Improved Differential-Linear Cryptanalysis of 7-round Chaskey with Partitioning

Gaëtan Leurent

Inria, France

Abstract. In this work we study the security of Chaskey, a recent lightweight MAC designed by Mouha *et al.*, currently being considered for standardization by ISO/IEC and ITU-T. Chaskey uses an ARX structure very similar to SipHash. We present the first cryptanalysis of Chaskey in the single user setting, with a differential-linear attack against 6 and 7 rounds, hinting that the full version of Chaskey with 8 rounds has a rather small security margin. In response to these attacks, a 12-round version has been proposed by the designers.

To improve the complexity of the differential-linear cryptanalysis, we refine a partitioning technique recently proposed by Biham and Carmeli to improve the linear cryptanalysis of addition operations. We also propose an analogue improvement of differential cryptanalysis of addition operations. Roughly speaking, these techniques reduce the data complexity of linear and differential attacks, at the cost of more processing time per data. It can be seen as the analogue for ARX ciphers of partial key guess and partial decryption for SBox-based ciphers.

When applied to the differential-linear attack against Chaskey, this partitioning technique greatly reduces the data complexity, and this also results in a reduced time complexity. While a basic differential-linear attack on 7 round takes 2^{78} data and time (respectively 2^{35} for 6 rounds), the improved attack requires only 2^{48} data and 2^{67} time (respectively 2^{25} data and 2^{29} time for 6 rounds). We also show an application of the partitioning technique to FEAL-8X, and we hope that this technique will lead to a better understanding of the security of ARX designs.

Keywords: Differential cryptanalysis, linear cryptanalysis, ARX, addition, partitioning, Chaskey, FEAL

1 Introduction

Linear cryptanalysis and differential cryptanalysis are the two major cryptanalysis techniques in symmetric cryptography. Differential cryptanalysis was introduced by Biham and Shamir in 1990 [6], by studying the propagation of differences in a cipher. Linear cryptanalysis was discovered in 1992 by Matsui [27,26], using a linear approximation of the non-linear round function.

In order to apply differential cryptanalysis (respectively, linear cryptanalysis), the cryptanalyst has to build differentials (resp. linear approximations) for each round of a cipher, such the output difference of a round matches the input

difference of the next round (resp. linear masks). The probability of the full differential or the imbalance of the full linear approximation is computed by multiplying the probabilities (respectively imbalances) of each round. This yields a statistical distinguisher for several rounds:

- A differential distinguisher is given by a plaintext difference δ_P and a ciphertext difference δ_C, so that the corresponding probability p is non-negligible:

$$p = \Pr\left[E(P \oplus \delta_P) = E(P) \oplus \delta_C\right] \gg 2^{-n}.$$

  The attacker collects $D = \mathcal{O}(1/p)$ pairs of plaintexts (P_i, P'_i) with $P'_i = P_i \oplus \delta_P$, and checks whether a pair of corresponding ciphertexts satisfies $C'_i = C_i \oplus \delta_C$. This happens with high probability for the cipher, but with low probability for a random permutation.
- A linear distinguisher is given by a plaintext mask χ_P and a ciphertext mask χ_C, so that the corresponding imbalance[1] ε is non-negligible:

$$\varepsilon = \left|2 \cdot \Pr\left[P[\chi_P] = C[\chi_C]\right] - 1\right| \gg 2^{-n/2}.$$

  The attacker collects $D = \mathcal{O}(1/\varepsilon^2)$ known plaintexts P_i and the corresponding ciphertexts C_i, and computes the observed imbalance $\hat{\varepsilon}$:

$$\hat{\varepsilon} = \left|2 \cdot \#\left\{i : P_i[\chi_P] = C_i[\chi_C]\right\}/D - 1\right|.$$

  The observed imbalance is close to ε for the attacked cipher, and smaller than $1/\sqrt{D}$ (with high probability) for a random function.

Last round attacks. The distinguishers are usually extended to a key-recovery attack on a few more rounds using partial decryption. The main idea is to guess the subkeys of the last rounds, and to compute an intermediate state value from the ciphertext and the subkeys. This allows to apply the distinguisher on the intermediate value: if the subkey guess was correct the distinguisher should succeed, but it is expected to fail for wrong key guesses. In a Feistel cipher, the subkey for one round is usually much shorter than the master key, so that this attack recovers a partial key without considering the remaining bits. This allows a divide and conquer strategy were the remaining key bits are recovered by exhaustive search. For an SBox-based cipher, this technique can be applied if the difference δ_C or the linear mask χ_C only affect a small number of SBoxes, because guessing the key bits affecting those SBoxes is sufficient to invert the last round.

ARX ciphers. In this paper we study the application of differential and linear cryptanalysis to ARX ciphers. ARX ciphers are a popular category of ciphers built using only additions ($x \boxplus y$), bit rotations ($x \lll n$), and bitwise xors ($x \oplus y$). These simple operations are very efficient in software and in hardware, but they

[1] The imbalance is also called correlation.

interact in complex ways that make analysis difficult and is expected to provide security. ARX constructions have been used for block ciphers (*e.g.* TEA, XTEA, FEAL, Speck), stream ciphers (*e.g.* Salsa20, ChaCha), hash functions (*e.g.* Skein, BLAKE), and for MAC algorithms (*e.g.* SipHash, Chaskey).

The only non-linear operation in ARX ciphers is the modular addition. Its linear and differential properties are well understood [38,34,25,33,40,30,15], and differential and linear cryptanalysis have been use to analyze many ARX designs (see for instance the following papers: [4,42,41,22,23,17,8,27]).

However, there is no simple way to extend differential or linear distinguishers to last-round attack for ARX ciphers. The problem is that they typically have 32-bit or 64-bit words, but differential and linear characteristics have a few active bits in each word.[2] Therefore a large portion of the key has to be guessed in order to perform partial decryption, and this doesn't give efficient attacks.

Besides, differential and linear cryptanalysis usually reach a limited number of rounds in ARX designs because the trails diverge quickly and we don't have good techniques to keep a low number of active bits. This should be contrasted with SBox-based designs where it is sometimes possible to build iterative trails, or trails with only a few active SBoxes per round. For instance, this is case for differential characteristics in DES [7] and linear trails in PRESENT [14].

Because of this, cryptanalysis methods that allow to divide a cipher E into two sub-ciphers $E = E_\perp \circ E_\top$ are particularly interesting for the analysis of ARX designs. In particular this is the case with boomerang attacks [39] and differential-linear cryptanalysis [21,5]. A boomerang attack uses differentials with probabilities $p_\top$ and $p_\perp$ in $E_\top$ and $E_\perp$, to build a distinguisher with complexity $\mathcal{O}(1/p_\top^2 p_\perp^2)$. A differential-linear attack uses a differential with probability p for $E_\top$ and a linear approximation with imbalance ε for $E_\perp$ to build a distinguisher with complexity about $\mathcal{O}(1/p^2\varepsilon^4)$ (using a heuristic analysis).

Table 1. Key-recovery attacks on Chaskey

Rounds	Data	Time	Gain	
6	2^{35}	2^{35}	1 bit	Differential-Linear
6	2^{25}	$2^{28.6}$	6 bits	Differential-Linear with partitioning
7	2^{78}	2^{78}	1 bit	Differential-Linear
7	2^{48}	2^{67}	6 bits	Differential-Linear with partitioning

Our results. In this paper, we consider improved techniques to attack ARX ciphers, with application to Chaskey. Since Chaskey has a strong diffusion, we start with differential-linear cryptanalysis, and we study in detail how to build a

[2] A notable counterexample is FEAL, which uses only 8-bit additions.

good differential-linear distinguisher, and how to improve the attack with partial key guesses.

Our main technique follows a recent paper by Biham and Carmeli [3], by partitioning the available data according to some plaintext and ciphertext bits. In each subset, some data bits have a fixed value and we can combine this information with key bit guesses to deduce bits after the key addition. These known bits result in improved probabilities for differential and linear cryptanalysis. While Biham and Carmeli considered partitioning with a single control bit (*i.e.* two partitions), and only for linear cryptanalysis, we extend this analysis to multiple control bits, and also apply it to differential cryptanalysis.

When applied to differential and linear cryptanalysis, this results in a significant reduction of the data complexity. Alternatively, we can extend the attack to a larger number of rounds with the same data complexity. Those results are very similar to the effect of partial key guess and partial decryption in a last-round attack: we turn a distinguisher into a key recovery attack, and we can add some rounds to the distinguisher. While this can increase the time complexity in some cases, we show that the reduced data complexity usually leads to a reduced time complexity. In particular, we adapt a convolution technique used for linear cryptanalysis with partial key guesses [16] in the context of partitioning.

These techniques result in significant improvements over the basic differential-linear technique: for 7 rounds of Chaskey (respectively 6 rounds), the differential-linear distinguisher requires 2^{78} data and time (respectively 2^{35}), but this can be reduced to 2^{48} data and 2^{67} time (respectively 2^{25} data and 2^{29} time) (see Table 1). The full version of Chaskey has 8 rounds, and is claimed to be secure against attacks with 2^{48} data and 2^{80} time.

The paper is organized as follows: we first explain the partitioning technique for linear cryptanalysis in Section 2 and for differential cryptanalysis in Section 3. We discuss the time complexity of the attacks in Section 4. Then we demonstrate the application of this technique to the differential-linear cryptanalysis of Chaskey in Section 5. Finally, we show how to apply the partitioning technique to reduce the data complexity of linear cryptanalysis against FEAL-8X in Appendix A.

2 Linear Analysis of Addition

We first discuss linear cryptanalysis applied to addition operations, and the improvement using partitioning. We describe the linear approximations using linear masks; for instance an approximation for E is written as $\Pr\left[E(x)[\chi'] = x[\chi]\right] = 1/2 \pm \varepsilon/2$ where χ and χ' are the input and output linear masks ($x[\chi]$ denotes $x[\chi_1] \oplus x[\chi_2] \oplus \cdots x[\chi_\ell]$, where $\chi = (\chi_1, \ldots \chi_\ell)$ and $x[\chi_i]$ is bit χ_i of x), and $\varepsilon \geq 0$ is the imbalance. We also denote the imbalance of a random variable x as $\mathcal{I}(x) = 2 \cdot \Pr[x = 0] - 1$, and $\varepsilon(x) = |\mathcal{I}(x)|$. We will sometimes identify a mask with the integer with the same binary representation, and use an hexadecimal notation.

