F}_2^n} \widehat{G}_1(u,\Gamma)\widehat{G}_2(\Gamma,\gamma)$$

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

Proposition 1 (Auto-Correlation-Linear Hull) *Let* $G_1, G_2 \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$. *For any* $\Delta,\gamma,\gamma' \in \mathbb{F}_2^n$ *we then have*

$$\mathrm{Aut}_{G_2\circ G_1}(\Delta,\gamma,\gamma') = \sum_{\Gamma,\Gamma'\in\mathbb{F}_2^n} \widehat{G}_2(\Gamma,\gamma)\widehat{G}_2(\Gamma',\gamma')\,\mathrm{Aut}_{G_1}(\Delta,\Gamma,\Gamma')\,.$$

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

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

□

So, as could be expected, the linear hull theorem has a natural extension to an auto-correlation-linear hull theorem and the approximation in Equation (5) corresponds to focusing on a single (Γ,Γ') while actually all pairs (Γ,Γ') have to be considered. It remains to see how restricting the input, i.e. partitioning, affects this expression.

6.1 Impact of Partitioning on the Correlation

We again consider a function $H : \mathbb{F}_2^n \to \mathbb{F}_2^n$, an input difference Δ, output-masks γ and γ' and two non-empty subsets M, N of $\mathbb{F}_2^n$. We are interested in

$$\mathrm{Aut}_H^{(M,N)}(\Delta,\gamma,\gamma') \coloneqq \frac{2^n}{|M||N|} \sum_{\substack{x\in\mathbb{F}_2^n \\ H(x)\in M, H(x\oplus\Delta)\in N}} (-1)^{\langle\gamma,H(x)\rangle\oplus\langle\gamma',H(x\oplus\Delta)\rangle}.$$

One would hope that one can still use Equation (4) with minor modifications. That is basically by replacing the correlation of H by its restricted version.

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

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

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 Lemmas 4 and 5 below, we can state the main insight of this Section as Proposition 2.

Lemma 4 *We have*

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

Proof We start by expanding the right-hand side of the equation, denoted by L, as follows

$$\begin{aligned}
L &= \sum_{u\in\mathbb{F}_2^n} \widehat{H^{-1}|_M}(\gamma,u)\widehat{H^{-1}|_N}(\gamma',u)(-1)^{\langle u,\Delta\rangle} \\
&= \frac{1}{|M||N|}\sum_{u\in\mathbb{F}_2^n}\left(\sum_{y\in M}(-1)^{\langle u,H^{-1}(y)\rangle\oplus\langle\gamma,y\rangle}\right)\left(\sum_{y'\in N}(-1)^{\langle u,H^{-1}(y')\rangle\oplus\langle\gamma',y'\rangle}\right)(-1)^{\langle u,\Delta\rangle} \\
&= \frac{1}{|M||N|}\sum_{y\in M,y'\in N}(-1)^{\langle\gamma,y\rangle\oplus\langle\gamma',y'\rangle}\sum_{u\in\mathbb{F}_2^n}(-1)^{\langle u,H^{-1}(y)\oplus H^{-1}(y')\oplus\Delta\rangle} \\
&= \frac{2^n}{|M||N|}\sum_{\substack{y\in M,y'\in N\\ H^{-1}(y')=H^{-1}(y)\oplus\Delta}}(-1)^{\langle\gamma,y\rangle\oplus\langle\gamma',y'\rangle}.
\end{aligned}$$

We now define x as $H^{-1}(y)$ and we get

$$L = \frac{2^n}{|M||N|}\sum_{\substack{x\in\mathbb{F}_2^n\\ H(x)\in M,H(x\oplus\Delta)\in N}}(-1)^{\langle\gamma,H(x)\rangle\oplus\langle\gamma',H(x\oplus\Delta)\rangle}$$

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

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

Lemma 5 *Let $H = G_2 \circ G_1\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ and let $S \subseteq \mathbb{F}_2^n$ be a non-empty set. Then*

$$\widehat{H|_S}(\gamma,\Gamma) = \sum_{\mu\in\mathbb{F}_2^n}\widehat{G}_2(\mu,\Gamma)\widehat{G_1|_S}(\gamma,\mu)$$

Proof We have

$$\sum_{\mu\in\mathbb{F}_2^n}\widehat{G}_2(\mu,\Gamma)\widehat{G_1|_S}(\gamma,\mu)$$

$$
\begin{aligned}
&= \frac{1}{2^n|S|} \sum_{\mu\in\mathbb{F}_2^n} \left(\sum_{y\in\mathbb{F}_2^n} (-1)^{\langle \Gamma, G_2(y)\rangle \oplus \langle \mu, y\rangle} \right) \left(\sum_{x\in S} (-1)^{\langle \mu, G_1(x)\rangle \oplus \langle \gamma, x\rangle} \right) \\
&= \frac{1}{2^n|S|} \sum_{y\in\mathbb{F}_2^n, x\in S} (-1)^{\langle \Gamma, G_2(y)\rangle \oplus \langle \gamma, x\rangle} \sum_{\mu\in\mathbb{F}_2^n} (-1)^{\langle \mu, y\oplus G_1(x)\rangle} \\
&= \frac{1}{|S|} \sum_{x\in S, y=G_1(x)} (-1)^{\langle \Gamma, G_2(y)\rangle \oplus \langle \gamma, x\rangle} \\
&= \frac{1}{|S|} \sum_{x\in S} (-1)^{\langle \Gamma, G_2(G_1(x))\rangle \oplus \langle \gamma, x\rangle} = \widehat{H|_S}(\gamma, \Gamma) \, .
\end{aligned}
$$

□

Proposition 2 (Auto-Correlation-Linear Hull with Restriction) *Let* $G_1, G_2 \colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ *and let* $M, N \subseteq \mathbb{F}_2^n$ *be non-empty sets. For any* $\Delta, \gamma, \gamma' \in \mathbb{F}_2^n$ *we then have*

$$
\mathrm{Aut}^{(M,N)}_{G_2 \circ G_1}(\Delta, \gamma, \gamma') = \sum_{\Gamma, \Gamma' \in \mathbb{F}_2^n} \widehat{G_2^{-1}|_M}(\gamma, \Gamma) \widehat{G_2^{-1}|_N}(\gamma', \Gamma') \, \mathrm{Aut}_{G_1}(\Delta, \Gamma, \Gamma') \, .
$$

Proof Starting with Lemma 4, we express the restricted auto-correlation as

$$
T = \mathrm{Aut}^{(M,N)}_{H}(\Delta, \gamma, \gamma') = \sum_{u\in\mathbb{F}_2^n} \widehat{H^{-1}|_M}(\gamma, u) \widehat{H^{-1}|_N}(\gamma', u) (-1)^{\langle u, \Delta\rangle},
$$

and substitute the correlations by using Lemma 5 as

$$
\widehat{H^{-1}|_M}(\gamma, u) = \sum_{\Gamma\in\mathbb{F}_2^n} \widehat{G_1^{-1}}(\Gamma, u) \widehat{G_2^{-1}|_M}(\gamma, \Gamma)
$$

and

$$
\widehat{H^{-1}|_N}(\gamma', u) = \sum_{\Gamma'\in\mathbb{F}_2^n} \widehat{G_1^{-1}}(\Gamma', u) \widehat{G_2^{-1}|_N}(\gamma', \Gamma').
$$

Doing this we get

$$
\begin{aligned}
T &= \sum_{u\in\mathbb{F}_2^n} \left(\sum_{\Gamma\in\mathbb{F}_2^n} \widehat{G_1^{-1}}(\Gamma, u) \widehat{G_2^{-1}|_M}(\gamma, \Gamma) \right) \left(\sum_{\Gamma'\in\mathbb{F}_2^n} \widehat{G_1^{-1}}(\Gamma', u) \widehat{G_2^{-1}|_N}(\gamma', \Gamma') \right) (-1)^{\langle u, \Delta\rangle} \\
&= \sum_{\Gamma, \Gamma'\in\mathbb{F}_2^n} \widehat{G_2^{-1}|_M}(\gamma, \Gamma) \widehat{G_2^{-1}|_N}(\gamma', \Gamma') \sum_{u\in\mathbb{F}_2^n} \widehat{G_1^{-1}}(\Gamma, u) \widehat{G_1^{-1}}(\Gamma', u) (-1)^{\langle u, \Delta\rangle} \, .
\end{aligned}
$$

Using the fact that, when considering the correlation of the inverse, input and output masks get swapped, we can rewrite this as

$$
T = \sum_{\Gamma, \Gamma'\in\mathbb{F}_2^n} \widehat{G_2^{-1}|_M}(\gamma, \Gamma) \widehat{G_2^{-1}|_N}(\gamma', \Gamma') \sum_{u\in\mathbb{F}_2^n} \widehat{G}_1(u, \Gamma) \widehat{G}_1(u, \Gamma') (-1)^{\langle u, \Delta\rangle}.
$$

The last part, according to Equation (4), is nothing else than the auto-correlation of G_1 and thus we conclude

$$T = \sum_{\Gamma,\Gamma' \in \mathbb{F}_2^n} \widehat{G_2^{-1}|_M}(\gamma,\Gamma)\widehat{G_2^{-1}|_N}(\gamma',\Gamma')\,\mathrm{Aut}_{G_1}(\Delta,\Gamma,\Gamma')$$

as claimed. □

Relation to Assumptions 1 and 2

Proposition 2 provides a more precise interpretation of Equation (3). Recall that G_1 corresponds to $E_2 \circ E_m \circ E_1$ and G_2 to F. The part ε_i and ε_j correspond directly to $\widehat{G_2^{-1}|_M}(\gamma,\Gamma)$ and $\widehat{G_2^{-1}|_M}(\gamma,\Gamma)$, while the value $q_{i,j}$ is now replaced by $\mathrm{Aut}_{G_1}(\Delta,\Gamma,\Gamma')$. Our main observation is that we can still consider the unrestricted auto-correlations of G_1 in this hull, which means that Assumption 1 is actually not needed, and Assumption 2, about the independence of the parts, can be replaced by the assumption that the hull is dominated by a single trail, that is a single choice of intermediate masks Γ, Γ'.

Moreover, using Proposition 2 allows to improve upon the correlation by considering multiple intermediate masks, as we demonstrate in the application to Chaskey in Sect. 9.

7 LLR-based Statistical Test

Let us consider our differential-linear attacks using N pairs. Following the notation introduced in Sect. 4.3, we let $(y, \tilde{y})$ be the ℓth pair and we assume that y and $\tilde{y}$ belong to the ith and jth partitions respectively (we recall that, for ease of notation, we do not make the dependency of $y, \tilde{y}, i, j$ on ℓ explicit). Then, let us consider

$$w_\ell = \langle y, \gamma^{(p_i)} \rangle \oplus \langle \tilde{y}, \gamma^{(p_j)} \rangle$$

for the ℓth pair $(y, \tilde{y})$, and consider the probability

$$\pi_\ell = \mathbf{Pr}_{y,\tilde{y}}\left[w_\ell = 0\right].$$

Note that $\pi_\ell = \frac{\rho_{i,j}+1}{2}$, and for simplicity, let C_ℓ denote this correlation for the ℓth pair (and the ith and jth partitions are used for this pair), i.e., $C_\ell = \rho_{i,j} = 2\pi_\ell - 1$.

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 $0 \le \pi_\ell \le 1$ are not necessarily distinct.

Let q_0 and q_1 be the probabilities that $w := (w_1, \ldots, w_N)$ is the result of sampling from D_0 (i.e. from the real distribution) or D_1 (i.e. the random

distribution) respectively. Thus

$$q_0 = \mathbf{Pr}\left[w = X \mid X \sim D_0\right] = \prod_{\ell=1}^{N} \pi_\ell^{w_\ell}(1-\pi_\ell)^{1-w_\ell},$$
$$q_1 = \mathbf{Pr}\left[w = X \mid X \sim D_1\right] = \prod_{\ell=1}^{N} 2^{-1} = 2^{-N}.$$

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

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

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.

Note that, when we use C_ℓ instead of π_ℓ, the LLR statistic is rewritten as

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

We can now determine the means and variances of the LLR statistic under D_0 and D_1 in terms of the correlations C_l.

