



HAL
open science

Differential and Linear Cryptanalysis of ARX with Partitioning

Gaëtan Leurent

► **To cite this version:**

Gaëtan Leurent. Differential and Linear Cryptanalysis of ARX with Partitioning: Application to FEAL and Chaskey. 2015. hal-01243166

HAL Id: hal-01243166

<https://inria.hal.science/hal-01243166>

Preprint submitted on 14 Dec 2015

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.

Differential and Linear Cryptanalysis of ARX with Partitioning

Application to FEAL and Chaskey

Gaëtan Leurent

Inria, France

Abstract. In this work, we refine a partitioning technique recently proposed by Biham and Carmeli to improve the linear cryptanalysis of addition operations, and we propose an analogue improvement of differential cryptanalysis of addition operations. These two techniques can reduce the data complexity of linear and differential attacks, at the cost of more processing time. Our technique can be seen as the analogue for ARX ciphers of partial key guess and partial decryption for SPN ciphers. We show a first application of the generalized linear partitioning technique on FEAL-8X, revisiting the attack of Biham and Carmeli. We manage to reduce the data complexity from 2^{14} to 2^{12} known plaintexts, while the time complexity increases from 2^{45} to 2^{47} .

Then, we use these techniques to analyze Chaskey, a recent MAC proposal by Mouha *et al.*, that is being studied for standardisation by ISO and ITU-T. Chaskey uses an ARX structure very similar to SipHash. We use a differential-linear attack with improvements from the partitioning technique, combined with a convolution-based method to reduce the time complexity. This leads to an attack on 6 rounds with 2^{25} data and $2^{28.6}$ time (verified experimentally), and an attack on 7 rounds with 2^{48} data and 2^{67} time. These results show that the full version of Chaskey with 8 rounds has a rather small security margin.

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 [29,28], 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 that 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:

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

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 $C'_i \oplus C_i = \delta_C$. 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 imbalance¹ ε is non-negligible:

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

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

$$\hat{\varepsilon} = |2 \cdot \# \{i : P[\chi_P] = C[\chi_C]\} / D - 1|.$$

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 SPN 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 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,

¹ also called correlation

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 [42,38,27,36,44,33,15], and differential and linear cryptanalysis have been used to analyse many ARX designs (see for instance the following papers: [4,46,45,24,25,18,8,29]).

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 SPN designs where it is sometimes possible to build iterative trails, or trails with only a few active SBoxes per round. For instance, this is the 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 [43] and differential-linear cryptanalysis [23,5]. 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).

Our results. In this paper we study in detail how key bit guesses can be used to analyze ARX ciphers. 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.

We then demonstrate our techniques with applications to FEAL-8X and Chaskey. We show that this results in a significant reduction of the data complexity, in exchange for some more processing time (for a fixed number of rounds). Alternatively, we can extend the attack to a larger number of rounds with a small increase of the 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

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

Table 1. Key-recovery attacks on Chaskey

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

Table 2. Known-plaintext attacks on FEAL-8X

Data	Time	Ref.
2^{14}	2^{64}	[29]
2^{24}	2^{24}	[2,39]
2^{14}	2^{45}	[3]
2^{10}	2^{62}	[3]
2^{12}	2^{47}	Section 4

distinguisher into a key recovery attack, and we can add some rounds to the distinguisher.

Reducing the data complexity at the expense of time complexity is interesting because the data complexity of an attack is often the limiting factor: in practice, it is quite easy to perform 2^{50} computations, but collecting (and storing) 2^{50} known or chosen plaintext is virtually impossible. Moreover, the reduction in data complexity can actually lead to a reduction in time complexity. Our techniques increases the processing time per plaintext, but since the number of required plaintext is reduced, the total complexity can in fact be reduced. In particular, this is the case of our attacks against Chaskey.

For FEAL-8X, we manage to reduce the data complexity of the best known plaintext attack with practical time complexity from 2^{14} to 2^{12} . For Chaskey, we show how to build efficient differential-linear distinguishers, to reduce the data complexity with the partitioning technique, and to reduce the time complexity using a convolution method. For 7 rounds of Chaskey, the resulting differential-linear distinguisher requires 2^{78} data and 2^{78} time, but this can be reduced to 2^{48} data and 2^{67} time, using partitioning. In addition, this result is quite interesting because standard bodies (ISO/IEC JTC1 SC27 and ISO/IEC JTC1 SC27) are currently conducting a preliminary study on the standardization of Chaskey [34], but little external cryptanalysis has been published so far. The full version of Chaskey has 8 rounds, and is claimed to be secure against attacks with 2^{48} data and 2^{80} time. Our attacks are summarized in Table 1 and 2.