We first study linear properties of the addition operation, and use an ARX cipher E as example. We denote the word size as w. We assume that the cipher starts with an xor key addition, and a modular addition of two state variables.[3] We denote the remaining operations as E', and we assume that we know a linear approximation $(\alpha, \beta, \gamma) \xrightarrow{E'} (\alpha', \beta', \gamma')$ with imbalance ε for E'. We further assume that the masks are sparse, and don't have adjacent active bits. Following previous works, the easier way to extend the linear approximation is to use the following masks for the addition:

$$(\alpha \oplus \alpha \gg 1, \alpha) \xrightarrow{\boxplus} \alpha. \tag{1}$$

As shown in Figure 1, this gives the following linear approximation for E:

$$(\alpha \oplus \alpha \gg 1, \beta \oplus \alpha, \gamma) \xrightarrow{E} (\alpha', \beta', \gamma'). \tag{2}$$

In order to explain our technique, we initially assume that α has a single active bit, *i.e.* $\alpha = 2^i$. We explain how to deal with several active bits in Section 2.3. If $i = 0$, the approximation of the linear addition has imbalance 1, but for other values of i, it is only $1/2$ [40]. In the following we study the case $i > 0$, where the linear approximation (2) for E has imbalance $\varepsilon/2$.

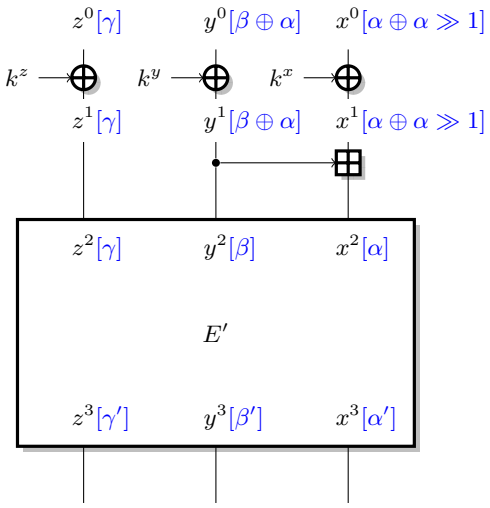

Fig. 1. Linear attack against the first addition

[3] This setting is quite general, because any operation before a key addition can be removed, as well as any linear operation after the key addition. Ciphers where the key addition is made with a modular addition do not fit this model, but the technique can easily be adapted.

2.1 Improved analysis with partitioning

We now explain the improved analysis of Biham and Carmeli [3]. A simple way to understand their idea is to look at the carry bits in the addition. More precisely, we study an addition operation $s = a \boxplus b$, and we are interested in the value $s[\alpha]$. We assume that $\alpha = 2^i, i > 0$, and that we have some amount of input/output pairs. We denote individual bits of a as $a_0, a_1, \ldots a_{n-1}$, where a_0 is the LSB (respectively, b_i for b and s_i for s). In addition, we consider the carry bits c_i, defined as $c_0 = 0$, $c_{i+1} = \mathrm{MAJ}(a_i, b_i, c_i)$ (where $\mathrm{MAJ}(a, b, c) = (a \wedge b) \vee (b \wedge c) \vee (c \wedge a)$). Therefore, we have $s_i = a_i \oplus b_i \oplus c_i$.

Note that the classical approximation $s_i = a_i \oplus a_{i-1} \oplus b_i$ holds with probability $3/4$ because $c_i = a_{i-1}$ with probability $3/4$. In order to improve this approximation, Biham and Carmeli partition the data according to the value of bits a_{i-1} and b_{i-1}. This gives four subsets:

`00` If $(a_{i-1}, b_{i-1}) = (0, 0)$, then $c_i = 0$ and $s_i = a_i \oplus b_i$.
`01` If $(a_{i-1}, b_{i-1}) = (0, 1)$, then $\varepsilon(c_i) = 0$ and $\varepsilon(s_i \oplus a_i \oplus a_{i-1}) = 0$.
`10` If $(a_{i-1}, b_{i-1}) = (1, 0)$, then $\varepsilon(c_i) = 0$ and $\varepsilon(s_i \oplus a_i \oplus a_{i-1}) = 0$.
`11` If $(a_{i-1}, b_{i-1}) = (1, 1)$, then $c_i = 1$ and $s_i = a_i \oplus b_i \oplus 1$.

If bits of a and b are known, filtering the data in subsets `00` and `11` gives a trail for the addition with imbalance 1 over one half of the data, rather than imbalance $1/2$ over the full data-set. This can be further simplified to the following:

$$s_i = a_i \oplus b_i \oplus a_{i-1} \qquad \text{if } a_{i-1} = b_{i-1} \tag{3}$$

In order to apply this analysis to the setting of Figure 1, we guess the key bits k^x_{i-1} and k^y_{i-1}, so that we can compute the values of x^1_{i-1} and y^1_{i-1} from x^0 and y^0. More precisely, an attack on E can be performed with a single (logical) key bit guess, using Eq. (3):

$$x^2_i = x^1_i \oplus y^1_i \oplus x^1_{i-1} \qquad \text{if } x^1_{i-1} = y^1_{i-1}$$
$$x^2_i = x^0_i \oplus y^0_i \oplus x^0_{i-1} \oplus k^x_i \oplus k^y_i \oplus k^x_{i-1} \qquad \text{if } x^0_{i-1} \oplus y^0_{i-1} = k^x_{i-1} \oplus k^y_{i-1}$$

If we guess the key bit $k^x_{i-1} \oplus k^y_{i-1}$, we can filter the data satisfying $x^0_{i-1} \oplus y^0_{i-1} = k^x_{i-1} \oplus k^y_{i-1}$, and we have $\varepsilon(x^2_i \oplus x^0_i \oplus y^0_i \oplus x^0_{i-1}) = 1$. Therefore the linear approximation (2) has imbalance ε. We need $1/\varepsilon^2$ data after the filtering for the attack to succeed, *i.e.* $2/\varepsilon^2$ in total. The time complexity is also $2/\varepsilon^2$ because we run the analysis with $1/\varepsilon^2$ data for each key guess. This is an improvement over a simple linear attack using (2) with imbalance $\varepsilon/2$, with $4/\varepsilon^2$ data.

Complexity. In general this partitioning technique multiply the data and time complexity by the following ratio:

$$R^D_{\mathrm{lin}} = \frac{\mu^{-1}/\widetilde{\varepsilon}^2}{1/\varepsilon^2} = \varepsilon^2/\mu\widetilde{\varepsilon}^2 \qquad R^T_{\mathrm{lin}} = \frac{2^\kappa/\widetilde{\varepsilon}^2}{1/\varepsilon^2} = 2^\kappa\varepsilon^2/\widetilde{\varepsilon}^2 \tag{4}$$

where μ is the fraction of data used in the attack, κ is the number of guessed key bits, ε is the initial imbalance, and $\widetilde{\varepsilon}$ is the improved imbalance for the selected subset. For Biham and Carmeli's attack, we have $\mu = 1/2$, $\kappa = 1$ and $\widetilde{\varepsilon} = 2\varepsilon$, hence $R^D_{\mathrm{lin}} = 1/2$ and $R^T_{\mathrm{lin}} = 1/2$.

2.2 Generalized partitioning

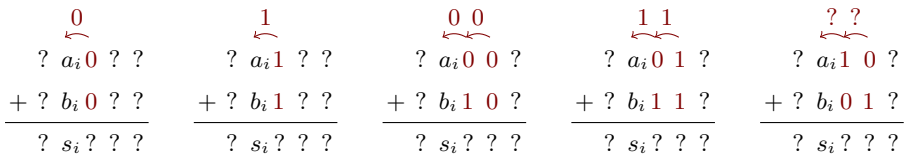

Fig. 2. Some cases of partitioning for linear cryptanalysis of an addition

We now refine the technique of Biham and Carmeli using several control bits. In particular, we analyze cases `01` and `10` with extra control bits a_{i-2} and b_{i-2} (some of the cases of shown in Figure 2):

`01.00` If $(a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}) = (0, 1, 0, 0)$,
then $c_{i-1} = 0$, $c_i = 0$ and $s_i = a_i \oplus b_i$.

`01.01` If $(a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}) = (0, 1, 0, 1)$,
then $\varepsilon(c_{i-1}) = 0$, $\varepsilon(c_i) = 0$, and $\varepsilon(s_i \oplus a_i \oplus a_{i-1}) = 0$.

`01.10` If $(a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}) = (0, 1, 1, 0)$,
then $\varepsilon(c_{i-1}) = 0$, $\varepsilon(c_i) = 0$, and $\varepsilon(s_i \oplus a_i \oplus a_{i-1}) = 0$.

`01.11` If $(a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}) = (0, 1, 1, 1)$,
then $c_{i-1} = 1$, $c_i = 1$ and $s_i = a_i \oplus b_i \oplus 1$.

`10.00` If $(a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}) = (1, 0, 0, 0)$,
then $c_{i-1} = 0$, $c_i = 0$ and $s_i = a_i \oplus b_i$.

`10.01` If $(a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}) = (1, 0, 0, 1)$,
then $\varepsilon(c_{i-1}) = 0$, $\varepsilon(c_i) = 0$, and $\varepsilon(s_i \oplus a_i \oplus a_{i-1}) = 0$.

`10.10` If $(a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}) = (1, 0, 1, 0)$,
then $\varepsilon(c_{i-1}) = 0$, $\varepsilon(c_i) = 0$, and $\varepsilon(s_i \oplus a_i \oplus a_{i-1}) = 0$.

`10.11` If $(a_{i-1}, b_{i-1}, a_{i-2}, b_{i-2}) = (1, 0, 1, 1)$,
then $c_{i-1} = 1$, $c_i = 1$ and $s_i = a_i \oplus b_i \oplus 1$.