Proposition 3 *Let $\mathcal{W}$ and $\mathcal{R}$ be the LLR statistics when w is chosen from D_1 and D_0, respectively. Then, the means $\mathbb{E}[\mathcal{W}]$ and $\mathbb{E}[\mathcal{R}]$ are estimated as*

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

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

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

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,

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

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

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

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

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

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

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

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

$$\mathbb{E}[\mathcal{R}] \approx \frac{1}{2}\sum_{\ell=1}^{N} C_\ell^2.$$

□

Proposition 4 *Let $\mathcal{W}$ and $\mathcal{R}$ be the LLR statistics when w is chosen from D_1 and D_0, respectively. Then, the variances* $\mathrm{Var}[\mathcal{W}]$ *and* $\mathrm{Var}[\mathcal{R}]$ *are estimated as*

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

Proof 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. We have experimentally verified that this is reasonable. With this in mind, we can write

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

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:

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

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

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

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

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

□

7.1 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

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

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

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

$$\frac{\mu_0 - \mu_1}{\sigma_1} = \frac{N\overline{C}}{\sqrt{N\overline{C}}} = \sqrt{N\overline{C}} = \sqrt{\sum_{\ell=1}^{N} C_\ell^2}. \tag{6}$$

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 [1], where the same gap can approximately be represented as $\frac{\sum_{\ell=1}^{N}|C_\ell|}{\sqrt{N}} = \sqrt{N\overline{c}^2}$, where $\overline{c} = \frac{1}{N}\sum_{\ell=1}^{N}|C_\ell|$. In other words, this gap is proportional to the squared value of the average of the (absolute value of the) correlation ($\overline{c}$), while for the LLR statistic the same gap is proportional to the average of squared correlations ($\overline{C}$). We remark that the latter is always larger than the former, as expected when using the LLR.

8 WHT-based Key Recovery Technique

We use the LLR statistic to recover the secret key. Recall that the LLR statistic is calculated as

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

where $w_\ell = \langle y, \gamma^{(y_\mathcal{P})}\rangle \oplus \langle \tilde{y}, \gamma^{(\tilde{y}_\mathcal{P})}\rangle$ and $(y, \tilde{y})$ is the ℓth pair in N pairs. Only ciphertext pair $(c, \tilde{c})$ can be observed by attackers. Which partition the pair belongs to is determined by $y_\mathcal{P}$ and $\tilde{y}_\mathcal{P}$. Therefore, 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 $k_\mathcal{P}$, we can get the following 1-bit representation:

$$\begin{aligned} w_\ell &= \langle y, \gamma^{(y_\mathcal{P})}\rangle \oplus \langle \tilde{y}, \gamma^{(\tilde{y}_\mathcal{P})}\rangle \\ &= \langle c, \gamma^{(y_\mathcal{P})}\rangle \oplus \langle \tilde{c}, \gamma^{(\tilde{y}_\mathcal{P})}\rangle \oplus \langle k, \gamma^{(y_\mathcal{P})} \oplus \gamma^{(\tilde{y}_\mathcal{P})}\rangle. \end{aligned}$$

We need not only $k_\mathcal{P}$ but also $\langle k, \gamma^{(y_\mathcal{P})} \oplus \gamma^{(\tilde{y}_\mathcal{P})}\rangle$ to compute w_ℓ. Let $k_\mathcal{R}$ denote involved key bits, and the size is $\dim W$, where a linear subspace W is defined by $W := \mathrm{Span}\{\gamma^{(p_i)} \oplus \gamma^{(p_j)} \mid i, j \in \{1, \ldots, s\}\}$. The trivial procedure would require guessing $k_\mathcal{P}$ and $k_\mathcal{R}$ for every pair, for a time complexity of $2N \times 2^{n_\mathcal{P} + \dim W}$, where $n_\mathcal{P}$ is the bit length of $k_\mathcal{P}$.

In this section, we introduce a more advanced procedure, where the Fast Walsh-Hadamard Transform (FWHT) is applied instead of guessing $k_\mathcal{R}$ for every pair. As a result, the time complexity is reduced to $2^{n_\mathcal{P}}(2N + \dim W \cdot 2^{\dim W})$.

8.1 Using the FWHT for Key Recovery

The first step in the key recovery procedure is guessing $k_{\mathcal{P}}$ to identify partitions. Once we have guessed these key bits, the first term of the LLR statistic, i.e., $\alpha \coloneqq \frac{1}{2}\sum_{\ell=1}^{N} \ln\left(1 - C_\ell^2\right)$, is constant and independent of w_ℓ. Thus, in order to determine the LLR, we compute $\mathrm{LLR}' = \mathrm{LLR} - \alpha$ as

$$\begin{aligned}\mathrm{LLR}' &= \frac{1}{2}\sum_{\ell=1}^{N} \ln\left(\frac{1-C_\ell}{1+C_\ell}\right)(-1)^{w_\ell} \\ &= \frac{1}{2}\sum_{\ell=1}^{N} \ln\left(\frac{1-C_\ell}{1+C_\ell}\right)(-1)^{\langle c,\gamma^{(y_{\mathcal{P}})}\rangle \oplus \langle \tilde{c},\gamma^{(\tilde{y}_{\mathcal{P}})}\rangle} \times (-1)^{\langle k,\gamma^{(y_{\mathcal{P}})}\oplus\gamma^{(\tilde{y}_{\mathcal{P}})}\rangle}.\end{aligned}$$

Then, by using an array β, whose element $\beta(\gamma)$ is defined as

$$\beta(\gamma) \coloneqq \frac{1}{2}\sum_{\ell=1:\gamma=\gamma^{(y_{\mathcal{P}})}\oplus\gamma^{(\tilde{y}_{\mathcal{P}})}}^{N} \ln\left(\frac{1-C_\ell}{1+C_\ell}\right)(-1)^{\langle c,\gamma^{(y_{\mathcal{P}})}\rangle \oplus \langle \tilde{c},\gamma^{(\tilde{y}_{\mathcal{P}})}\rangle},$$

LLR' is computed as

$$\mathrm{LLR}' = \sum_{\gamma\in W} \beta(\gamma) \times (-1)^{\langle k,\gamma\rangle}.$$

Given a real-valued function $f\colon \mathbb{F}_2^n \to \mathbb{R}$, the *Walsh-Hadamard transform* evaluates the function

$$\widehat{f}\colon \mathbb{F}_2^n \to \mathbb{R}, \quad x \mapsto \sum_{y\in\mathbb{F}_2^n} f(y) \times (-1)^{\langle x,y\rangle}.$$

A naive computation needs $\mathcal{O}(2^{2n})$ steps (additions and evaluations of f), i.e., for each $x \in \mathbb{F}_2^n$, we compute $(-1)^{\langle x,y\rangle} f(y)$ for each $y \in \mathbb{F}_2^n$. The *Fast Walsh-Hadamard transform* is a well-known recursive divide-and-conquer algorithm that evaluates the Walsh-Hadamard transform in $\mathcal{O}(n2^n)$ steps. We refer to e.g., [38, Section 2.3] for the details.

Algorithm 2 shows the attack procedure using the FWHT. We first collect N ciphertext pairs, and therefore, it needs $2N$ queries to E as the data complexity. We next guess $k_{\mathcal{P}}$ and prepare a real number α and the array of real numbers β to compute the LLR statistic. For every stored ciphertext pair, we identify partitions, get corresponding correlation $\rho_{i,j}$ and linear mask γ, and update α and β accordingly. We finally apply the FWHT to β and the LLR statistics are computed as $\alpha + \hat{\beta}(k_{\mathcal{L}})$. The overall running time can be estimated as $2^{n_{\mathcal{P}}}(2N + \dim W \cdot 2^{\dim W})$.

8.2 Success Probability of Algorithm 2

In the following, we assume that the distributions involved can be well estimated by normal approximations. This significantly simplifies the analysis.

Algorithm 2 Key-recovery

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

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

for $\ell \in \{1, \ldots, N\}$ **do**
  $x \xleftarrow{\$} \mathcal{U} \oplus a$
  $(c(\ell), \tilde{c}(\ell)) \leftarrow (E(x), E(x \oplus \Delta_{\text{in}}))$
end for
for $k'_{\mathcal{P}} \in \mathcal{P}$ **do**
  $\alpha \leftarrow 0$
  for $\gamma \in W$ **do**
    $\beta(\gamma) \leftarrow 0$
  end for
  for $\ell \in \{1, \ldots, N\}$ **do**
    $(c, \tilde{c}) \leftarrow (c(\ell), \tilde{c}(\ell))$
    Compute $y_{\mathcal{P}} = c_{\mathcal{P}} \oplus k'_{\mathcal{P}}$ and $\tilde{y}_{\mathcal{P}} = \tilde{c}_{\mathcal{P}} \oplus k'_{\mathcal{P}}$ to identify partitions.
    Identify $\mathcal{T}_i \times \mathcal{T}_j$ for $(y_{\mathcal{P}}, \tilde{y}_{\mathcal{P}})$ and get corresponding correlation $\rho_{i,j}$.
    $\gamma \leftarrow \gamma^{(y_{\mathcal{P}})} \oplus \gamma^{(\tilde{y}_{\mathcal{P}})}$
    $\alpha \leftarrow \alpha + \frac{1}{2} \ln(1 - \rho_{i,j}^2)$
    $\beta(\gamma) \leftarrow \beta(\gamma) + \frac{1}{2} \ln\left(\frac{1-\rho_{i,j}}{1+\rho_{i,j}}\right) (-1)^{\langle c, \gamma^{(y_{\mathcal{P}})} \rangle \oplus \langle \tilde{c}, \gamma^{(\tilde{y}_{\mathcal{P}})} \rangle}$
  end for
  Compute $\hat{\beta}$ by using the Fast Walsh-Hadamard Transform.
  $\mathcal{C}(k'_{\mathcal{P}}, k'_{\mathcal{R}}) \leftarrow \alpha + \hat{\beta}(k'_{\mathcal{R}})$
  if $\mathcal{C}(k'_{\mathcal{P}}, k'_{\mathcal{R}}) > \Theta$ **then**
    Save $(k'_{\mathcal{P}}, k'_{\mathcal{R}})$ as a key candidate.
  end if
end for

Note that we opted for a rather simple statistical model ignoring, in particular, the effect of the wrong key distribution by assuming the simple randomization hypothesis and ignoring the way we sample our plaintexts (i.e. known vs. chosen vs. distinct plaintext). Those effects might have major impact on the performance of attacks when the data complexity is close to the full code-book and the success probability and the gain are limited. However, none of this is the case for our parameters. In our concrete applications, we have verified the behavior experimentally wherever possible.

For the statistical model for the right key, this implies that the counter can be expected to approximately follow a normal distribution with parameters

$$\mathcal{C}(k_{\mathcal{P}}, k_{\mathcal{L}}) \sim \mathcal{N}\left(\frac{N}{2}\overline{C}, N\overline{C}\right),$$

where $\overline{C} = \frac{1}{N}\sum_{\ell=1}^{N} C_\ell^2$. The wrong key counters (under the simple randomization hypothesis) are approximately normally distributed with parameters

$$\mathcal{C}(k'_{\mathcal{P}}, k'_{\mathcal{L}}) \sim \mathcal{N}\left(-\frac{N}{2}\overline{C}, N\overline{C}\right).$$

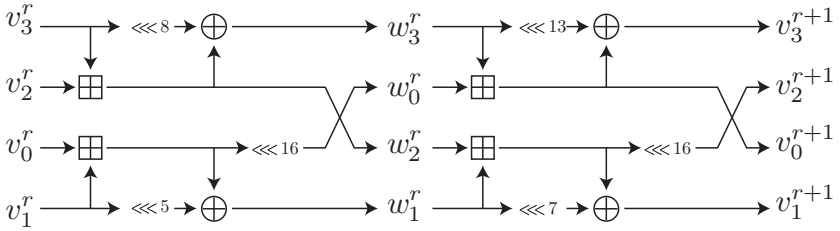


Fig. 7 The round function of Chaskey.

With this, we can deduce the following proposition.

Proposition 5 *After running Algorithm 2 for* p^{-1} *times, the probability that the correct key is among the key candidates is*

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

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

9 Application to Chaskey

Chaskey [10] is a lightweight MAC algorithm whose underlying primitive is an ARX-based permutation in an Even-Mansour construction, i.e., Chaskey-EM. The permutation operates on four 32-bit words, i.e., the block size is 128 bits. In the version proposed in [10], the permutation employs 8 rounds of the form depicted in Fig. 7. In [40], the author proposed a version with an increased number of rounds (from 8 to 12), and this version is currently standardized in ISO/IEC 29192-6:2019. The designers claim security up to 2^{80} computations as long as the data is limited to 2^{48} blocks.