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. Then we demonstrate the application of our techniques, to FEAL in Section 4 and to Chaskey in Section 5.

2 Linear Analysis of Addition

We discuss linear cryptanalysis applied to addition operations, and our improvement using partitioning. We describe the linear approximations using linear masks; for instance an approximation for E is written as $\Pr[E(x)[\chi'] = x[\chi]] = 1/2 \pm \varepsilon/2$ where χ and χ' are the input and output linear masks ($x[\chi]$ denotes $x[\chi_1] \oplus x[\chi_2] \oplus \dots \oplus x[\chi_\ell]$, and $x[\chi_i]$ is bit χ_i of x), and $\varepsilon > 0$ is the imbalance. We also denote the imbalance of a random variable x as $\varepsilon(x) = |2 \cdot \Pr[x = 0] - 1|$, and $\mathcal{I}(x) = |\varepsilon(x)|$. 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:

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

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

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

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 [44]. In the following we study the case $i > 0$, where the linear approximation (2) for E has imbalance $\varepsilon/2$.

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, \dots, 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 $c_0 = 0$, $c_{i+1} = \text{MAJ}(a_i, b_i, c_i)$ (where $\text{MAJ}(a, b, c) = (a \wedge b) \vee (b \wedge c) \vee (c \wedge a)$). Therefore, we have $s_i = a_i \oplus b_i \oplus c_i$.

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 they can still take advantage of partitioning techniques.

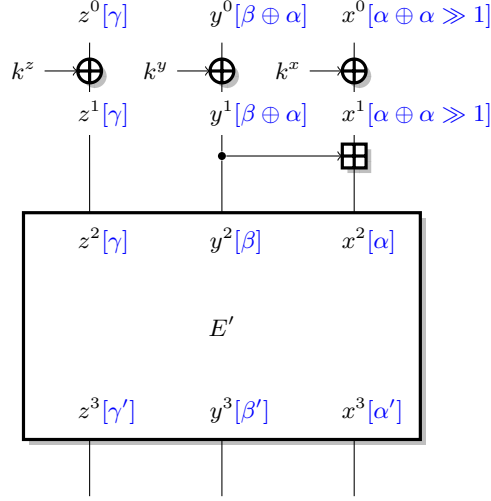


Fig. 1. Linear attack against the first addition

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 $\mathcal{I}(s_i \oplus a_i \oplus a_{i-1}) = 0$.
- 10 If $(a_{i-1}, b_{i-1}) = (1, 0)$, then $\varepsilon(c_i) = 0$ and $\mathcal{I}(s_i \oplus a_i \oplus a_{i-1}) = 0$.
- 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:

$$s_i = a_i \oplus b_i \oplus a_{i-1} \quad \text{if } a_{i-1} = b_{i-1} \quad (3)$$

In order to apply this analysis to the setting of Figure 1, we guess the key bits k_{i-1}^x and k_{i-1}^y , so that we can compute the values of x_{i-1}^1 and y_{i-1}^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):

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

If we guess the key bit $k_{i-1}^x \oplus k_{i-1}^y$, we can filter the data satisfying $x_{i-1}^0 \oplus y_{i-1}^0 = k_{i-1}^x \oplus k_{i-1}^y$, and we have $\mathcal{I}(x_i^2 \oplus x_i^0 \oplus y_i^0 \oplus x_{i-1}^0) = 1$. 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.

In general we can compute the reduction of the data complexity reached by this kind of technique over a regular linear analysis as

$$G_{\text{lin}} = \mu \times \hat{\varepsilon}^2 / \varepsilon^2$$

where μ is the fraction of data used in the attack, ε is the initial imbalance, and $\hat{\varepsilon}$ is the improved imbalance for the selected subset. For Biham and Carmeli's attack, we have $\mu = 1/2$ and $\hat{\varepsilon} = 2\varepsilon$, hence $G_{\text{lin}} = 2$.