This yields an improved partitioning because we now have a trail for the addition with imbalance 1 in 12 out of 16 subsets: `00.00`, `00.01`, `00.10`, `00.11`, `01.00`, `01.11`, `10.00`, `10.11`, `11.00`, `11.01`, `11.10`, `11.11`. We can also simplify this case analysis:

$$s_i = \begin{cases} a_i \oplus b_i \oplus a_{i-1} & \text{if } a_{i-1} = b_{i-1} \\ a_i \oplus b_i \oplus a_{i-2} & \text{if } a_{i-1} \neq b_{i-1} \text{ and } a_{i-2} = b_{i-2} \end{cases} \tag{5}$$

This gives an improved analysis of E by guessing more key bits. More precisely we need $k^x_{i-1} \oplus k^y_{i-1}$ and $k^x_{i-2} \oplus k^y_{i-2}$, as shown below:

$$x_i^2 = \begin{cases} x_i^1 \oplus y_i^1 \oplus x_{i-1}^1 & \text{if } x_{i-1}^1 = y_{i-1}^1 \\ x_i^1 \oplus y_i^1 \oplus x_{i-2}^1 & \text{if } x_{i-1}^1 \neq y_{i-1}^1 \text{ and } x_{i-2}^1 = y_{i-2}^1 \end{cases}$$

$$x_i^2 = \begin{cases} x_i^0 \oplus y_i^0 \oplus x_{i-1}^0 \oplus k_i^x \oplus k_i^y \oplus k_{i-1}^x & \text{if } x_{i-1}^0 \oplus y_{i-1}^0 = k_{i-1}^x \oplus k_{i-1}^y \\ x_i^0 \oplus y_i^0 \oplus x_{i-2}^0 \oplus k_i^x \oplus k_i^y \oplus k_{i-2}^x & \text{if } x_{i-1}^0 \oplus y_{i-1}^0 \neq k_{i-1}^x \oplus k_{i-1}^y \\ & \quad \text{and } x_{i-2}^0 \oplus y_{i-2}^0 = k_{i-2}^x \oplus k_{i-2}^y \end{cases}$$

$$\varepsilon(x_i^2 \oplus x_i^0 \oplus y_i^0 \oplus x_{i-1}^0) = 1 \qquad \text{if } x_{i-1}^0 \oplus y_{i-1}^0 = k_{i-1}^x \oplus k_{i-1}^y$$

$$\varepsilon(x_i^2 \oplus x_i^0 \oplus y_i^0 \oplus x_{i-2}^0) = 1 \qquad \begin{array}{l} \text{if } x_{i-1}^0 \oplus y_{i-1}^0 \neq k_{i-1}^x \oplus k_{i-1}^y \\ \quad \text{and } x_{i-2}^0 \oplus y_{i-2}^0 = k_{i-2}^x \oplus k_{i-2}^y \end{array}$$

Since this analysis yields different input masks for different subsets of the data, we use an analysis following multiple linear cryptanalysis [9]. We first divide the data into four subsets, depending on the value of $x^0_{i-1} \oplus y^0_{i-1}$ and $x^0_{i-2} \oplus y^0_{i-2}$, and we compute the measured (signed) imbalance $\hat{\mathcal{I}}[s]$ of each subset. Then, for each guess of the key bits $k^x_{i-1} \oplus k^y_{i-1}$, and $k^x_{i-2} \oplus k^y_{i-2}$, we deduce the expected imbalance $\mathcal{I}_k[s]$ of each subset, and we compute the distance to the observed imbalance as $\sum_s (\hat{\mathcal{I}}[s] - \mathcal{I}_k[s])^2$. According to the analysis of Biryukov, De Cannière and Quisquater, the correct key is ranked first (with minimal distance) with high probability when using $\mathcal{O}(1/c^2)$ samples, where $c^2 = \sum_i \mathcal{I}_i^2 = \sum_i \varepsilon_i^2$ is the capacity of the system of linear approximations. Since we use three approximations with imbalance ε, the capacity of the full system is $3\varepsilon^2$, and we need $1/3 \cdot 1/\varepsilon^2$ data in each subset after partitioning, *i.e.* $4/3 \cdot 1/\varepsilon^2$ in total.

Again, the complexity ratio of this analysis can be computed as $R^D_{\text{lin}} = \varepsilon^2/\mu\widetilde{\varepsilon}^2$ $R^T_{\text{lin}} = 2^\kappa \varepsilon^2/\widetilde{\varepsilon}^2$ With $\mu = 3/4$ and $\widetilde{\varepsilon} = 2\varepsilon$, we find:

$$R^D_{\text{lin}} = 1/3 \qquad\qquad R^T_{\text{lin}} = 1.$$

The same technique can be used to refine the partitioning further, and give a complexity ratio of $R^D_{\text{lin}} = 1/4 \times 2^\kappa/(2^\kappa - 1)$ when guessing κ bits.

Time complexity. In general, the time complexity of this improved partitioning technique is the same as the time complexity as the basic attack ($R^T_{\text{lin}} = 1$), because we have to repeat the analysis 4 times (for each key of the key bits) with one fourth of the amount of data. We describe some techniques to reduce the time complexity in Section 4.

2.3 Combining partitions

Finally, we can combine several partitions to analyze an addition with several active bits. If we use k_1 partitions for the first bit, and k_2 for the second bit, this yields a combined partition with $k_1 \cdot k_2$ cases. If the bits are not close to each other, the gains of each bit are multiplied. This can lead to significant improvements even though R_{lin} is small for a single active bit.

For more complex scenarios, we select the filtering bits assuming that the active bits don't interact, and we evaluate experimentally the probability in each subset. We can further study the matrix of probabilities to detect (logical) bits with no or little effect on the total capacity in order to improve the complexity of the attack. This will be used for our applications in Section 5 and Appendix A.

3 Differential Analysis of Addition

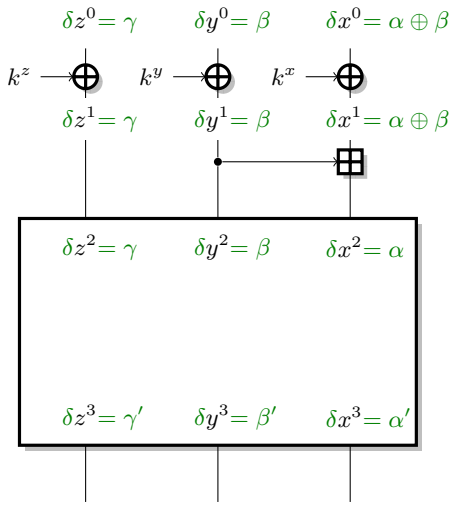

Fig. 3. Differential attack against the first addition

We now study differential properties of the addition. We perform our analysis in the same way as the analysis of Section 2, following Figure 3. We consider the first addition operation separately, and we assume that we know a differential $(\alpha, \beta, \gamma) \to (\alpha', \beta', \gamma')$ with probability p for the remaining of the cipher. Following previous works, a simple way to extend the differential is to linearize the first addition, yielding the following differences for the addition:

$$\alpha \oplus \beta, \beta \xrightarrow{\boxplus} \alpha.$$

Similarly to our analysis of linear cryptanalysis, we consider a single addition $s = a \boxplus b$, and we first assume that a single bit is active through the addition. However, we have to consider several cases, depending on how many input/output bits are active. The cases are mostly symmetric, but there are important differences in the partitioning.

3.1 Analysis of $(\alpha = 0, \beta = 2^i)$

With $i < w - 1$, the probability for the addition is $\Pr[(2^i, 2^i) \xrightarrow{\boxplus} 0] = 1/2$.

Improved analysis with structures. We first discuss a technique using multiple differentials and structures. More precisely, we use the following differentials for the addition:[4]

$$\mathcal{D}_1 : (2^i, 2^i) \xrightarrow{\boxplus} 0 \qquad \Pr\left[(2^i, 2^i) \xrightarrow{\boxplus} 0\right] = 1/2$$
$$\mathcal{D}_2 : (2^i \oplus 2^{i+1}, 2^i) \xrightarrow{\boxplus} 0 \qquad \Pr\left[(2^i \oplus 2^{i+1}, 2^i) \xrightarrow{\boxplus} 0\right] = 1/4$$

We can improve the probability of $\mathcal{D}_2$ using a partitioning according to (a_i, a_{i+1}):

`00` If $(a_i, a_{i+1}) = (0, 0)$, then $a' = a \boxplus 2^i \boxplus 2^{i+1}$ and $s \neq s'$.
`01` If $(a_i, a_{i+1}) = (0, 1)$, then $a' = a \boxminus 2^i$ and $\Pr[s = s'] = 1/2$.
`10` If $(a_i, a_{i+1}) = (1, 0)$, then $a' = a \boxplus 2^i$ and $\Pr[s = s'] = 1/2$.
`11` If $(a_i, a_{i+1}) = (1, 1)$, then $a' = a \boxminus 2^i \boxminus 2^{i+1}$ and $s \neq s'$.

This can be written as:

$$\Pr\left[(2^i, 2^i) \xrightarrow{\boxplus} 0\right] = 1/2$$
$$\Pr\left[(2^i \oplus 2^{i+1}, 2^i) \xrightarrow{\boxplus} 0\right] = 1/2 \qquad \text{if } a_i \neq a_{i+1}$$

The use of structures allows to build pairs of data for both differentials from the same data set. More precisely, we consider the following inputs:

$$p = (x^0, y^0, z^0) \qquad q = (x^0 \oplus 2^i, y^0 \oplus 2^i, z^0)$$
$$r = (x^0 \oplus 2^{i+1}, y^0, z^0) \qquad s = (x^0 \oplus 2^{i+1} \oplus 2^i, y^0 \oplus 2^i, z^0)$$

We see that (p, q) and (r, s) follow the input difference of $\mathcal{D}_1$, while (p, s) and (r, q) follow the input difference of $\mathcal{D}_2$. Moreover, we have from the partitioning:

$$\Pr[E(p) \oplus E(q) = (\alpha', \beta', \gamma')] = 1/2 \cdot p$$
$$\Pr[E(r) \oplus E(s) = (\alpha', \beta', \gamma')] = 1/2 \cdot p$$
$$\Pr[E(p) \oplus E(s) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \quad \text{if } x_i^0 \oplus x_{i+1}^0 \neq k_i^x \oplus k_{i+1}^x$$
$$\Pr[E(r) \oplus E(q) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \quad \text{if } x_i^0 \oplus x_{i+1}^0 = k_i^x \oplus k_{i+1}^x$$

For each key guess, we select three candidate pair out of a structure of four plaintexts, and every pair follows a differential for E with probability $p/2$. Therefore we need $2/p$ pairs, with a data complexity of $8/3 \cdot 1/p$ rather than $4 \cdot 1/p$.

In general this partitioning technique multiply the data and time complexity by the following ratio:

$$R^D_{\text{diff}} = \frac{\widetilde{p}^{-1} T/(\mu T^2/4)}{p^{-1} T/(T/2)} = \frac{2p}{\mu T \widetilde{p}} \qquad R^T_{\text{diff}} = 2^\kappa \mu R^D_{\text{diff}} = \frac{2^{\kappa+1} p}{T \widetilde{p}}, \tag{6}$$

[4] Note that in the application to E, we can modify the difference in x^1 but not in y^1.

where μ is the fraction of data used in the attack, κ is the number of guessed key bits, T is the number of plaintexts in a structure (we consider $T^2/4$ pairs rather than $T/2$ without structures) p is the initial probability, and $\widetilde{p}$ is the improved probability for the selected subset. Here we have $\mu = 3/4$, $\kappa = 1$, $T = 4$, and $\widetilde{p} = p$, hence

$$R^D_{\text{diff}} = 2/3 \qquad\qquad R^T_{\text{diff}} = 1$$

Moreover, if the differential trail is used in a boomerang attack, or in a differential-linear attack, it impacts the complexity twice, but the involved key bits are the same, and we only need to use the structure once. Therefore, the complexity ratio should be evaluated as:

$$R^D_{\text{diff-2}} = \frac{\widetilde{p}^{-2}T/(\mu T^2/4)}{p^{-2}T/(T/2)} = \frac{2p^2}{\mu T\widetilde{p}^2} \qquad R^T_{\text{diff-2}} = 2^\kappa \mu R^D_{\text{diff-2}} = \frac{2^{\kappa+1}p^2}{T\widetilde{p}^2}, \tag{7}$$

In this scenario, we have the same ratios:

$$R^D_{\text{diff-2}} = 2/3 \qquad\qquad R^T_{\text{diff-2}} = 1$$

Generalized partitioning. We can refine the analysis of the addition by partitioning according to (b_i). This gives the following:

$$\begin{aligned} \Pr\left[(2^i, 2^i) \to 0\right] &= 1 && \text{if } a_i \neq b_i \\ \Pr\left[(2^i \oplus 2^{i+1}, 2^i) \to 0\right] &= 1 && \text{if } a_i = b_i \text{ and } a_i \neq a_{i+1} \end{aligned}$$

This gives an attack with $T = 4$, $\mu = 3/8$, $\kappa = 2$ and $\widetilde{p} = 2p$, which yield the same ratio in a simple differential setting, but a better ratio for a boomerang or differential-linear attack:

$$\begin{aligned} R^D_{\text{diff}} &= 2/3 & R^T_{\text{diff}} &= 1 \\ R^D_{\text{diff-2}} &= 1/3 & R^T_{\text{diff-2}} &= 1/2 \end{aligned}$$

In addition, this analysis allows to recover an extra key bit, which can be useful for further steps of an attack.