Let $(v_0^r, v_1^r, v_2^r, v_3^r)$ be the input of the rth round function, and $(w_0^r, w_1^r, w_2^r, w_3^r)$ denotes the state after applying the half round for $(v_0^r, v_1^r, v_2^r, v_3^r)$. Please refer to Fig. 7 for each index of the branches.

9.1 Overview of Our Attack

We first show the high-level overview of our attack. Similar to the previous differential-linear attack from [19], we first divide the cipher into three sub ciphers. For the 7-round attack, we use E_1 covering 1.5 rounds, E_m covering 4 rounds, and E_2 covering 0.5 rounds, and the key-recovery, i.e., the function F, is covering 1 round. We also present a 7.5-round attack, where E_2 covers 1 round instead of 0.5 rounds.

The differential characteristic and the linear trail are applied to E_1 and E_2, respectively, while the experimental differential-linear distinguisher is applied to the middle part E_m. Note that, since the differential-linear distinguisher over E_m is constructed experimentally, its absolute correlation must be high enough

Table 2 Probability that adding one basis element 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

to be detectable by using a relatively small sampling space. Moreover, since it is practically infeasible to check *all* input differences and *all* output linear masks, we restricted ourselves to the case of an input difference of Hamming weight 1 and linear masks of the form $[i]$ or $[i, i+1]$, i.e., 1-bit or consecutive 2-bit linear masks. As a result, when there is a non-zero difference *only* in the 31st bit (msb) of w_0^1, i.e., $\Delta_m = (([]), ([]), ([31]), ([]))$, we observed the following two differential-linear distinguishers with correlations $2^{-5.1}$:

$$\mathrm{Aut}_{E_m}(\Delta_m, ([], [], [20], []), ([], [], [20], [])) \approx 2^{-5.1}, \tag{7}$$

$$\mathrm{Aut}_{E_m}(\Delta_m, ([], [], [20, 19], []), ([], [], [20, 19], [])) \approx 2^{-5.1}. \tag{8}$$

These correlations[4] are estimated using a set consisting of 2^{26} random samples of w^1. This is significant enough since the standard deviation assuming a normal distribution is 2^{13}. Note that we do not focus on the theoretical justification of this 4-round experimental differential-linear distinguisher in this paper and we start the analysis for E_1 and E_2 from the following subsection.

9.2 Using Conditional Differentials

Before we discuss the improved basis we have found using the conditional differential technique, we first recall the basis of $\mathcal{U}$ introduced in [1] (see the first two-row blocks in Table 2). Here, the threshold of the probability is relaxed from 0.95 to 0.93, and $v_3[30]$ (in red) is newly added in the basis. The conditional differential techniques provide us with four other basis elements with probability 1, which cannot be found by Algorithm 1 [1] (see the third-row block in Table 2). A linear subspace $\mathcal{U}$, formed by elements that don't affect the output with probability 1, and whose dimension is $18 + 4 = 22$ is finally used to attack 7-round Chaskey. In addition, all, i.e., $18 + 8 + 4 = 30$, basis elements are used to attack 7.5-round Chaskey.

[4]The first case is exactly the one shown in [19], but its correlation was reported as $2^{-6.1}$. We are not sure about the reason for this gap, but we think that $2^{-6.1}$ refers to the bias instead of the correlation.

In Appendix C, we provide the details on how to obtain these relations. We use the conditional differential framework and Fig. 18 in order to recover for free the value of some keybits and also to find additional bits of information for sampling and increase the dimension of $\mathcal{U}$ from 18 as given in [1] (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.

9.2.1 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 $k_2[26] \oplus k_3[26]$, $k_2[21] \oplus k_3[21]$, $k_2[26] \oplus k_3[27]$. Note that these techniques can be used after mounting concrete attacks shown in Sects. 9.3 and 9.4. Thus, this does not contribute to accelerating our key-recovery attacks.

9.2.2 Additional Space for Sampling

Compared with the linear subspace $\mathcal{U}$ shown in [1], 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 Sect. 5.3, and some more detailed explanations can be found in Appendix C for the interested reader using Fig. 18. We summarize this in the following lemma:

Lemma 6 *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* $E_1(x \oplus u) \oplus E_1(x \oplus u \oplus \Delta_{\text{in}}) = \Delta_m$*, 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 will not be used in the attack.

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

Lemma 7 *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* $E_1(x \oplus u) \oplus E_1(x \oplus u \oplus \Delta_{\text{in}}) = \Delta_m$ *where* E_1 *denotes* 1.5 *rounds of Chaskey.*

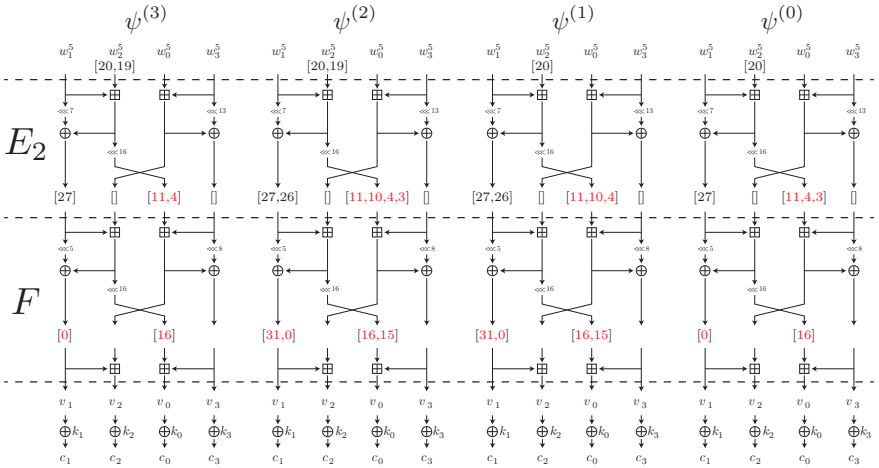


Fig. 8 Four 0.5-round linear trails for the 7-round attack.

We can build a 30-dimensional linear subspace such that all its elements verify simultaneously the differential with probability $2^{-17.28}$. For this, we consider the 22-dimensional linear subspace of Lemma 6 and add to its basis the 7 vectors from [1] and $v_3[30]$. Our 7.5-round attack uses this linear subspace.

9.3 The 7-Round Attack

9.3.1 List of Differential-Linear Distinguishers

As shown in Equations (7) and (8), we have two differential-linear distinguishers with correlations $2^{-5.1}$, where two output linear masks, $([],[],[20],[])$ and $([],[],[20,19],[])$, are available. By extending $([],[],[20],[])$ and $([],[],[20,19],[])$ by 0.5 rounds, respectively, we can get four linear masks (see Fig. 8). 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 absolute correlation is relatively high but lower than $2^{-6.43}$. Table 3 summarizes 12 output masks. For any $(i,j) \in \{0,1\} \times \{0,1\}$ and $(i,j) \in \{2,3\} \times \{2,3\}$, the correlations of the differential-linear distinguishers are estimated by the combination of two output masks as follows:

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

where $\mathrm{Aut}(\beta^{(i)}, \beta^{(j)}) = \mathrm{Aut}_{E_2 \circ E_m}(\Delta_m, \beta^{(i)}, \beta^{(j)})$ and $\delta_h^{(i)} \in \{1, -1\}$ is defined by the δ column in Table 3. Each correlation is estimated by using 2^{35} pairs.

Table 3 List of output linear masks after 6 rounds.

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

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:** $v_3[12], v_1[4], v_2[4], v_2[3], v_2[2], v_2[1], v_1[12], v_2[12], v_2[11], v_2[10], v_2[9]$

Considering that the lowest correlation is $2^{-11.06}$, an estimation with 2^{35} pairs is reliable enough. These differential-linear distinguishers are finally used to estimate the theoretical correlation by considering the auto-correlation-linear hull.

9.3.2 Theoretical Correlations with Auto-Correlation-Linear Hull

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 the linear trail $\psi^{(3)}$ is used for both texts in a pair. When $\beta_0^{(3)}$ is used, available linear masks and corresponding correlation are 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. 16 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, the 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 [15].

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 are 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, the 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

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

We now take the auto-correlation-linear hull 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

$$\widehat{F^{-1}|_{\mathcal{T}_{p_i}}}(\gamma^{p_i}, \beta_1^{(3)}) \times \widehat{F^{-1}|_{\mathcal{T}_{p_j}}}(\gamma^{p_j}, \beta_0^{(3)}) \times \mathrm{Aut}_{E_2 \circ E_m}(\Delta_m, \beta_1^{(3)}, \beta_0^{(3)})$$

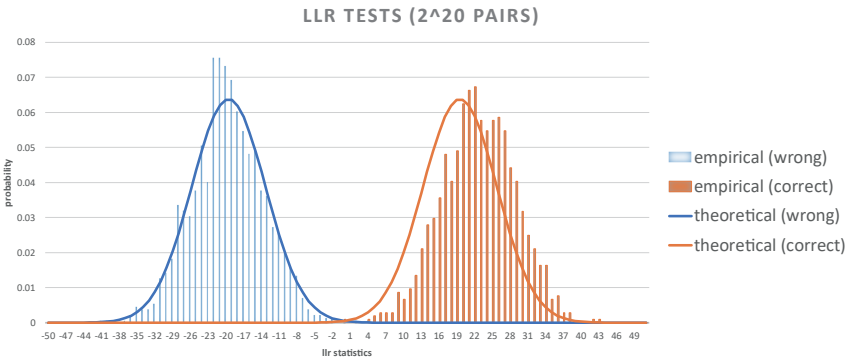


Fig. 9 Comparison with LLR statistics to attack 7-Round Chaskey.

$$2^{-0.677} \times 2^{-0.263} \times (\delta_1^{(3)} \times \delta_0^{(3)} \times 2^{-7.70}) = -2^{-8.64}.$$

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

9.3.3 Experimental Reports

The absolute correlation of each partition is high enough so that we experimentally verify our attack procedure. In our experiments, we used the right pair and the correct key to observe the LLR statistic for the correct case. On the other hand, the right pair is not used for the wrong case.

The LLR statistic depends on the sum of the squared correlation $N\overline{C} = \sum_{\ell=1}^{N} c_\ell^2$. We estimated $\overline{C} \approx 2^{-14.711}$, and $N\overline{C} \approx 39.1$ 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 $N\overline{C}/2$ (for a correct case) and $-N\overline{C}/2$ (for wrong case) and the standard deviation $\sqrt{N\overline{C}}$. By repeating our attack procedure 1024 times, two experimental histograms are drawn (see Fig. 9). 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 auto-correlation-linear hull 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.

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
$\psi^{(4)}$	$\beta_0^{(4)} = ([16,15],[31,0],[19,12,11,4],[19,12])$	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

As a result, the dimension of W is 10. Because of Lemma 6, 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, we have $N\overline{C} \approx 78.2$. 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:

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

9.4 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 absolute correlation is slightly lower. Table 5 shows 24 such output masks. For any $(i, j) \in \{0, 1, 2, 3\} \times \{0, 1, 2, 3\}$ and $(i, j) \in \{4, 5, 6, 7\} \times \{4, 5, 6, 7\}$, the correlations of the differential-linear distinguishers are estimated by the combination of two output masks as follows:

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

where $\mathrm{Aut}(\beta^{(i)}, \beta^{(j)}) = \mathrm{Aut}_{E_2 \circ E_m}(\Delta_m, \beta^{(i)}, \beta^{(j)})$ and $\delta_h^{(i)} \in \{1, -1\}$ is defined by the δ column in Table 5. These correlations are estimated by using 2^{40} pairs. Considering the lowest absolute correlation is $2^{-14.6}$, an 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 $s^R = k_2 \oplus k_3$.