2.2 Generalized partitioning

$$\begin{array}{ccccc}
\begin{array}{c} \color{red}{0} \\ \color{red}{\leftarrow} \\ ? x_i 0 ? ? \end{array} &
\begin{array}{c} \color{red}{1} \\ \color{red}{\leftarrow} \\ ? x_i 1 ? ? \end{array} &
\begin{array}{c} \color{red}{0\ 0} \\ \color{red}{\leftarrow} \\ ? x_i 0 0 ? \end{array} &
\begin{array}{c} \color{red}{1\ 1} \\ \color{red}{\leftarrow} \\ ? x_i 0 1 ? \end{array} &
\begin{array}{c} \color{red}{? ?} \\ \color{red}{\leftarrow} \\ ? x_i 1 0 ? \end{array} \\
+ ? y_i 0 ? ? & + ? y_i 1 ? ? & + ? y_i 1 0 ? & + ? y_i 1 1 ? & + ? y_i 0 1 ? \\
\hline
? a_i ? ? ? & ? a_i ? ? ? & ? a_i ? ? ? & ? a_i ? ? ? & ? a_i ? ? ?
\end{array}$$

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

$$s_i = \begin{cases} a_i \oplus b_i \oplus a_{i-1} & \text{if } a_{i-1} = b_{i-1} \\ a_i \oplus b_i \oplus a_{i-2} & \text{if } a_{i-1} \neq b_{i-1} \text{ and } a_{i-2} = b_{i-2} \end{cases} \quad (4)$$

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

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

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 $x_{i-1}^0 \oplus y_{i-1}^0$ and $x_{i-2}^0 \oplus y_{i-2}^0$, and we compute the measured (signed) imbalance $\hat{\varepsilon}_s$ of each subset. Then, for each guess of the key bits $k_{i-1}^x \oplus k_{i-1}^y$, and $k_{i-2}^x \oplus k_{i-2}^y$, we deduce the expected imbalance ε_s^k of each subset, and we compute the distance to the observed imbalance as $\sum_s (\hat{\varepsilon}_s - \varepsilon_s^k)^2$. 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 $c^2 = \sum_i \varepsilon_i^2$ 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 gain of this analysis can be computed as $G_{\text{lin}} = \mu \times \hat{\varepsilon}^2/\varepsilon^2$. With $\mu = 3/4$ and $\hat{\varepsilon} = 2\varepsilon$, we find $G_{\text{lin}} = 3$.

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

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 G_{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 Sections 4 and 5.

3 Differential Analysis of Addition

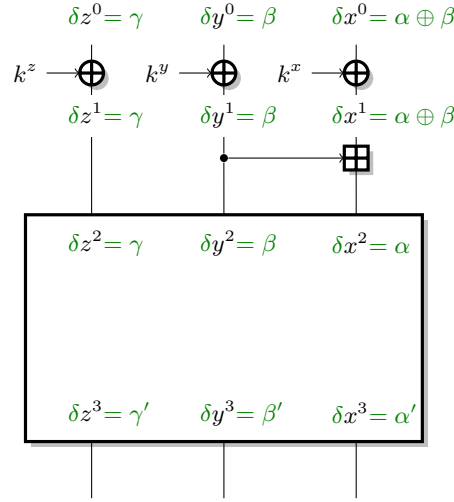


Fig. 3. Differential attack against the first addition

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 $(\alpha, \beta, \gamma) \rightarrow (\alpha', \beta', \gamma')$ with probability p for the remaining of the cipher. Following previous works, the simple way to extend the differential is to use 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

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

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:

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

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:

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

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:

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

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 we can compute the gain of this technique as

$$G_{\text{diff}} = \mu T / 2 \times \hat{p} / p$$

where μ is the fraction of data used in the attack, 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 \hat{p} is the improved probability for the selected subset. Here we have $\mu = 3/4$, $T = 4$, and $\hat{p} = p$, hence $G_{\text{diff}} = 3/2$.

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

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

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

This gives an attack with $T = 4$, $\mu = 3/8$ and $\hat{p} = 2p$, which yield the same gain $G_{\text{diff}} = 3/2$. However, this analysis allows to recover an extra key bit, which can be useful for further steps of an attack.

Moreover, if the differential trail is used in a boomerang attack, or in a differential-linear attack, the gain should be evaluated as:

$$G_{\text{diff-2}} = \mu T / 2 \times (\hat{p}/p)^2.$$

In this context, the partitioning with a single control bit still has a gain of $G_{\text{diff-2}} = 3/2$, but the partitioning with two control bits has a gain of $G_{\text{diff-2}} = 3$.

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

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

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:

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

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:

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

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 have $G_{\text{diff}} = 3/2$, with $\mu = 3/4$, $T = 4$, and $\hat{p} = p$.

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

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

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

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

This gives:

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

Unfortunately, we can only use a small fraction of the pairs $\mu = 3/16$. With $T = 4$ and $\hat{p} = 2p$, this yields $G_{\text{diff}} = 3/4$, which is worse than in the previous section.

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:

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

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 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 [41]. 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 [29]), 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].

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^{47} time.