Larger structure. Alternatively, we can use a larger structure to reduce the complexity: with a structure of size 2^t, we have an attack with a ratio $R^D_{\text{diff}} = 1/2 \times 2^\kappa/(2^\kappa - 1)$, by guessing $\kappa - 1$ key bits.

3.2 Analysis of $(\boldsymbol{\alpha = 2^i, \beta = 0})$

With $i < w - 1$, the probability for the addition is $\Pr[(2^i, 0) \xrightarrow{\boxplus} 2^i] = 1/2$.

Improved analysis with structures. As in the previous section, we consider multiple differentials, and use partitioning to improve the probability:

$$\mathcal{D}_1 : \qquad \Pr\left[(2^i, 0) \xrightarrow{\boxplus} 2^i\right] = 1/2$$

$$\mathcal{D}_2 : \qquad \Pr\left[(2^i \oplus 2^{i+1}, 0) \xrightarrow{\boxplus} 2^i\right] = 1/2 \qquad \text{if } a_i \neq a_{i+1}$$

We also use structures in order to build pairs of data for both differentials from the same data set. More precisely, we consider the following inputs:

$$p = (x^0, y^0, z^0) \qquad q = (x^0 \oplus 2^i, y^0, z^0)$$
$$r = (x^0 \oplus 2^{i+1}, y^0, z^0) \qquad s = (x^0 \oplus 2^{i+1} \oplus 2^i, y^0, z^0)$$

We see that (p, q) and (r, s) follow the input difference of $\mathcal{D}_1$, while (p, s) and (r, q) follow the input difference of $\mathcal{D}_2$. Moreover, we have from the partitioning:

$$\begin{aligned}
&\Pr[E(p) \oplus E(q) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \\
&\Pr[E(r) \oplus E(s) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \\
&\Pr[E(p) \oplus E(s) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \quad \text{if } x_i^0 \oplus x_{i+1}^0 \neq k_i^x \oplus k_{i+1}^x \\
&\Pr[E(r) \oplus E(q) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \quad \text{if } x_i^0 \oplus x_{i+1}^0 = k_i^x \oplus k_{i+1}^x
\end{aligned}$$

In this case, we also have $\mu = 3/4$, $T = 4$, and $\widetilde{p} = p$, hence

$$R^D_{\text{diff}} = 2/3 \qquad R^T_{\text{diff}} = 1$$
$$R^D_{\text{diff-2}} = 2/3 \qquad R^T_{\text{diff-2}} = 1$$

Generalized partitioning. Again, we can refine the analysis of the addition by partitioning according to (s_i). This gives the following:

$$\begin{aligned}
\Pr\left[(2^i, 0) \to 2^i\right] &= 1 \qquad \text{if } a_i = s_i \\
\Pr\left[(2^i \oplus 2^{i+1}, 0) \to 2^i\right] &= 1 \qquad \text{if } a_i \neq s_i \text{ and } a_i \neq a_{i+1}
\end{aligned}$$

Since we can not readily filter according to bits of s, we use the results of Section 2:

$$a_i \oplus b_i \oplus a_{i-1} = s_i \qquad \text{if } a_{i-1} = b_{i-1}$$

This gives:

$$\begin{aligned}
\Pr\left[(2^i, 0) \to 2^i\right] &= 1 \qquad \text{if } b_i = a_{i-1} \text{ and } a_{i-1} = b_{i-1} \\
\Pr\left[(2^i \oplus 2^{i+1}, 0) \to 2^i\right] &= 1 \qquad \text{if } b_i \neq a_{i-1} \text{ and } a_{i-1} = b_{i-1} \text{ and } a_i \neq a_{i+1}
\end{aligned}$$

Unfortunately, we can only use a small fraction of the pairs $\mu = 3/16$. With $T = 4$ and $\widetilde{p} = 2p$, this yields, an increase of the data complexity for a simple differential attack:

$$R^D_{\text{diff}} = 4/3 \qquad R^T_{\text{diff}} = 1/2$$
$$R^D_{\text{diff-2}} = 2/3 \qquad R^T_{\text{diff-2}} = 1/4$$

3.3 Analysis of $(\alpha = 2^i, \beta = 2^i)$

With $i < w - 1$, the probability for the addition is $\Pr[(0, 2^i) \xrightarrow{\boxplus} 2^i] = 1/2$.

The results in this section will be the same as in the previous section, but we have to use a different structure. Indeed when this analysis is applied to E, we can freely modify the difference in x^0 but not in y^0, because it would affect the differential in E'.

More precisely, we use the following differentials:

$$\begin{aligned} \mathcal{D}_1 : &\qquad \Pr\left[(0, 2^i) \xrightarrow{\boxplus} 2^i\right] = 1/2 \\ \mathcal{D}_2 : &\qquad \Pr\left[(2^{i+1}, 2^i) \xrightarrow{\boxplus} 2^i\right] = 1/2 \qquad \text{if } a_{i+1} \neq b_i \end{aligned}$$

and the following structure:

$$\begin{aligned} p &= (x^0, y^0, z^0) & q &= (x^0, y^0 \oplus 2^i, z^0) \\ r &= (x^0 \oplus 2^{i+1}, y^0, z^0) & s &= (x^0 \oplus 2^{i+1}, y^0 \oplus 2^i, z^0) \end{aligned}$$

This yields:

$$\begin{aligned} &\Pr[E(p) \oplus E(q) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \\ &\Pr[E(r) \oplus E(s) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \\ &\Pr[E(p) \oplus E(s) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \quad \text{if } x_i^1 \oplus x_{i+1}^0 \neq k_i^y \oplus k_{i+1}^x \\ &\Pr[E(r) \oplus E(q) = (\alpha', \beta', \gamma')] = 1/2 \cdot p \quad \text{if } x_i^1 \oplus x_{i+1}^0 = k_i^y \oplus k_{i+1}^x \end{aligned}$$

4 Improving the Time Complexity

The analysis of the previous sections assume that we repeat the distinguisher for each key guess, so that the data complexity is reduced in a very generic way. When this is applied to differential or linear cryptanalysis, it usually result in an increased time complexity ($R^T > 1$). However, when the distinguisher is a simple linear of differential distinguisher, we can perform the analysis in a more efficient way, using the same techniques that are used in attacks with partial key guess against SBox-based ciphers. For linear cryptanalysis, we use a variant of Matsui's Algorithm 2 [26], and the improvement using convolution algorithm [16]; for differential cryptanalysis we filter out pairs that can not be a right pair for any key. In the best cases, the time complexity of the attacks can be reduced to essentially the data complexity.

4.1 Linear analysis

We follow the analysis of Matsui's Algorithm 2, with a distillation phase using counters to keep track of the important features of the data, and an analysis phase for every key that requires only the counters rather than the full dataset.

More precisely, let us explain this idea within the setting of Section 2.2 and Figure 1. For each key guess, the attacker computes the observed imbalance over a subset $\mathcal{S}_k$ corresponding to the data with $x^0_{i-1} \oplus y^0_{i-1} = k^x_{i-1} \oplus k^y_{i-1}$, or $\left(x^0_{i-1} \oplus y^0_{i-1} \neq k^x_{i-1} \oplus k^y_{i-1} \text{ and } x^0_{i-2} \oplus y^0_{i-2} = k^x_{i-2} \oplus k^y_{i-2}\right)$:

$$\begin{aligned}\hat{\mathcal{I}} &= \mathcal{I}_{\mathcal{S}_k}(P[\chi_P] \oplus C[\chi_C]) \\ &= 1/\left|\mathcal{S}_k\right| \times \sum_{\mathcal{S}_k} (-1)^{P[\chi_P] \oplus C[\chi_C]}\end{aligned}$$

where (using $\alpha = 2^i$)

$$\begin{aligned}P[\chi_P] \oplus C[\chi_C] &= x_i^2 \oplus y^2[\beta] \oplus z^2[\gamma] \oplus x^3[\alpha'] \oplus y^3[\beta'] \oplus z^3[\gamma'] \\ &= \begin{cases} x_i^0 \oplus y_i^0 \oplus x_{i-1}^0 \oplus y^0[\beta] \oplus z^0[\gamma] \oplus x^3[\alpha'] \oplus y^3[\beta'] \oplus z^3[\gamma'] \\ \qquad \text{if } x^0_{i-1} \oplus y^0_{i-1} = k^x_{i-1} \oplus k^y_{i-1} \\ x_i^0 \oplus y_i^0 \oplus x_{i-2}^0 \oplus y^0[\beta] \oplus z^0[\gamma] \oplus x^3[\alpha'] \oplus y^3[\beta'] \oplus z^3[\gamma'] \\ \qquad \text{if } x^0_{i-1} \oplus y^0_{i-1} \neq k^x_{i-1} \oplus k^y_{i-1} \\ \qquad \text{and } x^0_{i-2} \oplus y^0_{i-2} = k^x_{i-2} \oplus k^y_{i-2} \end{cases}\end{aligned}$$

Therefore, the imbalance can be efficiently reconstructed from a series of 2^4 counters keeping track of the amount of data satisfying every possible value of the following bits:

$$\begin{gathered} x_i^0 \oplus y_i^0 \oplus x_{i-1}^0 \oplus y^0[\beta] \oplus z^0[\gamma] \oplus x^3[\alpha'] \oplus y^3[\beta'] \oplus z^3[\gamma'], \\ x^0_{i-1} \oplus x^0_{i-2}, \quad x^0_{i-1} \oplus y^0_{i-1}, \quad x^0_{i-2} \oplus y^0_{i-2} \end{gathered}$$

This results in an attack where the time complexity is equal to the data complexity, plus a small cost to compute the imbalance. The analysis phase require only about 2^6 operations in this case (adding 2^4 counters for 2^2 key guesses). When the amount of data required is larger than 2^6, the analysis step is negligible.