9.4.1 Experimental Reports

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

We estimated $\overline{C} \approx 2^{-24.37}$, and $N\overline{C} \approx 12.38$ when $N = 2^{28}$ pairs are used. Figure 10 shows the comparison of the LLR statistics, where the theoretical distribution is drawn by the normal distribution with mean $N\overline{C}/2$ (for a correct case) and $-N\overline{C}/2$ (for wrong case) and the standard deviation $\sqrt{N\overline{C}}$. By repeating our attack procedure 256 times, two experimental histograms

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

$\boldsymbol{\zeta_1}$ $(v_2^7[23], v_0^7[23, 22])$	$\mathcal{P}_1 \ni p_i \cong (s^R[22], s^R[21], s^R[20], s^R[19], s^R[18])$ **Linear:** $v_3[23], v_2[23], v_2[22], v_2[21], v_2[20], v_2[19], v_2[18], v_2[17]$
$\boldsymbol{\zeta_2}$ $(v_0^7[25], v_0^7[25, 24])$	$\mathcal{P}_2 \ni p_i \cong (s^L[24], s^L[23])$ **Linear:** $v_1[25], v_0[25], v_0[24], v_0[23], v_0[22]$
$\boldsymbol{\zeta_3}$ $(w_0^6[16], v_0^7[16, 15])$	$\mathcal{P}_3 \ni p_i \cong (v_3[28] \oplus v_0[27, 14], s^L[15], s^L[14], s^L[28], s^L[27])$ **Linear:** $v_3[29], v_1[16], v_0[16], v_0[15], v_0[14], v_0[13], v_1[29], v_0[29], v_0[28], v_0[27], v_0[26]$
$\boldsymbol{\zeta_4}$ $(w_2^6[19], v_0^7[19, 18])$	$\mathcal{P}_4 \ni p_i \cong (v_1[25] \oplus v_2[8, 1], s^R[2], s^R[1], s^R[9], s^R[8])$ **Linear:** $v_1[26], v_3[3], v_2[3], v_2[2], v_2[1], v_3[10], v_2[10], v_2[9], v_2[8], v_2[7]$
$\boldsymbol{\zeta_5}$ $(w_2^6[12], v_0^7[12, 11])$	$\mathcal{P}_5 \ni p_i \cong (v_1[18] \oplus v_2[26, 1], s^R[27], s^R[26], s^R[2], s^R[1])$ **Linear:** $v_1[19], v_3[28], v_2[28], v_2[27], v_2[26], v_2[25], v_3[3], v_2[3], v_2[2], v_2[1])$
$\boldsymbol{\zeta_6}$ $(w_2^6[4], v_0^7[4, 3])$	$\mathcal{P}_6 \ni p_i \cong (v_1[10] \oplus v_2[25, 18], s^R[19], s^R[18], s^R[26], s^R[25])$ **Linear:** $v_1[11], v_3[20], v_2[20], v_2[19], v_2[18], v_2[17], v_3[27], v_2[27], v_2[26], v_2[25], v_2[24]$
$\boldsymbol{\zeta_7}$ $(v_0^7[31])$	$\mathcal{P}_7 \ni p_i \cong (s^L[30], s^L[29], s^L[28], s^L[27])$ **Linear:** $v_1[31], v_0[31], v_0[30], v_0[29], v_0[28], v_0[27], v_0[26]$

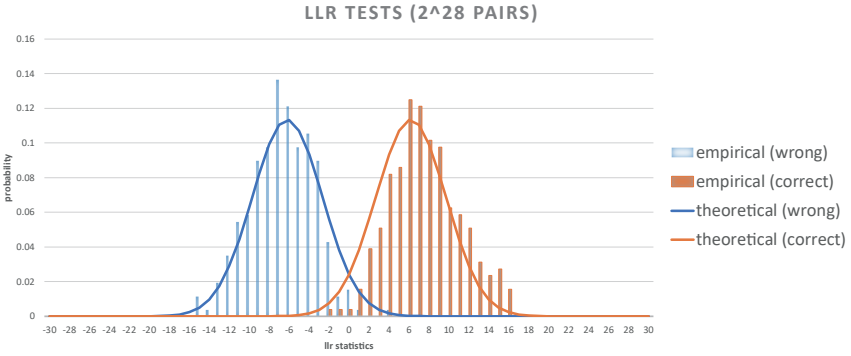


Fig. 10 Comparison with LLR statistics to attack 7.5-Round Chaskey.

are drawn. Similar to the 7-round attack, a slight gap between the theoretical distribution and experimental histogram is observed in the correct case. We again expect that the reason comes from the other auto-correlation-linear hull that we do not consider. 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 7. 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, $N\overline{C} \approx 40.78$. With a success probability of 90%, we can construct a 22.5-bit filter, which is insufficient 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

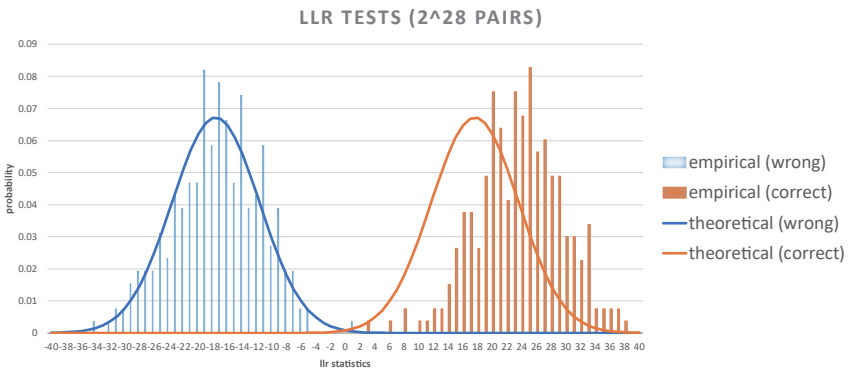


Fig. 11 Comparison with LLR statistics to attack 7.5-Round Chaskey when multiple linear masks are used every partition.

estimate the time complexity as

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

9.4.2 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 the 7.5-round attack, there are $2 \times 4 \times 4 = 32$ linear approximations, and we choose only one approximation with the highest absolute correlation. However, why do we 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 assume that each approximation is independent. Fortunately, since our attack can be verified experimentally, we implemented our attack under this controversial assumption.

We estimated $\overline{C} \approx 2^{-22.86}$. When $N = 2^{28}$ pairs are used, $N\overline{C} \approx 35.26$, which increases from 12.38. Figure 11 shows the comparison of the LLR statistics, where the theoretical distribution is drawn by the normal distribution with mean $N\overline{C}/2$ (for a correct case) and $-N\overline{C}/2$ (for wrong case) and the standard deviation $\sqrt{N\overline{C}}$. By repeating our attack procedure 256 times, two experimental histograms are drawn. Despite the controversial assumption, our theoretical estimation can simulate the experimental result nicely. Therefore, for the application to 7.5-round Chaskey, we conclude that using 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, $N\overline{C} \approx 116.16$. With a success probability of 90%, we can construct a 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

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

10 Application to ChaCha

The internal state of ChaCha is represented by a 4×4 matrix whose elements are 32-bit vectors. In this section, the input state for the rth round function is represented as

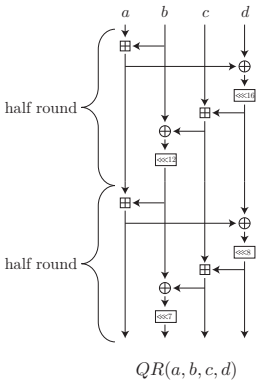
$$\begin{pmatrix} v_0^r & v_1^r & v_2^r & v_3^r \\ v_4^r & v_5^r & v_6^r & v_7^r \\ v_8^r & v_9^r & v_{10}^r & v_{11}^r \\ v_{12}^r & v_{13}^r & v_{14}^r & v_{15}^r \end{pmatrix}.$$

The QR (an abbreviation for *quarterround*) function is applied in odd and even rounds on every column and diagonal, respectively. We also introduce the notion of a *half round*, in which the QR function is divided into two subfunctions depicted in Fig. 12. Let w^r be the internal state after applying a half round on v^r. Moreover, we use the term *branches* for a, b, c and d, as shown in Fig. 12.

In the initial state of ChaCha, a 128-bit constant is loaded into the first row, a 128- or 256-bit secret key is loaded into the second and third rows, and a 64-bit counter and 64-bit nonce are loaded into the fourth row. In other words, the first three rows in v^0 are fixed. For r-round ChaCha, the odd and even round functions are iteratively applied, and the feed-forward values $v_i^0 \boxplus v_i^r$ is given as the key stream for all i. Note that we can compute v_i^r for $i \in \{0, 1, 2, 3, 12, 13, 14, 15\}$ because corresponding v_i^0 is known.

10.1 Overview of Our Attack

We use the same attack strategy as for Chaskey. The cipher is divided into the sub ciphers E_1 covering 1 round, E_m covering 2.5 rounds, and E_2 covering 1.5 rounds to attack 6 rounds, where the key recovery is applied the last single round (F). One difference to Chaskey is the domain space that the attacker can control. In particular, we cannot control branches a, b, and c because fixed constants and the fixed secret key are loaded into these states. Thus, only branch d can be varied. It implies that active bit positions for input differences are limited to branch d, and a difference Δ_m with Hamming weight 1 after E_1 will not be available due to the property of the round function. Therefore, we first need to generate consistent Δ_m whose Hamming weight is minimized.



$$(v_0^{r+1}, v_4^{r+1}, v_8^{r+1}, v_{12}^{r+1}) = QR(v_0^r, v_4^r, v_8^r, v_{12}^r)$$
$$(v_1^{r+1}, v_5^{r+1}, v_9^{r+1}, v_{13}^{r+1}) = QR(v_1^r, v_5^r, v_9^r, v_{13}^r)$$
$$(v_2^{r+1}, v_6^{r+1}, v_{10}^{r+1}, v_{14}^{r+1}) = QR(v_2^r, v_6^r, v_{10}^r, v_{14}^r)$$
$$(v_3^{r+1}, v_7^{r+1}, v_{11}^{r+1}, v_{15}^{r+1}) = QR(v_3^r, v_7^r, v_{11}^r, v_{15}^r)$$

$$(v_0^{r+2}, v_5^{r+2}, v_{10}^{r+2}, v_{15}^{r+2}) = QR(v_0^{r+1}, v_5^{r+1}, v_{10}^{r+1}, v_{15}^{r+1})$$
$$(v_1^{r+2}, v_6^{r+2}, v_{11}^{r+2}, v_{12}^{r+2}) = QR(v_1^{r+1}, v_6^{r+1}, v_{11}^{r+1}, v_{12}^{r+1})$$
$$(v_2^{r+2}, v_7^{r+2}, v_8^{r+2}, v_{13}^{r+2}) = QR(v_2^{r+1}, v_7^{r+1}, v_8^{r+1}, v_{13}^{r+1})$$
$$(v_3^{r+2}, v_4^{r+2}, v_9^{r+2}, v_{14}^{r+2}) = QR(v_3^{r+1}, v_4^{r+1}, v_9^{r+1}, v_{14}^{r+1})$$

Fig. 12 The odd and even round functions of ChaCha.

The following shows such differential characteristics over one QR function:

$$\Delta_{\text{in}} = (([]), ([]), ([]), ([i])) \;\rightarrow\; \Delta_m = (([i+28]), ([i+31, i+23, i+11, i+3]), ([i+24, i+16, i+4]), ([i+24, i+4])) \,.$$

The probability that pairs with input difference Δ_{in} satisfy this characteristic is 2^{-5} on average. We discuss the properties of this differential characteristic in Sect. 10.2 in more detail.

We next evaluate an experimental differential-linear distinguisher for the middle part E_m. When the Hamming weight of Γ_m is 1, and the active bit is in the lsb, it allows the absolute correlation of linear trails for E_2 to be lower. For $i = 6$, i.e., $\Delta_m = (([2]), ([5, 29, 17, 9]), ([30, 22, 10]), ([30, 10]))$, we find the following four differential-linear distinguishers:

$$\text{Aut}_{E_m}(\Delta_{m,j}, \alpha_j, \alpha_j) = 2^{-8.3}$$

for $j \in \{0, 1, 2, 3\}$, where $\Delta_{m,j}$ is a difference such that $\Delta(v_j^1, v_{j+4}^1, v_{j+8}^1, v_{j+12}^1) = \Delta_m$ (and other branches are constant), and α_j is a linear mask such that the lsb of the branch $w_{(j+1) \bmod 4}^3$ is 1 (and the others are 0). When this experimental distinguisher is combined with the differential characteristic for E_1, it covers 3.5 rounds with a 1-bit output linear mask Γ_m. This differential-linear distinguisher is improved by 0.5 rounds from the previous distinguisher with 1-bit output linear mask (see [20, 22]).