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 $x[10115554] \rightarrow y[04031004]$ with imbalance of approximately 2^{-3} .

We partition the data according to the following 4 bits⁵ (note that all those bits can be computed in the input of round 6 with the 22-bit key guess of $DK7$):

$$\begin{aligned} b_0 &= f_{0,3} \oplus f_{1,3} \oplus f_{2,2} \oplus f_{3,2} & b_1 &= f_{0,3} \oplus f_{1,3} \oplus f_{2,3} \oplus f_{3,3} \\ b_2 &= f_{0,3} \oplus f_{1,3} \oplus f_{2,5} \oplus f_{3,5} & b_3 &= f_{0,2} \oplus f_{1,2} \oplus f_{0,3} \oplus f_{1,3} \end{aligned}$$

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):

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

⁵ We use Biham and Carmeli's notation $f_{i,j}$ for bit j of input word i

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 gain can be computed as:

$$G_{\text{lin}} = 1/16 \times 2.49/2^{-6} \approx 10$$

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 give a gain of only:

$$G_{\text{lin}} = 1/2 \times 0.1/2^{-6} \approx 3.2$$

On the other hand the time complexity is increased because we have to guess 4 more key bits, and repeat the analysis 16 times. 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.

5 Application to Chaskey

Chaskey is a recent MAC proposal designed jointly by researchers from COSIC and Hitachi [35]. 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.

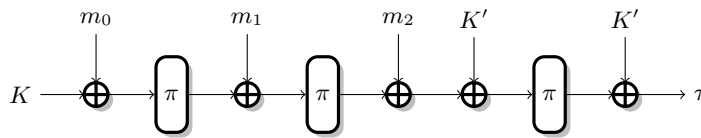


Fig. 4. Chaskey mode of operation (full block message)

So far, the only external cryptanalysis results on Chaskey are generic attacks in the multi-user setting [30]. 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 standardisation.

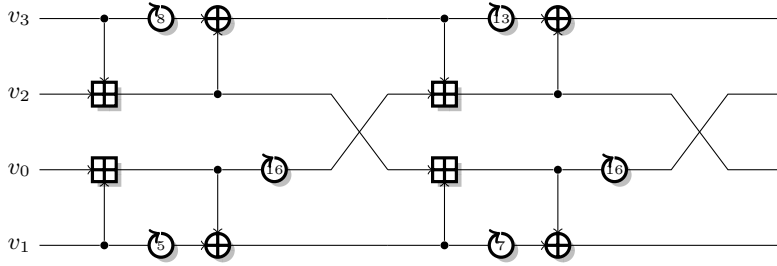


Fig. 5. One round of the Chaskey permutation. The full permutation has 8 rounds.

Table 3. Probabilities of the best differential characteristics of Chaskey reported by the designers [35]

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

5.1 Differential-linear cryptanalysis

The best differential characteristics found by the designers of Chaskey quickly become unusable when the number of rounds increase (See Table 3). 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 difference 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 [47,31,11,26,22,40] or differential-linear cryptanalysis [20]. Since Chaskey never uses the inverse permutation, we cannot apply a boomerang attack, and we focus on 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 $|2 \cdot \#\{i : C[\chi_o] = C'[\chi_o]\} / D - 1|$. 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 $\delta_o[\chi_i] = 0$.

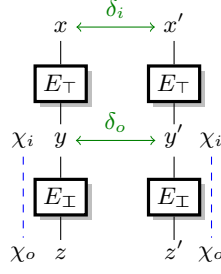


Fig. 6. Differential-linear cryptanalysis

- Otherwise, we expect that $E_{\top}(P)[\chi_i] \oplus E_{\top}(P')[\chi_i]$ is not biased. This gives the following:

$$\Pr [E_{\top}(P)[\chi_i] \oplus E_{\top}(P')[\chi_i] = 0] = (1 - p) \cdot 1/2 + p = 1/2 + p/2 \quad (5)$$

$$\mathcal{I}(E_{\top}(P)[\chi_i] \oplus E_{\top}(P')[\chi_i]) = p \quad (6)$$

- We also have $\mathcal{I}(E_{\top}(P)[\chi_i] \oplus C[\chi_o]) = \mathcal{I}(E_{\top}(P')[\chi_i] \oplus C'[\chi_o]) = \varepsilon$ from the linear approximations. Combining with (8), we get $\mathcal{I}(C[\chi_o] \oplus C'[\chi_o]) = p\varepsilon^2$.