When several partitions are combined (with several active bits in the first additions), the number of counters increases to 2^b, where b is the number of control bits. To reduce the complexity of the analysis phase, we can use a convolution algorithm (following [16]), so that the cost of the distillation is only $\mathcal{O}(b \cdot 2^b)$ rather than $\mathcal{O}(2^\kappa \cdot 2^b)$. This will be explained in more details with the application to Chaskey in Section 5.

In general, there is a trade-off between the number of partitioning bits, and the complexity. A more precise partitioning allows to reduce the data complexity, but this implies a larger set of counters, hence a larger memory complexity. When the number of partitioning bits reaches the data complexity, the analysis phase becomes the dominant phase, and the time complexity is larger than the data complexity.

4.2 Differential analysis

For a differential attack with partitioning, we can also reduce the time complexity, by filtering pairs before the analysis phase. In the following, we assume that

we use a simple differential distinguisher with output difference δ', following Section 3 (where $\delta' = (\alpha', \beta', \gamma')$)

We first define a linear function L with rank $n-1$ (where n is the block size), so that $L(\delta') = 0$. In particular, any pair $x, x' = x \oplus \delta'$ satisfies $L(x) = L(x')$. This allows to detect collisions by looking at all *values* in a structure, rather than all *pairs* in a structure. We just compute $L(E(x))$ for all x's in a structure, and we look for collisions.

5 Application to Chaskey

Chaskey is a recent MAC proposal designed jointly by researchers from COSIC and Hitachi [32]. The mode of operation of Chaskey is based on CBC-MAC with an Even-Mansour cipher; but it can also be described as a permutation-based design as seen in Figure 4. Chaskey is designed to be extremely fast on 32-bit micro-controllers, and the internal permutation follows an ARX construction with 4 32-bit words based on SipHash; it is depicted in Figure 5. Since the security of Chaskey is based on an Even-Mansour cipher, the security bound has a birthday term $\mathcal{O}(TD \cdot 2^{-128})$. More precisely, the designers claim that it should be secure up to 2^{48} queries, and 2^{80} computations.

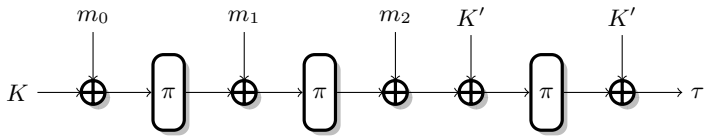

Fig. 4. Chaskey mode of operation (full block message)

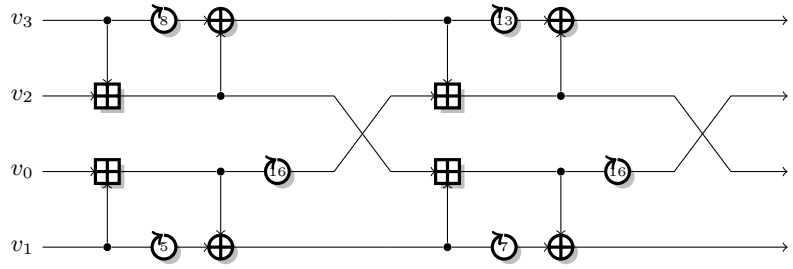

Fig. 5. One round of the Chaskey permutation. The full permutation has 8 rounds.

So far, the only external cryptanalysis results on Chaskey are generic attacks in the multi-user setting [28]. The only analysis of the permutation is in the submission document; the best result is a 4 round bias, that can probably be

extended into a 5 round attack following the method of attacks against the Salsa family[1]. It is important to try more advanced techniques in order to understand the security of Chaskey, in particular because it is being considered for standardization.

5.1 Differential-linear cryptanalysis

Table 2. Probabilities of the best differential characteristics of Chaskey reported by the designers [32]

Rounds:	1	2	3	4	5	6	7	8
Probability:	1	2^{-4}	2^{-16}	2^{-37}	$2^{-73.1}$	$2^{-132.8}$	$2^{-205.6}$	$2^{-289.9}$

The best differential characteristics found by the designers of Chaskey quickly become unusable when the number of rounds increase (See Table 2). The designers also report that those characteristics have an "hourglass structure": there is a position in the middle where a single bit is active, and this small difference is expanded by the avalanche effect when propagating in both direction. This is typical of ARX designs: short characteristics have a high probability, but after a few rounds the differences cannot be controlled and the probability decrease very fast. The same observation typically holds also for linear trails.

Because of these properties, attacks that can divide the cipher E in two parts $E = E_\perp \circ E_\top$ and build characteristics or trail for both half independently – such as the boomerang attack or differential-linear cryptanalysis – are particularly interesting. In particular, many attacks on ARX designs are based on the boomerang attack [43,29,11,24,20,36] or differential-linear cryptanalysis [19]. Since Chaskey never uses the inverse permutation, we cannot apply a boomerang attack, and we focus on differential-linear cryptanalysis.

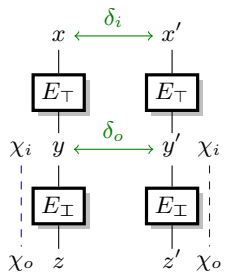

Fig. 6. Differential-linear cryptanalysis

Differential-linear cryptanalysis uses a differential $\delta_i \xrightarrow{E_\top} \delta_o$ with probability p for $E_\top$, and a linear approximation $\chi_i \xrightarrow{E_\perp} \chi_o$ with imbalance ε for $E_\perp$ (see Figure 6). The attacker uses pairs of plaintexts (P_i, P_i') with $P_i' = P_i \oplus \delta_i$, and computes the observed imbalance $\hat{\varepsilon} = |2 \cdot \#\{i : C_i[\chi_o] = C_i'[\chi_o]\}/D - 1|$. Following the heuristic analysis of [5], the expected imbalance is about $p\varepsilon^2$, which gives an attack complexity of $\mathcal{O}(2/p^2\varepsilon^4)$:

- A pair of plaintext satisfies $E_\top(P) \oplus E_\top(P') = \delta_o$ with probability p. In this case, we have $E_\top(P)[\chi_i] \oplus E_\top(P')[\chi_i] = \delta_o[\chi_i]$. Without loss of generality, we assume that $\delta_o[\chi_i] = 0$.
- Otherwise, we expect that $E_\top(P)[\chi_i] \oplus E_\top(P')[\chi_i]$ is not biased. This gives the following:

$$\Pr\left[E_\top(P)[\chi_i] \oplus E_\top(P')[\chi_i] = 0\right] = p + (1-p) \cdot 1/2 = 1/2 + p/2 \tag{8}$$

$$\varepsilon(E_\top(P)[\chi_i] \oplus E_\top(P')[\chi_i]) = p \tag{9}$$

- We also have $\varepsilon(E_\top(P)[\chi_i] \oplus C[\chi_o]) = \varepsilon(E_\top(P')[\chi_i] \oplus C'[\chi_o]) = \varepsilon$ from the linear approximations. Combining with (9), we get $\varepsilon(C[\chi_o] \oplus C'[\chi_o]) = p\varepsilon^2$.

A more rigorous analysis has been recently provided by Blondeau, Leander and Nyberg [13], but since we use experimental values to evaluate the complexity of our attacks, this heuristic explanation will be sufficient.

5.2 Using partitioning

A differential-linear distinguisher can easily be improved using the results of Section 2 and 3. We can improve the differential and linear part separately, and combine the improvements on the differential-linear attack. More precisely, we have to consider structures of plaintexts, and to guess some key bits in the differential and linear parts. We partition all the potential pairs in the structures according to the input difference, and to the filtering bits in the differential and linear part; then we evaluate the observed imbalance $\hat{\mathcal{I}}[s]$ in every subset s. Finally, for each key guess k, we compute the expected imbalance $\mathcal{I}_k[s]$ for each subset s, and then we evaluate the distance between the observed and expected imbalances as $L(k) = \sum_s (\hat{\mathcal{I}}[s] - \mathcal{I}_k[s])^2$ (following the analysis of multiple linear cryptanalysis [10]).

While we follow the analysis of multiple linear cryptanalysis to evaluate the complexity of our attack, we use each linear approximation on a different subset of the data, partitioned according to the filtering bits. In particular, we don't have to worry about the independence of the linear approximations.

If we use structures of size T, and select a fraction μ_{diff} of the input pairs with an improved differential probability $\widetilde{p}$, and a fraction μ_{lin} of the output pairs with an improved linear imbalance $\widetilde{\varepsilon}$, the data complexity of the attack is $\mathcal{O}(\mu_{\text{lin}}\mu_{\text{diff}}^2 T/2 \times 2/\widetilde{p}^2\widetilde{\varepsilon}^4)$. This corresponds to a complexity ratio of $R^D_{\text{diff-2}} {R^D_{\text{lin}}}^2$.

More precisely, using differential filtering bits p_{diff} and linear filtering bits c_{lin}, the subsets are defined by the input difference Δ, the plaintext bits $P[p_{\text{diff}}]$

and the cipher text bits $C[c_{\text{lin}}]$ and $C'[c_{\text{lin}}]$, with $C = E(P)$ and $C' = E(P \oplus \Delta)$. In practice, for every P, P' in a structure, we update the value of $\hat{\mathcal{I}}[P \oplus P', P[p_{\text{lin}}], C[c_{\text{diff}}], C'[c_{\text{diff}}]]$.

We also take advantage of the Even-Mansour construction of Chaskey, without keys inside the permutation. Indeed, the filtering bits used to define the subsets s correspond to the key bits used in the attack. Therefore, we only need to compute the expected imbalance for the zero key, and we can deduce the expected imbalance for an arbitrary key as $\mathcal{I}_{k_{\text{diff}},k_{\text{lin}}}[\Delta, p, c, c'] = \mathcal{I}_0[\Delta, p \oplus k_{\text{lin}}, c \oplus k_{\text{diff}}, c' \oplus k_{\text{diff}}]$.

Time complexity. This description lead to an attack with low time complexity using an FFT algorithm, as described previously for linear cryptanalysis [16] and multiple linear cryptanalysis [18]. Indeed, the distance between the observed and expected imbalance can be written as:

$$\begin{aligned} L(k) &= \sum_s (\hat{\mathcal{I}}[s] - \mathcal{I}_k[s])^2 \\ &= \sum_s (\hat{\mathcal{I}}[s] - \mathcal{I}_0[s \oplus \phi(k)])^2, \quad \text{where } \phi(k_{\text{diff}}, k_{\text{lin}}) = (0, k_{\text{lin}}, k_{\text{diff}}, k_{\text{diff}}) \\ &= \sum_s \hat{\mathcal{I}}[s]^2 + \sum_s \mathcal{I}_0[s \oplus \phi(k)]^2 - 2 \sum_s \hat{\mathcal{I}}[s] \mathcal{I}_0[s \oplus \phi(k)], \end{aligned}$$

where only the last term depend on the key. Moreover, this term can be seem as the $\phi(k)$-th component of the convolution $\mathcal{I}_0 * \hat{\mathcal{I}}$. Using the convolution theorem, we can compute the convolution efficiently with an FFT algorithm.