10.2 Differential Part

The QR function is independently applied to each column in the first round. Therefore, when the output difference of one QR function is restricted by Δ_m, the input of the other three QR functions are trivially independent of the output difference. It implies that we have 96 independent bits, and we can easily amplify the probability of the differential-linear distinguisher. On

the other hand, we face a different problem: the probability of the differential characteristic $(\Delta_{\rm in}, \Delta_m)$ highly depends on the value of the secret key. For example, for $\Delta v_{12}^0[6] = 1$, we expect that there is a pair $(v_{12}^0, v_{12}^0 \oplus \texttt{0x00000020})$ satisfying $\Delta(v_0^1, v_4^1, v_8^1, v_{12}^1) = \Delta_m$, but it depends on the constant v_0^0 and the key values v_4^0 and v_8^0. We cannot find such a pair for 292 out of 1024 randomly generated keys in our experiments. On the other hand, when we can find it, i.e., on 732 out of 1024 keys, the average probability satisfying $\Delta(v_0^1, v_4^1, v_8^1, v_{12}^1) = \Delta_m$ is $2^{-4.5}$. This experiment implies the existence of *"strong keys"* against our attack[5]. However, note that we can vary the columns in which we put a difference, which involves different key values. Since the fraction of "strong keys" is not so high, i.e., 292/1024, we can assume that there is at least one column in which no "strong key" is chosen with very high probability.

To determine the factor p, for 1024 randomly generated keys, we evaluated p^{-1} randomly chosen nonces and counters, where the branch in which we induce the difference is also randomly chosen. As a result, we can find a right pair on 587 keys with $p^{-1} = 2^5$ iterations. Therefore, with $p = 2^{-5}$, we assume that we can find a right pair with a probability of $1/2$ in this stage of the attack.

In the following, we explain our attack for the case that v_{12}^0 is active, $\Delta(v_0^1, v_4^1, v_8^1, v_{12}^1) = \Delta_m$. Note that the analysis for the other three cases follows the same argument.

10.3 Linear Part for the 6-Round Attack

To attack 6-round ChaCha, we first construct a 5-round differential-linear distinguisher, where 1.5-round linear trails are appended (i.e. the E_2 part) to the 3.5-round experimental differential-linear distinguisher from the previous section. We have two 1.5-round linear trails given by

$$\mathbf{Cor}[w_1^3[0] \oplus \psi^{(1)}] = 2^{-1}\ , \qquad \mathbf{Cor}[w_1^3[0] \oplus \psi^{(0)}] = -2^{-1}\ ,$$

where $\psi^{(1)} = \psi \oplus v_{10}^5[6]$ and $\psi^{(0)} = \psi \oplus v_{14}^5[6]$, and

$$\begin{aligned}\psi = (v_5^5[19,7] \oplus v_{10}^5[19,7] \oplus v_{15}^5[8,0]) \oplus (v_1^5[0] \oplus v_6^5[26] \oplus v_{11}^5[0]) \\ \oplus (v_{13}^5[0]) \oplus (v_3^5[0] \oplus v_9^5[12] \oplus v_{14}^5[7])\ .\end{aligned}$$

Figure 13 shows the two 1.5-round linear trails. Since their correlations are $\pm 2^{-1}$, we have 2×2 differential-linear distinguishers on 5 rounds whose correlations are $\pm 2^{-10.3}$. Note that the sign of each correlation is deterministic according to the output linear mask.

10.4 Key Recovery for the 6-Round Attack

Our 6-round attack uses these 5-round differential-linear distinguishers, and the 1-round key recovery is shown in Fig. 14. Let $\vec{c} = (c_0, \ldots, c_{15})$ be the

[5]The theoretical justification is discussed in [27] after the proposal of our original paper [1].

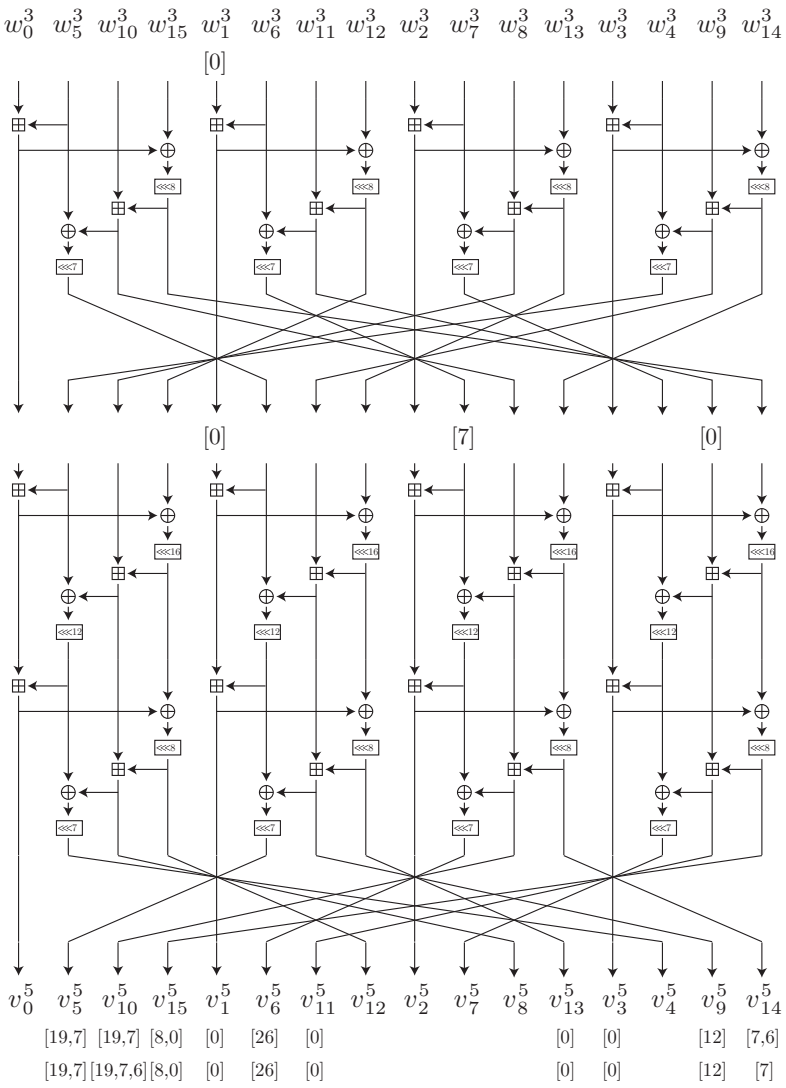


Fig. 13 Two linear trails for 1.5-round ChaCha.

corresponding output, and let $\vec{v} = (v_0, \ldots, v_{15})$ be the sixteen 32-bit values before the secret key is added. Note that the secret key is only added with half of the state and public values are added with the other state. Therefore, we simply regard $v_i = c_i$ for $i \in \{0, 15, 1, 12, 2, 13, 3, 14\}$.

First, we partially extend two linear masks for the last round to be linearly computed. Figure 14 summarizes the extended linear masks, where we need to compute the bits labeled by a red color. Moreover, for simplicity, we introduce t_0, t_{10}, t_{11}, and t_3 as depicted in Fig. 14.

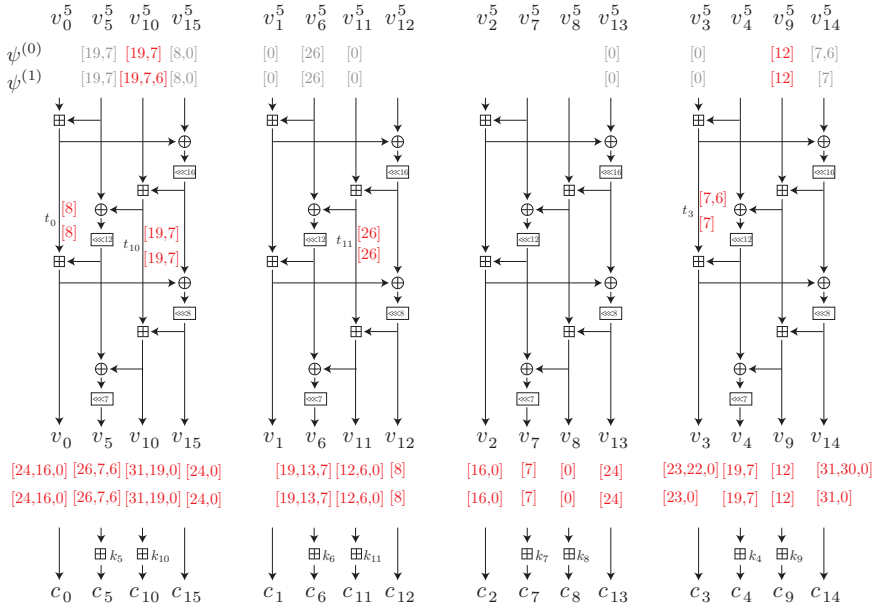


Fig. 14 Key recovery for 6-round ChaCha.

Each bit in $\vec{v}$ to which the secret key is not added can be computed for free. For the other bits, we need to guess some key bits first. We first explain the simple case, i.e., we compute $v_i[j]$ from c_i. As an example, we focus on $v_7[7]$, which involves k_7 nonlinearly. We apply the partition technique to compute this bit. By guessing $k_7[6]$ and $k_7[5]$ (remember that $k_7[7]$ cancels out in the differential-linear approximation), (3/4) data is available with correlation ± 1, and the remaining (1/4) data[6] is available with correlation -2^{-1}. Since $v_i[0]$ is linearly computed by $c_i[0]$, there are 13 simple partition points in which we need to guess key bits. In total, we need to guess a 26-bit key.

Computing bits in $\vec{v}^5$ and $\vec{t}$ is a bit more complicated than the simple case above. For example, let us consider $v_9^5[12]$, and this bit can be computed as

$$\begin{aligned} v_9^5[12] &= (c_9 \boxminus k_9 \boxminus c_{14} \boxminus (c_3 \oplus (v_{14} \ggg 8)))[12] \\ &= ((c_9 \boxminus c_{14} \boxminus (c_3 \oplus (v_{14} \ggg 8))) \boxminus k_9)[12]. \end{aligned}$$

Since we can compute $(c_9 \boxminus c_{14} \boxminus (c_3 \oplus (v_{14} \ggg 8)))$ for free, this case is equivalent to the simple case. We also use this equivalent transformation for t_{10}, t_{11}, and $v_{10}[19]$. In total, we have 6 such partition points, and some partition points can share the same key, e.g., 2-bit key $k_{10}[18]$ and $k_{10}[17]$ is already guessed to compute $v_{10}[19]$. Guessing 4 bits of additional key is enough to compute each

[6] This correlation is estimated originally when the key k_7 changes randomly, but k_7 is a fixed constant. These correlations are much higher or lower according to the fixed key, but on key average, which is the natural attack assumption for symmetric-key ciphers, the average correlation is -2^{-1}.

bit. Since we have two linear masks $\psi^{(0)}$ and $\psi^{(1)}$, we dynamically change an applied linear mask according to the data such that correlations to compute $v_{10}^5[7]/v_{10}^5[7,6]$ become ± 1.

We cannot use the equivalent transformation to compute bits in t_0 and t_3. Then, we further extend this linear mask with correlation 2^{-1}. For example, we have the following approximations

$$t_0[8] \approx v_0[8,7] \oplus v_5[15] \oplus v_{10}[8] \oplus 1, \quad t_0[8] \approx v_0[8] \oplus v_5[15,14] \oplus v_{10}[8,7],$$

for $t_0[8]$ with correlation 2^{-1}, and we can use preferable approximations depending on the data. Namely, we first guess $k_{10}[7]$ and determine which linear approximations are available. Then, we guess $k_5[14]$ and $k_5[13]$ and compute $v_5[15]$ (resp. $v_5[15,14]$). In order words, by guessing 3-bit key, 3/4 data is available with correlation $\pm 2^{-1}$ and 1/4 data is available with correlation $\pm 2^{-2}$. We also use the same technique for $t_3[7]/t_3[7,6]$.

10.4.1 Estimating the Average of the Squared Correlation

Based on the analysis above, we estimate the average of the squared correlation. We suppose each partitioning point is independent when its indicator uses different bits to calculate the average.

We start evaluating the function involving the 1st diagonal.