This gives the following fast analysis:

1. Compute the expected imbalance $\mathcal{I}_0[s]$ of the differential-linear distinguisher for the zero key, for every subset s.
2. Collect D plaintext-ciphertext pairs, and compute the observed imbalance $\hat{\mathcal{I}}[s]$ of each subset.
3. Compute the convolution $\mathcal{I} * \hat{\mathcal{I}}$, and find k that maximizes coefficient $\phi(k)$.

5.3 Differential-linear Cryptanalysis of Chaskey

In order to find good differential-linear distinguishers for Chaskey, we use a heuristic approach. We know that most good differential characteristics and good linear trails have an "hourglass structure", with a single active bit in the middle. If a good differential-linear characteristics is given with this "hourglass structure", we can divide E in three parts $E = E_\perp \circ E_\intercal \circ E_\top$, so that the single active bit in the differential characteristic falls between $E_\top$ and $E_\intercal$, and the single active bit in the linear trail falls between $E_\intercal$ and $E_\perp$. We use this decomposition to look for good differential-linear characteristics: we first divide E in three parts, and we look for a differential characteristic $\delta_i \xrightarrow{E_\top} \delta_o$ in $E_\top$ (with probability p), a differential-linear characteristic $\delta_o \xrightarrow{E_\intercal} \chi_i$ in $E_\intercal$ (with imbalance b), and a linear characteristic $\chi_i \xrightarrow{E_\perp} \chi_o$ in $E_\perp$ (with imbalance ε), where δ_o and χ_i have a single

active bit. This gives a differential-linear distinguisher with imbalance close to $bp\varepsilon^2$:

- We consider a pair of plaintext (P, P') with $P' = P \oplus \delta_i$, and we denote $X = E_\top(P)$, $Y = E_\mathtt{I}(X)$, $C = E_\bot(Y)$.
- We have $X \oplus X' = \delta_o$ with probability p. In this case, $\varepsilon(Y[\chi_i] \oplus Y'[\chi_i]) = b$
- Otherwise, we expect that $Y[\chi_i] \oplus Y'[\chi_i]$ is not biased. This gives the following:

$$\Pr\left[Y[\chi_i] \oplus Y'[\chi_i] = 0\right] = (1-p) \cdot 1/2 + p \cdot (1/2 + b/2) = 1/2 + bp/2 \quad (10)$$
$$\varepsilon(Y[\chi_i] \oplus Y'[\chi_i]) = bp \quad (11)$$

- We also have $\varepsilon(Y[\chi_i] \oplus C[\chi_o]) = \varepsilon(Y'[\chi_i] \oplus C'[\chi_o]) = \varepsilon$ from the linear approximations. Combining with (11), we get $\varepsilon(C[\chi_o] \oplus C'[\chi_o]) = bp\varepsilon^2$.

In the $E_\mathtt{I}$ section, we can see the characteristic as a small differential-linear characteristic with a single active input bit and a single active output bit, or as a truncated differential where the input difference has a single active bit and the output value is truncated to a single bit. In other words, we use pairs of values with a single bit difference, and we look for a biased output bit difference.

We ran an exhaustive search over all possible decompositions $E = E_\bot \circ E_\mathtt{I} \circ E_\top$ (varying the number of rounds), and all possible positions for the active bits i at the input of $E_\mathtt{I}$ and the biased bit[5] j at the output of $E_\mathtt{I}$. For each candidate, we evaluate experimentally the imbalance $\varepsilon(E_\mathtt{I}(x)[j] \oplus E_\mathtt{I}(x \oplus 2^i)[j])$, and we study the best differential and linear trails to build the full differential-linear distinguisher. This method is similar to the analysis of the Salsa family by Aumasson *et al.* [1]: they decompose the cipher in two parts $E = E_\bot \circ E_\mathtt{I}$, in order to combine a biased bit in $E_\mathtt{I}$ with an approximation of $E_\bot$.

This approach allows to identify good differential-linear distinguisher more easily than by building full differential and linear trails. In particular, we avoid most of the heuristic problems in the analysis of differential-linear distinguishers (such as the presence of multiple good trails in the middle) by evaluating experimentally $\varepsilon(E_\mathtt{I}(x)[j] \oplus E_\mathtt{I}(x \oplus 2^i)[j])$ without looking for explicit trails in the middle. In particular, the transition between $E_\top$ and $E_\mathtt{I}$ is a transition between two differential characteristics, while the transition between $E_\mathtt{I}$ and $E_\bot$ is a transition between two linear characteristics.

5.4 Attack against 6-round Chaskey

The best distinguisher we identified for an attack against 6-round Chaskey uses 1 round in $E_\top$, 4 rounds in $E_\mathtt{I}$, and 1 round in $E_\bot$. The optimal differences and masks are:

- Differential for $E_\top$ with probability $p_\top \approx 2^{-5}$:

$$v_0[26], v_1[26], v_2[6, 23, 30], v_3[23, 30] \xrightarrow{E_\top} v_2[22]$$

[5] We also consider pairs of adjacent bits, following the analysis of [15].

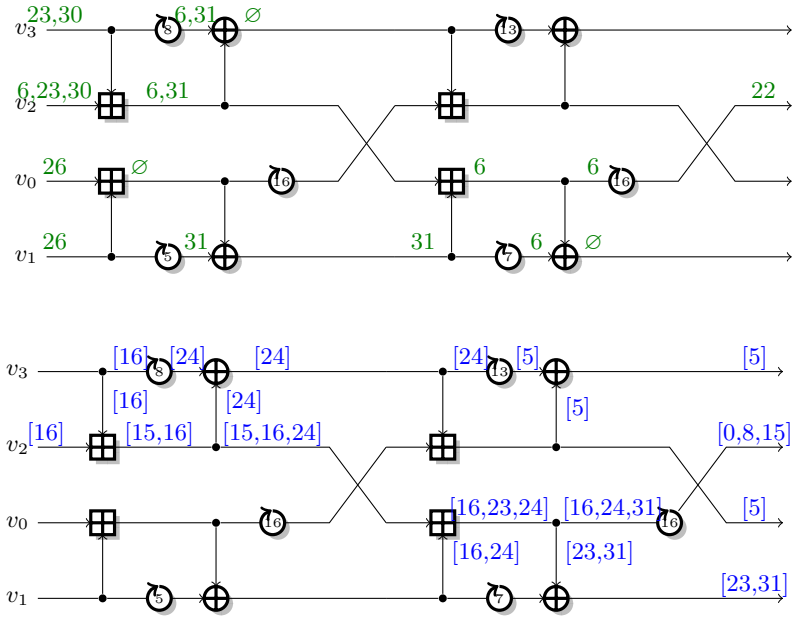

Fig. 7. 6-round attack: differential characteristic, and linear trail.

– Biased bit for $E_\top$ with imbalance $\varepsilon_\top = 2^{-6.05}$:

$$v_2[22] \xrightarrow{E_\top} v_2[16]$$

– Linear approximations for $E_\perp$ with imbalance $\varepsilon_\perp = 2^{-2.6}$:

$$v_2[16] \xrightarrow{E_\perp} v_0[5], v_1[23,31], v_2[0,8,15], v_3[5]$$

The differential and linear trails are shown in Figure 7. The expected imbalance is $p_\top \cdot \varepsilon_\top \cdot \varepsilon_\perp^2 = 2^{-16.25}$. This gives a differential-linear distinguisher with expected complexity in the order of $D = 2/p_\top^2 \varepsilon_\top^2 \varepsilon_\perp^4 \approx 2^{33.5}$.

We can estimate the data complexity more accurately using [12, Eq. (11)]: we need about $2^{34.1}$ pairs of samples in order to reach a false positive rate of 2^{-4}. Experimentally, with 2^{34} pairs of samples (*i.e.* 2^{35} data), the measured imbalance is larger than $2^{-16.25}$ with probability 0.5; with random data, it is larger than $2^{-16.25}$ with probability 0.1. This matches the predictions of [12], and confirms the validity of our differential-linear analysis.

This simple differential-linear attack is more efficient than generic attacks against the Even-Mansour construction of Chaskey. It follows the usage limit of Chaskey, and reaches more rounds than the analysis of the designers. Moreover, it we can be improved significantly using the results of Sections 2 and 3.

Analysis of linear approximations with partitioning. To make the description easier, we remove the linear operations at the end, so that the linear

trail becomes:

$$v_2[16] \xrightarrow{E_\perp} v_1[16,24], v_2[16,23,24], v_3[24]$$

We select control bits to improve the probability of the addition between v_1 and v_2 on active bits 16 and 24. Following the analysis of Section 2.2, we need $v_1[14] \oplus v_2[14]$ and $v_1[15] \oplus v_2[15]$ as control bits for active bit 16. To identify more complex control bits, we consider $v_1[14,15,22,23]$, $v_2[14,15,22,23]$ as potential control bits, as well as $v_3[23]$ because it can affect the addition on the previous half-round. Then, we evaluate the bias experimentally (using the round function as a black box) in order to remove redundant bits. This leads to the following 8 control bits:

$$\begin{array}{llll} v_1[14] \oplus v_2[14] & v_1[14] \oplus v_1[15] & v_1[22] & v_1[23] \\ v_1[15] \oplus v_2[15] & v_1[15] \oplus v_3[23] & v_2[22] & v_2[23] \end{array}$$

This defines 2^8 partitions of the ciphertexts, after guessing 8 key bits. We evaluated the bias in each partition, and we found that the combined capacity is $c^2 = 2^{6.84}$. This means that we have the following complexity ratio

$$R_{\text{lin}}^D = 2^{-2 \cdot 2.6} / 2^{-8} 2^{6.84} \approx 2^{-4} \tag{12}$$

Analysis of differential with partitioning. There are four active bits in the first additions:

- Bit 23 in $v_2 \boxplus v_3$: $(2^{23}, 2^{23}) \xrightarrow{\boxplus} 0$
- Bit 30 in $v_2 \boxplus v_3$: $(2^{30}, 2^{30}) \xrightarrow{\boxplus} 2^{31}$
- Bit 6 in $v_2 \boxplus v_3$: $(2^6, 0) \xrightarrow{\boxplus} 2^6$
- Bit 26 in $v_0 \boxplus v_1$: $(2^{26}, 2^{26}) \xrightarrow{\boxplus} 0$

Following the analysis of Section 3, we can use additional input differences for each of them. However, we reach a better trade-off by selected only three of them. More precisely, we consider 2^3 input differences, defined by δ_i and the following extra active bits:

$$v_2[24] \qquad v_2[31] \qquad v_0[27]$$

As explained in Section 2, we build structures of 2^4 plaintexts, where each structure provides 2^3 pairs for every input difference, *i.e.* 2^6 pairs in total.

Following the analysis of Section 3, we use the following control bits to improve the probability of the differential:

$$\begin{array}{lll} v_2[23] \oplus v_2[24] & v_2[30] \oplus v_3[30] & v_0[26] \oplus v_0[27] \\ v_2[24] \oplus v_3[23] & & v_0[27] \oplus v_1[26] \end{array}$$

This divides each set of pairs into 2^5 subsets, after guessing 5 key bits. In total we have 2^8 subsets to analyze, according to the control bits and the multiple

differentials. We found that, for 18 of those subsets, there is a probability 2^{-2} to reach δ_o (the probability is 0 for the remaining subsets). This leads to a complexity ratio:

$$R^D_{\text{diff}} = \frac{2 \cdot 2^{-5}}{18/2^8 \times 2^4 \times 2^{-2}} = 2/9$$
$$R^D_{\text{diff-2}} = \frac{2 \cdot 2^{2\times -5}}{18/2^8 \times 2^4 \times 2^{2\times -2}} = 1/36$$

This corresponds to the analysis of Section 3: we have a ratio of 2/3 for bits $v_2[23]$ and $v_0[27]$ (Section 3.1), and a ratio of 1/2 for $v_2[31]$ in the simple linear case. In the differential-linear case, we have respectively ratios of 1/3 and 1/4.

Finally, the improved attack requires a data complexity in the order of:

$$R^{D\ 2}_{\text{lin}} R^D_{\text{diff-2}} D \approx 2^{20.3}.$$

We can estimate the data complexity more accurately using the analysis of Biryukov, De Cannière and Quisquater [9]. First, we give an alternate description of the attack similar the multiple linear attack framework. Starting from D chosen plaintexts, we build 2^2D pairs using structures, and we keep $N = 18 \cdot 2^{-8} \cdot 2^{-14} \cdot 2^2 D$ samples per approximation after partitioning the differential and linear parts. The imbalance of the distinguisher is $2^{-2} \cdot 2^{-6.05} \cdot 2^{6.84} = 2^{-1.21}$. Following [9, Corollary 1], the gain of the attack with $D = 2^{24}$ is estimated as 6.6 bits, *i.e.* the average key rank should be about 42 (for the 13-bit subkey).

Using the FFT method of Section 5.2, we perform the attack with 2^{24} counters $\hat{\mathcal{I}}[s]$. Each structure of 2^4 plaintexts provides 2^6 pairs, so that we need 2^2D operations to update the counters. Finally, the FFT computation require $24 \times 2^{24} \approx 2^{28.6}$ operations.

We have implemented this analysis, and it runs in about 10s on a single core of a desktop PC.[6] Experimentally, we have a gain of about 6 bits (average key rank of 64 with 128 experiments); this validates our theoretical analysis. We also notice some key bits don't affect the distinguisher and cannot be recovered. On the other hand, the gain of the attack can be improved using more data, and further trade-offs are possible using larger or smaller partitions.

5.5 Attack against 7-round Chaskey

The best distinguisher we identified for an attack against 7-round Chaskey uses 1.5 round in $E_\top$, 4 rounds in $E_\perp$, and 1.5 round in $E_\perp$. The optimal differences and masks are:

- Differential for $E_\top$ with probability $p_\top = 2^{-17}$:
  $v_0[8, 18, 21, 30], v_1[8, 13, 21, 26, 30], v_2[3, 21, 26], v_3[21, 26, 27] \xrightarrow{E_\top} v_0[31]$

[6] Haswell microarchitecture running at 3.4 GHz

– Biased bit for $E_\top$ with imbalance $\varepsilon_\top = 2^{-6.1}$:
  $v_0[31] \xrightarrow{E_\top} v_2[20]$
– Linear approximations for $E_\perp$ with imbalance $\varepsilon_\perp = 2^{-7.6}$:
  $v_2[20] \xrightarrow{E_\perp} v_0[0, 15, 16, 25, 29], v_1[7, 11, 19, 26], v_2[2, 10, 19, 20, 23, 28], v_3[0, 25, 29]$

This gives a differential-linear distinguisher with expected complexity in the order of $D = 2/p_\top^2 \varepsilon_\top^2 \varepsilon_\perp^4 \approx 2^{77.6}$. This attack is more expensive than generic attacks against on the Even-Mansour cipher, but we now improve it using the results of Sections 2 and 3.

Analysis of linear approximations with partitioning. We use an automatic search to identify good control bits, starting from the bits suggested by the result of Section 2. We identified the following control bits:

$v_1[3] \oplus v_1[11] \oplus v_3[10]$	$v_1[3] \oplus v_1[11] \oplus v_3[11]$	$v_0[15] \oplus v_3[14]$
$v_0[15] \oplus v_3[15]$	$v_1[11] \oplus v_1[18] \oplus v_3[17]$	$v_1[11] \oplus v_1[18] \oplus v_3[18]$
$v_1[3] \oplus v_2[2]$	$v_1[3] \oplus v_2[3]$	$v_1[11] \oplus v_2[9]$
$v_1[11] \oplus v_2[10]$	$v_1[11] \oplus v_2[11]$	$v_1[18] \oplus v_2[17]$
$v_1[18] \oplus v_2[18]$	$v_1[2] \oplus v_1[3]$	$v_1[9] \oplus v_1[11]$
$v_1[10] \oplus v_1[11]$	$v_1[17] \oplus v_1[18]$	$v_0[14] \oplus v_0[15]$
$v_0[15] \oplus v_1[3] \oplus v_1[11] \oplus v_1[18]$		

Note that the control bits identified in Section 2 appear as linear combinations of those control bits.

This defines 2^{19} partitions of the ciphertexts, after guessing 19 key bits. We evaluated the bias in each partition, and we found that the combined capacity is $c^2 = 2^{14.38}$. This means that we gain the following factor:

$$R_{\text{lin}}^D = 2^{-2 \cdot 7.6} / 2^{-19} 2^{14.38} \approx 2^{-10.5} \tag{13}$$

This example clearly shows the power of the partitioning technique: using a few key guesses, we essentially avoid the cost of the last layer of additions.

Analysis of differential with partitioning. We consider 2^9 input differences, defined by δ_i and the following extra active bits:

$v_0[9]$	$v_0[22]$	$v_0[31]$	$v_0[19]$	
$v_0[14]$	$v_0[27]$	$v_2[22]$	$v_2[27]$	$v_2[4]$

As explained in Section 2, we build structures of 2^{10} plaintexts, where each structure provides 2^9 pairs for every input difference, *i.e.* 2^{18} pairs in total.

Again, we use an automatic search to identify good control bits, starting from the bits suggested in Section 3. We use the following control bits to improve the

probability of the differential:

$v_0[4] \oplus v_2[3]$	$v_2[22] \oplus v_3[21]$	$v_2[27] \oplus v_3[26]$	$v_2[27] \oplus v_3[27]$
$v_2[3] \oplus v_2[4]$	$v_2[21] \oplus v_2[22]$	$v_2[26] \oplus v_2[27]$	$v_0[9] \oplus v_1[8]$
$v_0[14] \oplus v_1[13]$	$v_0[27] \oplus v_1[26]$	$v_0[30] \oplus v_1[30]$	$v_0[8] \oplus v_0[9]$
$v_0[18] \oplus v_0[19]$	$v_0[21] \oplus v_0[22]$		

This divides each set of pairs into 2^{14} subsets, after guessing 14 key bits. In total we have 2^{23} subsets to analyze, according to the control bits and the multiple differentials. We found that, for 17496 of those subsets, there is a probability 2^{-11} to reach δ_o (the probability is 0 for the remaining subsets). This leads to a ratio:

$$R^D_{\text{diff-2}} = \frac{2 \cdot 2^{-2 \cdot 17}}{17496/2^{23} \times 2^{10} \times 2^{-2 \cdot 11}} = 1/4374 \approx 2^{-12.1}$$

Finally, the improved attack requires a data complexity of:

$$R^D_{\text{lin}}{}^2 R^D_{\text{diff-2}} D \approx 2^{44.5}.$$

Again, we can estimate the data complexity more accurately using [9]. In this attack, starting from N_0 chosen plaintexts, we build $2^8 N_0$ pairs using structures, and we keep $N = 17496 \cdot 2^{-23} \cdot 2^{-38} \cdot 2^8 N_0$ samples per approximation after partitioning the differential and linear parts. The imbalance of the distinguisher is $2^{-11} \cdot 2^{-6.1} \cdot 2^{14.38} = 2^{-2.72}$. Following [9, Corollary 1], the gain of the attack with $N_0 = 2^{48}$ is estimated as 6.3 bits, *i.e.* the average rank of the 33-bit subkey should be about $2^{25.7}$. Following the experimental results of Section 5.4, we expect this to estimation to be close to the real gain (the gain can also be increased if more than 2^{48} data is available).

Using the FFT method of Section 5.2, we perform the attack with 2^{61} counters $\hat{\mathcal{I}}[s]$. Each structure of 2^{10} plaintexts provides 2^{18} pairs, so that we need $2^8 D$ operations to update the counters. Finally, the FFT computation require $61 \times 2^{61} \approx 2^{67}$ operations.

This attack recovers only a few bits of a 33-bit subkey, but an attacker can run the attack again with a different differential-linear distinguisher to recover other key bits. For instance, a rotated version of the distinguisher will have a complexity close to the optimal one, and the already known key bits can help reduce the complexity.

Conclusion

In this paper, we have described a partitioning technique inspired by Biham and Carmeli's work. While Biham and Carmeli consider only two partitions and a linear approximation for a single subset, we use a large number of partitions, and linear approximations for every subset to take advantage of all the data. We also introduce a technique combining multiple differentials, structures, and

partitioning for differential cryptanalysis. This allows a significant reduction of the data complexity of attacks against ARX ciphers, and is particularly efficient with boomerang and differential-linear attacks.

Our main application is a differential-linear attack against Chaskey, that reaches 7 rounds out of 8. In this application, the partitioning technique allows to go through the first and last additions almost for free. This is very similar to the use of partial key guess and partial decryption for SBox-based ciphers. This is an important result because standard bodies (ISO/IEC JTC1 SC27 and ITU-T SG17) are currently considering Chaskey for standardization, but little external cryptanalysis has been published so far. After the first publications of these results, the designers of Chaskey have proposed to standardize a new version with 12 rounds [31].

Acknowledgement

We would like to thank Nicky Mouha for enriching discussions about those results, and the anonymous reviewers for their suggestions to improve the presentation of the paper.

The author is partially supported by the French Agence Nationale de la Recherche through the BRUTUS project under Contract ANR-14-CE28-0015.

A Appendix: Application to FEAL-8X

We now present application of our techniques to reduce the data complexity of differential and linear attacks.

FEAL is an early block cipher proposed by Shimizu and Miyaguchi in 1987 [37]. FEAL uses only addition, rotation and xor operations, which makes it much more efficient than DES in software. FEAL has inspired the development of many cryptanalytic techniques, in particular linear cryptanalysis.