- The indicator to compute $t_0[8]$ is $(c_5 \oplus k_5)[14]$, $(c_5 \oplus k_5)[13]$, and $(c_{10} \oplus k_{10})[7]$. As discussed before, the average of the squared correlation is $3/4 \times (2^{-1})^2 + 1/4 \times (2^{-2})^2$.
- The indicator to compute $v_5[26]$ is the 25th and 24th bits in $(c_5 \oplus k_5)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_5[7,6]$ is the 6th and 5th bits in $(c_5 \oplus k_5)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_{10}^5[19]$ is the 18th and 17th bits in $((c_{10} \boxminus c_{15} \boxminus (c_0 \oplus (c_{15} \ggg 8))) \oplus k_{10})$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_{10}^5[7]$ (or $v_{10}^5[7,6]$) is the 6th bit in $((c_{10} \boxminus c_{15} \boxminus (c_0 \oplus (c_{15} \ggg 8))) \oplus k_{10})$. Here, we change the applied linear mask according to the observed ciphertext such that correlations become 1. Thus, the average of the squared correlation is 1.
- The indicator to compute $t_{10}[19]$ is the 18th and 17th bits in $((c_{10} \boxminus c_{15}) \oplus k_{10})$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $t_{10}[7]$ is the 6th and 5th bits in $((c_{10} \boxminus c_{15}) \oplus k_{10})$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.

- The indicator to compute $v_{10}[31]$ is the 30th and 29th bits in $(c_{10} \oplus k_{10})$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_{10}[19]$ is the 18th and 17th bits in $(c_{10} \oplus k_{10})$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.

In total, we guess 6 key bits in k_5 and 7 key bits in k_{10}. The average of the squared correlation is the product of these nine partitioning points, i.e., about $2^{-4.396}$.

We similarly evaluate the function involving the 2nd, 3rd, and 4th diagonals.

- The indicator to compute $v_6[19]$ is the 18th and 17th bits in $(c_6 \oplus k_6)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_6[13]$ is the 12th and 11th bits in $(c_6 \oplus k_6)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_6[7]$ is the 6th and 5th bits in $(c_6 \oplus k_6)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $t_{11}[26]$ is the 25th and 24th bits in $((c_{11} \boxminus c_{12}) \oplus k_{11})$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_{11}[12]$ is the 11th and 10th bits in $(c_{11} \oplus k_{11})$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_{11}[6]$ is the 5th and 4th bits in $(c_{11} \oplus k_{11})$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.

For the 2nd diagonal, we guess 6 key bits in k_6 and 6 key bits in k_{11}. The average of the squared correlation is about $2^{-1.797}$.

- The indicator to compute $v_7[7]$ is the 6th and 5th bits in $(c_7 \oplus k_7)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.

For the 3rd diagonal, we guess 2 key bits in k_7. The average of the squared correlation is about $2^{-0.300}$.

- The indicator to compute $t_3[7,6]$ or $t_3[7]$ is $(c_4 \oplus k_4)[13]$, $(c_4 \oplus k_4)[12]$, and $(c_9 \oplus k_9)[6]$. As discussed before, the average of the squared correlation is $3/4 \times (2^{-1})^2 + 1/4 \times (2^{-2})^2$.
- The indicator to compute $v_4[19]$ is the 18th and 17th bits in $(c_4 \oplus k_4)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.

- The indicator to compute $v_4[7]$ is the 6th and 5th bits in $(c_4 \oplus k_4)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_9^5[12]$ is the 11th and 10th bits in $((c_9 \boxminus c_{14} \boxminus (c_3 \oplus (c_{14} \ggg 8))) \oplus k_9)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.
- The indicator to compute $v_9[12]$ is the 11th and 10th bits in $(c_9 \oplus k_9)$. As discussed before, the average of the squared correlation is $3/4 \times (1)^2 + 1/4 \times (2^{-1})^2$.

For the 4th diagonal, we guess 6 key bits in k_4 and 3 key bits in k_9. The average of the squared correlation is about $2^{-3.498}$.

Therefore, we guess 36 key bits in total, and the average of the squared correlation (for the linear part of one of each pair) is $2^{-9.991}$. The average of the squared correlation for the whole approximation is estimated by

$$\overline{C} = (2^{-10.3})^2 \times (2^{-9.991}) \times (2^{-9.991}) \approx 2^{-40.582}.$$

Note that, unlike Chaskey, once these key bits are correctly guessed, all linearly involved bits are either determined or canceled out by XORing another text. It implies $\dim(W) = 0$, and we do not need to proceed with the FWHT.

10.4.2 Data and Time Complexities and Success Probability

Based on the LLR statistic, we estimate the data complexity and the corresponding success probability.

To find a right pair, we repeat Algorithm 2 2^5 times. If we use a right pair and guess the correct key, the LLR statistic follows the normal distribution $\mathcal{N}(\frac{N}{2}\overline{C}, N\overline{C})$ when the correct key is guessed. On the other hand, we assume that it follows $\mathcal{N}(-\frac{N}{2}\overline{C}, N\overline{C})$ for either using a wrong pair or wrong guess.

We need to filter $(5 + 36)$-bit wrong guess by this difference of the normal distributions. By using Proposition 5, the expected number of wrong keys is less than 1 when

$$\Theta \geq \sqrt{N\overline{C}} \times \left(\Phi^{-1}(1 - 2^{-41}) - \frac{N}{2}\overline{C} \right).$$

When we use $N = 2^{47}$ pairs, $\Theta \approx 23.303$ and[7] $p_{\text{success}} = 0.491$. For this success probability, the data complexity is $2^{1+47+5} = 2^{53}$. We guess 2^{36} keys for each texts, the required time complexity is $2^{53+36} = 2^{89}$.

10.5 Another 6-Round Attack

The aforementioned attack is the straightforward application of our attack framework, and it could be optimal considering the data complexity. Interestingly, we have another strategy where less time complexity is possible, although it increases data complexity.[8]

[7] Note that it means that the success probability is $0.491 \times 2 = 0.982$ under the condition that the right pair is successfully obtained during 2^5 iterations.

[8] This is the same attack proposed in our original paper [1].

In the aforementioned attack, we guessed many key bits for each ciphertext. Now, instead of guessing many key bits, we deduce $k_{\mathcal{P}}$ for observed ciphertexts such that the absolute correlations become 1 (except for $t_0[8]$ and $t_3[7]$ or $t_3[7,6]$). For example, to compute $v_5[26]$, we first check $(c_5[25]\|c_5[24])$. When we guess $(k_5[25]\|k_5[24])$ as 00, the indicator is 11 and the absolute correlation is lower than 1. For another guessing, we have representations whose correlation is ± 1. We skip guessing the key as 00 for this ciphertext only to reduce the time complexity.

We have 21 partitioning points, and 3/4 data is available for each point. Only for one point, i.e., $v_{10}^5[7]/v_{10}^5[7,6]$, we do not need to reduce the available data by changing the applied linear masks dynamically. In summary, the fraction of available partitions is $(3/4)^{20} \approx 2^{-8.3}$. Both texts in each pair must belong to an available partition, and the fraction of available pairs is $2^{-16.6}$. The final correlation is $2^{-10.3} \times (2^{-2}) \times (2^{-2}) = 2^{-14.3}$, and the average of the squared correlation is estimated by $\overline{C} = 2^{-28.6}$.

To find a right pair, we repeat Algorithm 2 2^5 times. By using Proposition 5, the expected number of wrong keys is less than 1 when

$$\Theta \geq \sqrt{N^*\overline{C}} \times \left(\Phi^{-1}(1-2^{-41}) - \frac{N^*}{2}\overline{C}\right),$$

where $N^* = N \times 2^{-16.6}$. When we use $N = 2^{52}$ pairs, $N^* = 2^{35.4}$ and $\Theta \approx 19.693$ and[9] $p_{\text{success}} \approx 0.5$. For this success probability, the data complexity is $2^{1+52+5} = 2^{58}$.

On this attack, we do not need to guess 2^{36} keys for all 2^{58} data. On each data, we guess available key bits only. Therefore, the time complexity is estimated as

$$1/p \times (2N + 2N^* \times 2^{n_{\mathcal{P}}}) = 2^5 \times (2^{53} + 2^{36.4} \times 2^{36}) \approx 2^{77.4}.$$

10.6 The 7-Round Attack

Unfortunately, 7-round ChaCha is too complicated to apply our technique to the linear part. On the other hand, thanks to our contribution for the differential part, we find a new differential-linear distinguisher which is improved by 0.5 rounds. Therefore, to confirm the effect of our contribution for the differential part, we use the known technique, i.e., the probabilistic neutral bits (PNB) approach, for the key-recovery attack against 7-round ChaCha. The PNB-based key recovery is a fully experimental approach. We refer to [22] for the details and simply summarize the technique as follows:

- Let the correlation in the forward direction (a.k.a, differential-linear distinguisher) after r rounds be ϵ_d.
- Let n be the number of PNBs given by a correlation γ. Namely, even if we flip one bit in PNBs, we still observe correlation γ.
- Let the correlation in the backward direction, where all PNB bits are fixed to 0 and non-PNB bits are fixed to the correct ones, be ϵ_a.

[9] Note that it means that the success probability is almost 1 under the condition that the right pair is successfully obtained during 2^5 iterations.

Then, the time complexity of the attack is estimated as $2^{256-n}N + 2^{256-\alpha}$, where the data complexity N is given as

$$N = \left(\frac{\sqrt{\alpha \log(4)} + 3\sqrt{1 - \epsilon_a^2 \epsilon_d^2}}{\epsilon_a \epsilon_d} \right)^2,$$

where α is a parameter that the attacker can choose.

In our case, we use a 4-round differential-linear distinguisher with correlation $\epsilon_d = 2^{-8.3}$. Under pairs generated by the technique shown in 10.2, we experimentally estimated the PNBs. With $\gamma = 0.35$, we found 74 PNBs, where non-zero bits of the following bit-vectors represent PNB:

$$\begin{aligned} v_4 &: \texttt{0x00098080} \\ v_5 &: \texttt{0x8CFFE7FC} \\ v_6 &: \texttt{0xF8087FC0} \\ v_7 &: \texttt{0x0000403C} \\ v_8 &: \texttt{0x80000100} \\ v_9 &: \texttt{0xF8198183} \\ v_{10} &: \texttt{0x80700007} \\ v_{11} &: \texttt{0xF8000000}. \end{aligned}$$

Then, the correlation is $\epsilon_a = 2^{-10.6769}$. Then, with $\alpha = 36$, we have $N = 2^{43.83}$ and the time complexity is $2^{225.86}$. Again, since we need to repeat this procedure p^{-1} times, the data and time complexities are $2^{48.83}$ and $2^{230.86}$, respectively.[10]

11 Conclusion and Future Work

We presented new ideas for differential-linear attacks and, particularly, the best attacks on ChaCha,[11] one of the most widely used ciphers in practice, and Chaskey. We hope that our framework finds more applications. In particular, we think that it is a promising future work to investigate other ARX designs for our ideas.

Besides the direct application of our framework to more primitives, our work raises several more fundamental questions. As explained in the experimental verification, we sometimes observe higher LLR statistics than expected, making the attacks more efficient than estimated. The gap would come from

[10] When we estimate ϵ_a, we used the average correlation. When we used the median instead of the average, $\epsilon_a = 2^{-11.1687}$. Then, the data and time complexities are $2^{49.7856}$ and $2^{231.823}$, respectively.

[11] Some follow-up works [25–27, 41] have been proposed after our original proposal [1]. Our attack is still the best for 6-round attack in the context of key recovery. Even for 7 rounds, there have not been follow-up works that essentially improve the complexity yet. On the other hand, Coutinho and Neto presented more efficient distinguishing attacks in [41], and Miyashita, Ito, and Miyaji showed the key-recovery attack on 7.25 rounds in [25].

the difficulty of estimating the accurate correlation of all partitions. Our paper does not solve how to estimate these correlations accurately and efficiently.

Another important open question is in the 7-round attack on ChaCha. We applied the partitioning technique to the 6-round attack. Our result outperforms PNB techniques, which is an experiment-based key recovery technique. Unfortunately, using this technique in 7-round ChaCha is too complicated. A more simple and powerful key recovery procedure exploiting the partitioning technique would beat the PNB-based key recovery like the 6-round attack. However, it is still an open question in our paper.

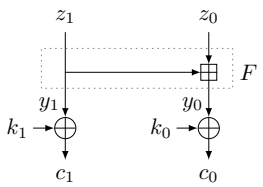
Acknowledgments

We thank the reviewers for their detailed and helpful comments. We further thank Lukas Stennes for checking the application of our framework to ChaCha in a first version of this paper. We also thank Juan del Carmen Grados Vásquez for pointing out the use of the median to evaluate the PNB-based key recovery. This work was partially funded by *Deutsche Forschungsgemeinschaft (DFG)* under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 714294 - acronym QUASYModo).

A Summary of Partitioning

We summarize various partition rules for modular addition. Note that we can verify the correlation of each case experimentally because they have a very high absolute correlation.

A.1 Single Modular Addition

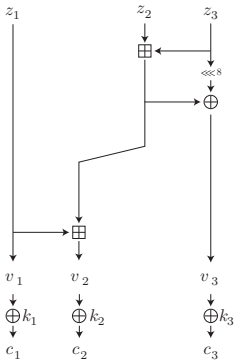


b_0b_1	$z_0[i]$		$z_0[i, i-1]$	
	γ	ε	γ	ε
00	11100	-1	11110	-1
01	11100	-1	11101	-2^{-1}
10	11010	-1	11000	1
11	11001	-2^{-1}	11000	1

Fig. 15 Partitions for a 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. 15), we represent each element of $\mathcal{P}$ as two-bit values b_0b_1, therefore dividing the whole set into four subsets

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



$b_0b_1b_2b_3b_4$	$z_2[11]$		$z_2[11,10]$	
	γ	ε	γ	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. 16 Partition for two consecutive modular additions.

where $s = \bar{y}_1 \oplus y_0$. Note that these partition can be constructed by guessing two bits of key information, i.e., $(k_1 \oplus k_0)[i-1]$ and $(k_1 \oplus k_0)[i-2]$. 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. 15.

A.2 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 7-round Chaskey, is shown in Fig. 16.

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. 16). We split the ciphertext into 2^5 partitions (this time indexed by

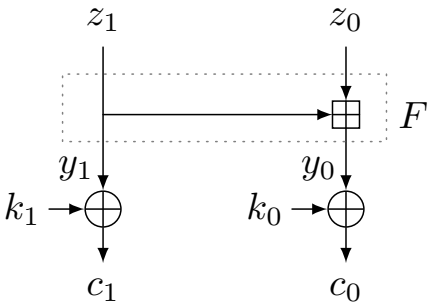


Fig. 17 A simple toy example with a single modular addition.

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

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

where γ is appropriately chosen following Fig. 16. We remark that this new way of partitoning the ciphertexts allows us to find high-absolute-correlation masks for all the 32 partitions, up from the 24 used with the original [1].

B Understanding Partition Points

B.1 A Simple Toy Example

We transfer the above terminology to the simple toy example given in Fig. 17 and already discussed earlier in Sect. 2.2. In this example, for a fixed $i \geq 2$, we want to evaluate $z_0[i]$ or $z_0[i] \oplus z_0[i-1]$ by using the partitioning rules as expressed in Lemma 2 and Lemma 3. For this, we say that $(z_0[i], z_0[i] \oplus z_0[i-1])$ defines a *partition point* ζ. This partition point gives rise to a 2-dimensional subspace $\mathcal{P}$ which can be defined by two parity check equations, i.e., $\mathcal{P}$ is a complement space of the space

$$\mathcal{R} = \{(x_1, x_0) \in \mathbb{F}_2^{2m} \mid x_0[i-1] \oplus \bar{x}_1[i-1] = 0 \text{ and } x_0[i-2] \oplus \bar{x}_1[i-2] = 0\} \ .$$

For example, $\mathcal{P}$ can be chosen as $\{([], []), ([i-1], []), ([i-2], []), ([i-2, i-1], [])\}$.

To demonstrate the attack from the previous section, we split $\mathbb{F}_2^{2m}$ into the direct sum $\mathcal{P} \oplus \mathcal{R}$. By the isomorphism between $\mathcal{P}$ and $\mathbb{F}_2^2$, we can identify

the elements $p \in \mathcal{P}$ by two-bit values $p \cong b_0 b_1$, where b_0 indicates the parity of $x_0[i-1] \oplus \bar{x}_1[i-1]$ and b_1 indicates the parity of $x_0[i-2] \oplus \bar{x}_1[i-2]$. We then consider the following four tuples $(\mathcal{T}_{b_0b_1}, \Gamma_{\text{out}}^{(b_0b_1)}, \gamma^{(b_0b_1)})$ and corresponding $\varepsilon_{b_0b_1}$, whose definition come from the properties presented in Lemma 2 and Lemma 3:

$$\begin{aligned}
&\mathcal{T}_{00} = \mathcal{R} \oplus 00 = \mathcal{S}_{00} \;\; \Gamma_{\text{out}}^{(00)} = ([], [i]) \\
&\gamma^{(00)} = ([i], [i, i-1]) \; \varepsilon_{00} = -1 \\
&\mathcal{T}_{01} = \mathcal{R} \oplus 01 = \mathcal{S}_{01} \;\; \Gamma_{\text{out}}^{(01)} = ([], [i]) \\
&\gamma^{(01)} = ([i], [i, i-1]) \; \varepsilon_{01} = -1 \\
&\mathcal{T}_{10} = \mathcal{R} \oplus 10 = \mathcal{S}_{10} \;\; \Gamma_{\text{out}}^{(10)} = ([], [i]) \\
&\gamma^{(10)} = ([i], [i, i-2]) \; \varepsilon_{10} = -1 \\
&\mathcal{T}_{11} = \mathcal{R} \oplus 11 = \mathcal{S}_{11} \;\; \Gamma_{\text{out}}^{(11)} = ([], [i]) \\
&\gamma^{(11)} = ([i], [i, i-3]) \; \varepsilon_{11} = -2^{-1} \, .
\end{aligned}$$

and

$$\begin{aligned}
&\mathcal{T}_{00} = \mathcal{R} \oplus 00 = \mathcal{S}_{00} \qquad \Gamma_{\text{out}}^{(00)} = ([], [i, i-1]) \\
&\gamma^{(00)} = ([i], [i, i-1, i-2]) \; \varepsilon_{00} = -1 \\
&\mathcal{T}_{01} = \mathcal{R} \oplus 01 = \mathcal{S}_{01} \qquad \Gamma_{\text{out}}^{(01)} = ([], [i, i-1]) \\
&\gamma^{(01)} = ([i], [i, i-1, i-3]) \; \varepsilon_{01} = -2^{-1} \\
&\mathcal{T}_{10} = \mathcal{R} \oplus 10 = \mathcal{S}_{10} \qquad \Gamma_{\text{out}}^{(10)} = ([], [i, i-1]) \\
&\gamma^{(10)} = ([i], [i]) \qquad\qquad\;\; \varepsilon_{10} = 1 \\
&\mathcal{T}_{11} = \mathcal{R} \oplus 11 = \mathcal{S}_{11} \qquad \Gamma_{\text{out}}^{(11)} = ([], [i, i-1]) \\
&\gamma^{(11)} = ([i], [i]) \qquad\qquad\;\; \varepsilon_{11} = 1 \, .
\end{aligned}$$

For example, we give an intuition for the choice of the second tuple when $(y_1, y_0) \in \mathcal{S}_{01}$. Lemma 2 tells us that $\langle([], [i]), (z_1, z_0)\rangle = \langle([i], [i, i-1]), (y_1, y_0)\rangle \oplus 1$, i.e., $\varepsilon_{01} = \mathbf{Cor}_{y \in \mathcal{T}_{01}}[\langle([], [i]), z\rangle \oplus \langle([i], [i, i-1]), y\rangle] = -1$. On the other hand, Lemma 3 tells us that there is no linear representation with absolute correlation 1. Thus, if available, we should use $\Gamma_{\text{out}}^{(01)} = ([], [i])$ for this subset.

We further have

$$\begin{aligned}
W &= \text{Span}\{\gamma^{(a)} \oplus \gamma^{(b)} \mid a, b \in \mathbb{F}_2^2\} \\
&= \{([], []), ([], [i-1]), ([], [i-2]), ([], [i-1, i-2]), \\
&\quad ([], [i-3]), ([], [i-1, i-3]), ([], [i-2, i-3]), ([], [i-1, i-2, i-3])\},
\end{aligned}$$

and we could recover the three bits, $k_0[i-1]$, $k_0[i-2]$, and $k_0[i-3]$, by the last step using the fast Walsh-Hadamard transform.

B.2 Toy Example Using Multiple Partition Points

Let us now look at another example which consists of two branches of the structure depicted in Fig. 17 in parallel, i.e., $(y_3, y_2, y_1, y_0) = (F(z_3, z_2), F(z_1, z_0))$ and $c_i = y_i \oplus k_i$. By using a single partition point as done in the above

example, we can only evaluate the parity of at most two (consecutive) bits of $z = (z_3, z_2, z_1, z_0)$. Instead of just one single partition point, we can also consider multiple partition points. For example, if we want to evaluate the parity involving three non-consecutive bits of $z = (z_3, z_2, z_1, z_0)$, we can use three partition points, i.e.

$$\begin{aligned}\zeta_1 &= (z_0[i], z_0[i] \oplus z_0[i-1]) \ , \\ \zeta_2 &= (z_0[j], z_0[j] \oplus z_0[j-1]) \ , \\ \zeta_3 &= (z_2[\ell], z_2[\ell] \oplus z_2[\ell-1]) \ ,\end{aligned}$$

where $i, j, \ell \geq 3$. In a specific attack, the choice of the partition points depends on the definition of the linear trail. Those partition points give rise to three subspaces $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$, defined by two parity-check equations each, i.e., $\mathcal{P}_i$ is a complement space of $\mathcal{R}_i$, where

$$\begin{aligned}\mathcal{R}_1 &= \{(x_3, x_2, x_1, x_0) \in \mathbb{F}_2^{4m} \,|\, x_0[i-1] \oplus \bar{x}_1[i-1] = 0, x_0[i-2] \oplus \bar{x}_1[i-2] = 0\} \\ \mathcal{R}_2 &= \{(x_3, x_2, x_1, x_0) \in \mathbb{F}_2^{4m} \,|\, x_0[j-1] \oplus \bar{x}_1[j-1] = 0, x_0[j-2] \oplus \bar{x}_1[j-2] = 0\} \\ \mathcal{R}_3 &= \{(x_3, x_2, x_1, x_0) \in \mathbb{F}_2^{4m} \,|\, x_2[\ell-1] \oplus \bar{x}_3[\ell-1] = 0, x_2[\ell-2] \oplus \bar{x}_3[\ell-2] = 0\}.\end{aligned}$$

By defining[12] $\mathcal{P} = \mathcal{P}_1 \oplus \mathcal{P}_2 \oplus \mathcal{P}_3$ and $\mathcal{R}$ to be a complement space of $\mathcal{P}$, we split $\mathbb{F}_2^{4m}$ into the direct sum $\mathcal{P} \oplus \mathcal{R}$.

We can identify the elements $p \in \mathcal{P}$ by $n_{\mathcal{P}}$-bit values $p \cong b_0 b_1 \dots b_{n_{\mathcal{P}}-1}$. We can then again define tuples

$$(\mathcal{T}_{b_0 b_1 \dots b_{n_{\mathcal{P}}-1}}, \Gamma_{\text{out}}^{(b_0 b_1 \dots b_{n_{\mathcal{P}}-1})}, \gamma^{(b_0 b_1 \dots b_{n_{\mathcal{P}}-1})}) \tag{9}$$

by using the properties presented in Lemma 2 and Lemma 3. For example, if $n_{\mathcal{P}} = 6$, we can define

$$\begin{aligned}\mathcal{T}_{010101} = \{(x_3, x_2, x_1, x_0) \in \mathbb{F}_2^{4m} \,|\, & x_0[i-1] \neq x_1[i-1], x_0[i-2] = x_1[i-2], \\ & x_0[j-1] \neq x_1[j-1], x_0[j-2] = x_1[j-2], \\ & x_2[\ell-1] \neq x_3[\ell-1], x_2[\ell-2] = x_3[\ell-2]\} \ ,\end{aligned}$$

$\Gamma_{\text{out}}^{(010101)} = ([], [\ell], [], [i, j])$, $\gamma^{(010101)} = ([\ell], [\ell, \ell-1], [i, j], [i, i-1, j, j-1])$, and $\varepsilon_{010101} = -1$ by using the first case of Lemma 2.