At the rump session of CRYPTO 2012, Matsui announced a challenge for low data complexity attacks on FEAL-8X using only known plaintexts. At the time, the best practical attack required 2^{24} known plaintexts [2] (Matsui and Yamagishi had non-practical attacks with as little as 2^{14} known plaintext [27]), but Biham and Carmeli won the challenge with a new linear attack using 2^{15} known plaintexts, and introduced the partitioning technique to reduce the data complexity to 2^{14} [3]. Later Sakikoyama *et al.* improved this result using multiple linear cryptanalysis, with a data complexity of only 2^{12} [35].

We now explain how to apply the generalized partitioning to attack FEAL-8X. Our attack follows the attack of Biham and Carmeli [3], and uses the generalized partitioning technique to reduce the data complexity further. The attack by Biham and Carmeli requires 2^{14} data and about 2^{45} time, while our attack needs only 2^{12} data, and 2^{45} time. While the attack of Sakikoyama *et al.* is more efficient with the same data complexity, this shows a simple example of application of the generalized partitioning technique.

The attacks are based on a 6-round linear approximation with imbalance 2^{-5}, using partial encryption for the first round (with a 15 bit key guess), and partial decryption for the last round (with a 22 bit key guess). This allows to compute enough bits of the state after the first round and before the last round, respectively, to compute the linear approximation. For more details of the attack, we refer the reader to the description of Biham and Carmeli [3].

In order to improve the attack, we focus on the round function of the second-to-last round. The corresponding linear approximation is $x[\texttt{10115554}] \rightarrow y[\texttt{04031004}]$ with imbalance of approximately 2^{-3}.

We partition the data according to the following 4 bits[7] (note that all those bits can be computed in the input of round 6 with the 22-bit key guess of $DK7$):

$$b_0 = f_{0,3} \oplus f_{1,3} \oplus f_{2,2} \oplus f_{3,2} \qquad b_1 = f_{0,3} \oplus f_{1,3} \oplus f_{2,3} \oplus f_{3,3}$$
$$b_2 = f_{0,3} \oplus f_{1,3} \oplus f_{2,5} \oplus f_{3,5} \qquad b_3 = f_{0,2} \oplus f_{1,2} \oplus f_{0,3} \oplus f_{1,3}$$

The probability of the linear approximation in each subset is as follows (indexed by the value of b_3, b_2, b_1, b_0):

$$p_{0000} = 0.250 \quad p_{0001} = 0.270 \quad p_{0010} = 0.531 \quad p_{0011} = 0.746$$
$$p_{0100} = 0.406 \quad p_{0101} = 0.699 \quad p_{0110} = 0.750 \quad p_{0111} = 0.652$$
$$p_{1000} = 0.254 \quad p_{1001} = 0.469 \quad p_{1010} = 0.730 \quad p_{1011} = 0.750$$
$$p_{1100} = 0.652 \quad p_{1101} = 0.750 \quad p_{1110} = 0.699 \quad p_{1111} = 0.406$$

This gives a total capacity $c^2 = \sum_i (2 \cdot p_i - 1)^2 = 2.49$, using subsets of $1/16$ of the data. For reference, a linear attack without partitioning has a capacity $(2^{-3})^2$, therefore the complexity ratio can be computed as:

$$R^D_{\text{lin}} = 2^{-6}/(1/16 \times 2.49) \approx 1/10$$

This can be compared to Biham and Carmeli's partitioning, where they use a single linear approximation with capacity 0.1 for $1/2$ of the data, this gives a ratio of only:

$$R^D_{\text{lin}} = 2^{-6}/(1/2 \times 0.1) \approx 1/3.2$$

With a naive implementation of this attack, we have to repeat the analysis 16 times, for each guess of 4 key bits. Since the data is reduced by a factor 4, the total time complexity increases by a factor 4 compared to the attack on Biham and Carmeli. This result in an attack with 2^{12} data and 2^{47} time.

However, the time complexity can also be reduced using counters, because the 4 extra key bits only affect the choice of the partitions. This leads to an attack with 2^{12} data and 2^{43} time.

References

1. Aumasson, J.P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In: Nyberg, K. (ed.) FSE. Lecture Notes in Computer Science, vol. 5086, pp. 470–488. Springer (2008)

[7] We use Biham and Carmeli's notation $f_{i,j}$ for bit j of input word i

2. Biham, E.: On Matsui's Linear Cryptanalysis. In: Santis, A.D. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 950, pp. 341–355. Springer (1994)
3. Biham, E., Carmeli, Y.: An Improvement of Linear Cryptanalysis with Addition Operations with Applications to FEAL-8X. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8781, pp. 59–76. Springer (2014)
4. Biham, E., Chen, R., Joux, A.: Cryptanalysis of SHA-0 and Reduced SHA-1. J. Cryptology 28(1), 110–160 (2015)
5. Biham, E., Dunkelman, O., Keller, N.: Enhancing Differential-Linear Cryptanalysis. In: Zheng, Y. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 2501, pp. 254–266. Springer (2002)
6. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990)
7. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
8. Biham, E., Shamir, A.: Differential Cryptanalysis of Feal and N-Hash. In: Davies, D.W. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 547, pp. 1–16. Springer (1991)
9. Biryukov, A., Cannière, C.D., Quisquater, M.: On Multiple Linear Approximations. In: Franklin, M.K. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 3152, pp. 1–22. Springer (2004)
10. Biryukov, A., Cannière, C.D., Quisquater, M.: On Multiple Linear Approximations. IACR Cryptology ePrint Archive, Report 2004/57 (2004)
11. Biryukov, A., Nikolic, I., Roy, A.: Boomerang Attacks on BLAKE-32. In: Joux, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 6733, pp. 218–237. Springer (2011)
12. Blondeau, C., Gérard, B., Tillich, J.P.: Accurate estimates of the data complexity and success probability for various cryptanalyses. Des. Codes Cryptography 59(1-3), 3–34 (2011)
13. Blondeau, C., Leander, G., Nyberg, K.: Differential-Linear Cryptanalysis Revisited. In: Cid, C., Rechberger, C. (eds.) FSE. Lecture Notes in Computer Science, vol. 8540, pp. 411–430. Springer (2014)
14. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES. Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer (2007)
15. Cho, J.Y., Pieprzyk, J.: Crossword Puzzle Attack on NLS. In: Biham, E., Youssef, A.M. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 4356, pp. 249–265. Springer (2006)
16. Collard, B., Standaert, F.X., Quisquater, J.J.: Improving the Time Complexity of Matsui's Linear Cryptanalysis. In: Nam, K.H., Rhee, G. (eds.) ICISC. Lecture Notes in Computer Science, vol. 4817, pp. 77–88. Springer (2007)
17. Gilbert, H., Chassé, G.: A Statistical Attack of the FEAL-8 Cryptosystem. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO. Lecture Notes in Computer Science, vol. 537, pp. 22–33. Springer (1990)
18. Hermelin, M., Nyberg, K.: Dependent Linear Approximations: The Algorithm of Biryukov and Others Revisited. In: Pieprzyk, J. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 5985, pp. 318–333. Springer (2010)
19. Huang, T., Tjuawinata, I., Wu, H.: Differential-linear cryptanalysis of icepole. In: FSE 2015 (2015)

20. Lamberger, M., Mendel, F.: Higher-Order Differential Attack on Reduced SHA-256. IACR Cryptology ePrint Archive, Report 2011/37 (2011)
21. Langford, S.K., Hellman, M.E.: Differential-Linear Cryptanalysis. In: Desmedt, Y. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 839, pp. 17–25. Springer (1994)
22. Leurent, G.: Analysis of Differential Attacks in ARX Constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT. Lecture Notes in Computer Science, vol. 7658, pp. 226–243. Springer (2012)
23. Leurent, G.: Construction of Differential Characteristics in ARX Designs Application to Skein. In: Canetti, R., Garay, J.A. (eds.) CRYPTO (1). Lecture Notes in Computer Science, vol. 8042, pp. 241–258. Springer (2013)
24. Leurent, G., Roy, A.: Boomerang Attacks on Hash Function Using Auxiliary Differentials. In: Dunkelman, O. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 7178, pp. 215–230. Springer (2012)
25. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Matsui, M. (ed.) FSE. Lecture Notes in Computer Science, vol. 2355, pp. 336–350. Springer (2001)
26. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993)
27. Matsui, M., Yamagishi, A.: A New Method for Known Plaintext Attack of FEAL Cipher. In: Rueppel, R.A. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 658, pp. 81–91. Springer (1992)
28. Mavromati, C.: Key-recovery attacks against the mac algorithm chaskey. In: SAC 2015 (2015)
29. Mendel, F., Nad, T.: Boomerang Distinguisher for the SIMD-512 Compression Function. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 7107, pp. 255–269. Springer (2011)
30. Miyano, H.: Addend dependency of differential/linear probability of addition. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences 81(1), 106–109 (1998)
31. Mouha, N.: Chaskey: a MAC Algorithm for Microcontrollers - Status Update and Proposal of Chaskey-12 -. IACR Cryptology ePrint Archive, Report 2015/1182 (2015)
32. 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.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8781, pp. 306–323. Springer (2014)
33. Mouha, N., Velichkov, V., Cannière, C.D., Preneel, B.: The Differential Analysis of S-Functions. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 6544, pp. 36–56. Springer (2010)
34. Nyberg, K., Wallén, J.: Improved Linear Distinguishers for SNOW 2.0. In: Robshaw, M.J.B. (ed.) FSE. Lecture Notes in Computer Science, vol. 4047, pp. 144–162. Springer (2006)
35. Sakikoyama, S., Todo, Y., Aoki, K., Morii, M.: How Much Can Complexity of Linear Cryptanalysis Be Reduced? In: Lee, J., Kim, J. (eds.) ICISC. Lecture Notes in Computer Science, vol. 8949, pp. 117–131. Springer (2014)
36. Sasaki, Y.: Boomerang Distinguishers on MD4-Family: First Practical Results on Full 5-Pass HAVAL. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7118, pp. 1–18. Springer (2011)

37. Shimizu, A., Miyaguchi, S.: Fast Data Encipherment Algorithm FEAL. In: Chaum, D., Price, W.L. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 304, pp. 267–278. Springer (1987)
38. Tardy-Corfdir, A., Gilbert, H.: A Known Plaintext Attack of FEAL-4 and FEAL-6. In: Feigenbaum, J. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 576, pp. 172–181. Springer (1991)
39. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer (1999)
40. Wallén, J.: Linear Approximations of Addition Modulo 2^n. In: Johansson, T. (ed.) FSE. Lecture Notes in Computer Science, vol. 2887, pp. 261–273. Springer (2003)
41. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 3621, pp. 17–36. Springer (2005)
42. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 3494, pp. 19–35. Springer (2005)
43. Yu, H., Chen, J., Wang, X.: The Boomerang Attacks on the Round-Reduced Skein-512. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 287–303. Springer (2012)