We can also use the three partition points to compute the parity of more than three bits of z. For example, if $n_{\mathcal{P}} = 6$, by using Lemma 2 and 3, we can define

$$\begin{aligned}\mathcal{T}_{001011} = \{(x_3, x_2, x_1, x_0) \in \mathbb{F}_2^{4m} \,|\, & x_0[i-1] \neq x_1[i-1], x_0[i-2] \neq x_1[i-2], \\ & x_0[j-1] = x_1[j-1], x_0[j-2] \neq x_1[j-2], \\ & x_2[\ell-1] = x_3[\ell-1], x_2[\ell-2] = x_3[\ell-2]\} \ ,\end{aligned}$$

[12] Note that $\mathcal{P}$ is not necessarily a *direct* sum of $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$. In other words, the dimension of $\mathcal{P}$ might be smaller than 6, for instance if $i = j$, i.e., $\zeta_1 = \zeta_2$.

and

$$\Gamma_{\text{out}}^{(\texttt{001011})} = ([], [\ell, \ell - 1], [], [i, i - 1, j])$$
$$\gamma^{(\texttt{001011})} = ([\ell], [\ell], [i, j], [i, i - 1, i - 2, j, j - 2]) \ , \quad \varepsilon_{\texttt{001011}} = 1 \ ,$$

which evaluates the parity of five bits of z. Again, several choices for the definition of the tuples in Equation (9) are possible.

B.3 Analysis for Two Consecutive Modular Additions

To avoid the usage of long linear trails and to reduce the data complexity, we may use the partition technique for the more complicated structure of two consecutive modular additions. Inspired by the round function of Chaskey, we consider the case depicted in Fig. 16.

Suppose that we have two partition points, i.e.,

$$\zeta_1 = (z_2[i], z_2[i] \oplus z_2[i - 1]),$$
$$\zeta_2 = (z_3[j], z_3[j, j - 1]),$$

where $i, j \geq 3$. We use the same strategy described in Appendix B.2. Namely, we identify the elements $p \in \mathcal{P}$ by $(5 + 2)$-bit values, where 5-bit and 2-bit indicators come from the partition point ζ_1 and ζ_2, respectively. The applied linear mask and corresponding correlation can be computed as depicted in Figs. 15 and 16.

C Exploiting the Conditions for Finding Chaskey Relations

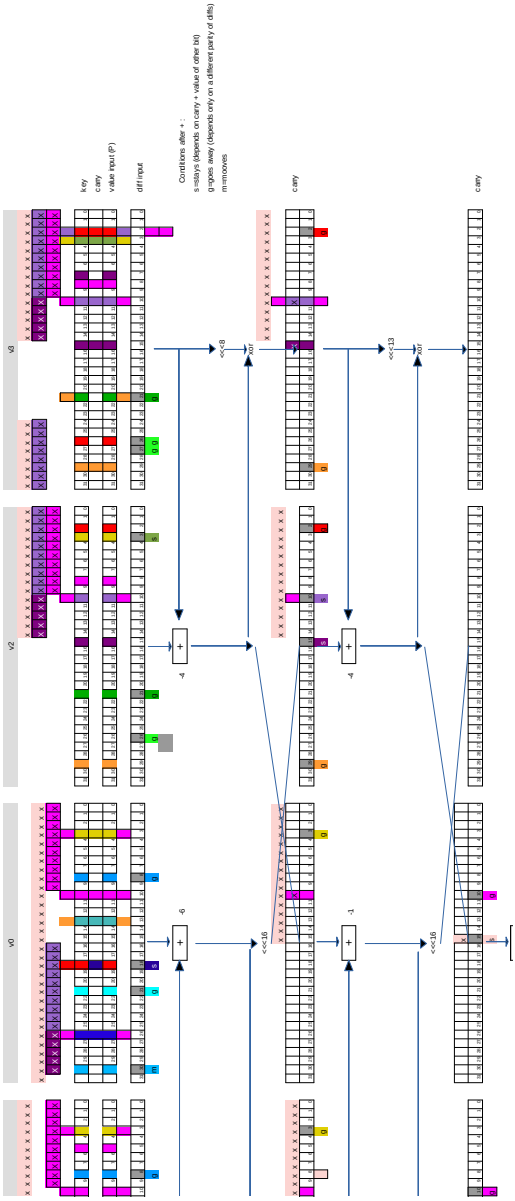


Fig. 18 Conditions on the differential transitions for the 3 first rounds of Chaskey

In Fig. 18, 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 color 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 [1] 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 v_2 and [23,22,20,19,18,17,16] from v_3. The differences are represented in grey. Dependencies in colors. 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 color of the bits with differences 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 v_2 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 v_2 and v_3, but as said in Sect. 5.3, the bits of previous positions to 26 and 27 will not affect this orange carry anymore due to the particular configuration of 26 and 27. The bits provided in [1] that are neutral with very high probability are 20 and 19 from v_1 and 31, 20 and 19 from v_0 and 25 and 24 of v_3.

Let us now see how can we use the conditional differential ideas and Fig. 18 in order to recover for free the value of some keybits and also to find additional bits of information for sampling and increasing the dimension of $\mathcal{U}$ from 18 as given in [1] (and involving exclusively one-bit relations) to 22, or 23 if one-bit relation on the key is known.

Additional space for sampling

Using Fig. 18 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: $v_1[8]$. However, 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 v_1 after 1 round will be known 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 parity of the other orange bit with a 'g': we flip $v_2[29]$ from the first round, that does not 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 these relations as well as exhaustively searched all the ones with weight at most 3, and found that no other such relations exist.

References

[1] 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, Cham (2020)

[2] Broll, M., Canale, F., David, N., Flórez-Gutiérrez, A., Leander, G., Naya-Plasencia, M., Todo, Y.: Further improving differential-linear attacks: Applications to Chaskey and Serpent. IACR Cryptol. ePrint Arch. **2021**, 820 (2021). https://eprint.iacr.org/2021/820

[3] Shimizu, A., Miyaguchi, S.: Fast data encipherment algorithm FEAL. In: Chaum, D., Price, W.L. (eds.) EUROCRYPT '87, Proceedings. LNCS, vol. 304, pp. 267–278. Springer, Berlin, Heidelberg (1987)

[4] Bernstein, D.J.: The Salsa20 family of stream ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 84–97. Springer, Berlin, Heidelberg (2008)

[5] Bernstein, D.J.: ChaCha, a variant of Salsa20 (2008). http://cr.yp.to/chacha.html

[6] Aumasson, J.-P., Henzen, L., Meier, W., Phan, R.C.-W.: SHA-3 proposal Blake. Submission to NIST (2008)

[7] Aumasson, J., Neves, S., Wilcox-O'Hearn, Z., Winnerlein, C.: BLAKE2: simpler, smaller, fast as MD5. In: Jr., M.J.J., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013, Proceedings. LNCS, vol. 7954, pp. 119–135. Springer, Berlin, Heidelberg (2013)

[8] Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., Biryukov, A.: Design strategies for ARX with provable bounds: Sparx and LAX. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Proceedings, Part I. LNCS, vol. 10031, pp. 484–513. Springer, Berlin, Heidelberg (2016)

[9] Beierle, C., Biryukov, A., dos Santos, L.C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q.: Lightweight AEAD and hashing using the Sparkle permutation family. IACR Trans. Symmetric Cryptol. **2020**(S1), 208–261 (2020)

[10] 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, Cham (2014)

[11] Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002, Revised Papers. LNCS, vol. 2365, pp. 112–127. Springer, Berlin, Heidelberg (2002)

[12] Todo, Y., Leander, G., Sasaki, Y.: Nonlinear invariant attack: Practical attack on full SCREAM, iSCREAM, and Midori64. J. Cryptol. **32**(4), 1383–1422 (2019)

[13] Khovratovich, D., Nikolic, I.: Rotational cryptanalysis of ARX. In: Hong, S., Iwata, T. (eds.) FSE 2010, Revised Selected Papers. LNCS, vol. 6147, pp. 333–346. Springer, Berlin, Heidelberg (2010)

[14] Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991)

[15] Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT '93, Proceedings. LNCS, vol. 765, pp. 386–397. Springer, Berlin, Heidelberg (1993)

[16] Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Matsui, M. (ed.) FSE 2001, Revised Papers. LNCS, vol. 2355, pp. 336–350. Springer, Berlin, Heidelberg (2001)

[17] Wallén, J.: Linear approximations of addition modulo 2^n. In: Johansson, T. (ed.) FSE 2003, Revised Papers. LNCS, vol. 2887, pp. 261–273. Springer, Berlin, Heidelberg (2003)

[18] Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Desmedt, Y. (ed.) CRYPTO '94, Proceedings. LNCS, vol. 839, pp. 17–25. Springer, Berlin, Heidelberg (1994)

[19] 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, Berlin, Heidelberg (2016)

[20] Choudhuri, A.R., Maitra, S.: Significantly improved multi-bit differentials for reduced round Salsa and ChaCha. IACR Trans. Symmetric Cryptol. **2016**(2), 261–287 (2016)

[21] Dey, S., Sarkar, S.: Improved analysis for reduced round Salsa and Chacha. Discrete Appl. Math. **227**, 58–69 (2017)

[22] Aumasson, J., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New features of latin dances: Analysis of Salsa, ChaCha, and Rumba. In: Nyberg, K. (ed.) FSE 2008, Revised Selected Papers. LNCS, vol. 5086, pp. 470–488. Springer, Berlin, Heidelberg (2008)

[23] Shi, Z., Zhang, B., Feng, D., Wu, W.: Improved key recovery attacks on reduced-round Salsa20 and ChaCha. In: Kwon, T., Lee, M., Kwon, D. (eds.) ICISC 2012, Revised Selected Papers. LNCS, vol. 7839, pp. 337–351. Springer, Berlin, Heidelberg (2012)

[24] Maitra, S.: Chosen IV cryptanalysis on reduced round ChaCha and Salsa. Discrete Appl. Math. **208**, 88–97 (2016)

[25] Miyashita, S., Ito, R., Miyaji, A.: Pnb-focused differential cryptanalysis of ChaCha stream cipher. IACR Cryptol. ePrint Arch. **2021**, 1537 (2021). https://eprint.iacr.org/2021/1537 (to appear at ACISP 2022)

[26] Coutinho, M., Neto, T.C.S.: Improved linear approximations to ARX ciphers and attacks against ChaCha. In: Canteaut, A., Standaert, F. (eds.) EUROCRYPT 2021, Proceedings, Part I. LNCS, vol. 12696, pp. 711–740. Springer, Cham (2021)

[27] Dey, S., Dey, C., Sarkar, S., Meier, W.: Revisiting cryptanalysis on ChaCha from Crypto 2020 and Eurocrypt 2021. IEEE Trans. Inf. Theory (2022). https://doi.org/10.1109/TIT.2022.3171865

[28] Coutinho, M., Neto, T.C.S.: Improved linear approximations to ARX ciphers and attacks against ChaCha. IACR Cryptol. ePrint Arch. **2021**, 224 (2021). https://eprint.iacr.org/2021/224

[29] 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, Cham (2014)

[30] Neyman, J., Pearson, E.S.: On the problem of the most efficient tests of statistical hypotheses. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character **231**, 289–337 (1933)

[31] 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, Berlin, Heidelberg (2004)

[32] 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, Berlin, Heidelberg

(2012)

[33] Collard, B., Standaert, F., Quisquater, J.: Improving the time complexity of Matsui's linear cryptanalysis. In: Nam, K., Rhee, G. (eds.) ICISC 2007, Proceedings. LNCS, vol. 4817, pp. 77–88. Springer, Berlin, Heidelberg (2007)

[34] Biham, E., Dunkelman, O., Keller, N.: Enhancing differential-linear cryptanalysis. In: Zheng, Y. (ed.) ASIACRYPT 2002, Proceedings. LNCS, vol. 2501, pp. 254–266. Springer, Berlin, Heidelberg (2002)

[35] 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, Cham (2019)

[36] 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, Berlin, Heidelberg (2010)

[37] Blondeau, C., Leander, G., Nyberg, K.: Differential-linear cryptanalysis revisited. J. Cryptol. **30**(3), 859–888 (2017)

[38] Carlet, C.: Boolean Functions for Cryptography and Coding Theory. Cambridge University Press, Cambridge (2021)

[39] Nyberg, K.: Linear approximation of block ciphers. In: Santis, A.D. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Berlin, Heidelberg (1994)

[40] Mouha, N.: Chaskey: a MAC algorithm for microcontrollers - status update and proposal of Chaskey-12 -. IACR Cryptol. ePrint Arch. **2015**, 1182 (2015). https://eprint.iacr.org/2015/1182

[41] Coutinho, M., Neto, T.C.S.: New multi-bit differentials to improve attacks against ChaCha. IACR Cryptol. ePrint Arch. **2020**, 350 (2020). https://eprint.iacr.org/2020/350