A more rigorous analysis has been recently provided by Blondeau, Leander and Nyberg [13], but since we use experimental values for the complexity, the 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 $\hat{\varepsilon}[s]$ in every subset s . Finally, for each key guess k , we compute the expected imbalance $\varepsilon_k[s]$ for each subset s , and then we evaluate the distance between the observed and expected imbalances as $L(k) = \sum_s (\hat{\varepsilon}[s] - \varepsilon_k[s])^2$ (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 \hat{p} , and a fraction μ_{lin} of the output pairs with an improved linear imbalance $\hat{\varepsilon}$, the data complexity of the attack is $\mathcal{O}(\mu_{\text{lin}}\mu_{\text{diff}}^2 T/2 \times 2/\hat{p}^2 \hat{\varepsilon}^4)$. This corresponds to a gain of $G_{\text{diff-2}}G_{\text{lin}}^2$.

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 $\hat{\varepsilon}[P \oplus P', P[p_{\text{lin}}], C[c_{\text{diff}}], C'[c_{\text{diff}}]]$.

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 $\varepsilon_{k_{\text{diff}}, k_{\text{lin}}}[\Delta, p, c, c'] = \varepsilon_0[\Delta, p \oplus k_{\text{lin}}, c \oplus k_{\text{diff}}, c' \oplus k_{\text{diff}}]$.

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 [19]. Indeed, the distance between the observed and expected imbalance can be written as:

$$\begin{aligned} L(k) &= \sum_s (\hat{\varepsilon}[s] - \varepsilon_k[s])^2 \\ &= \sum_s (\hat{\varepsilon}[s] - \varepsilon_0[s \oplus \phi(k)])^2, \quad \text{where } \phi(k_{\text{diff}}, k_{\text{lin}}) = (0, k_{\text{lin}}, k_{\text{diff}}, k_{\text{diff}}) \\ &= \sum_s \hat{\varepsilon}[s]^2 + \sum_s \varepsilon_0[s \oplus \phi(k)]^2 - 2 \sum_s \hat{\varepsilon}[s] \varepsilon_0[s \oplus \phi(k)], \end{aligned}$$

where only the last term depend on the key. Moreover, this term can be seen as the $\phi(k)$ -th component of the convolution $\varepsilon_0 * \hat{\varepsilon}$. 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 $\varepsilon_0[s]$ of the differential-linear distinguisher for the zero key, for every subset s .
2. Collect D plaintext-ciphertext pairs, and compute the observed imbalance $\hat{\varepsilon}[s]$ of each subset.
3. Compute the convolution $\varepsilon * \hat{\varepsilon}$, 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 $E = E_{\perp} \circ E_{\top} \circ E_{\top}$, so that the single active bit in the differential characteristic falls between E_{\top} and E_{\perp} , and the single active bit in the linear trail falls between E_{\top} 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 $\delta_o \xrightarrow{E_\top} \chi_i$ in E_\top (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 $X = E_\top(P)$, $Y = E_\top(X)$, $C = E_\perp(Y)$.
- We have $X \oplus X' = \delta_o$ with probability p . In this case, $\mathcal{I}(Y[\chi_i] \oplus Y'[\chi_i]) = b$
- Otherwise, we expect that $Y[\chi_i] \oplus Y'[\chi_i]$ is not biased. This gives the following:

$$\Pr [Y[\chi_i] \oplus Y'[\chi_i] = 0] = (1 - p) \cdot 1/2 + p \cdot (1/2 + b/2) = 1/2 + bp/2 \quad (7)$$

$$\mathcal{I}(Y[\chi_i] \oplus Y'[\chi_i]) = bp \quad (8)$$

- We also have $\mathcal{I}(Y[\chi_i] \oplus C[\chi_o]) = \mathcal{I}(Y'[\chi_i] \oplus C'[\chi_o]) = \varepsilon$ from the linear approximations. Combining with (8), we get $\mathcal{I}(C[\chi_o] \oplus C'[\chi_o]) = bp\varepsilon^2$.

In the E_\top 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 $E = E_\perp \circ E_\top \circ E_\top$ (varying the number of rounds), and all possible positions for the active bits i at the input of E_\top and the biased bit⁶ j at the output of E_\top . For each candidate, we evaluate experimentally the imbalance $\mathcal{I}(E_\top(x)[2^j] \oplus E_\top(x \oplus 2^i)[2^j])$, 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 $E = E_\perp \circ E_\top$, in order to combine a biased bit in E_\top with an approximation of E_\perp .

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 $\mathcal{I}(E_\top(x)[2^j] \oplus E_\top(x \oplus 2^i)[2^j])$ without looking for explicit trails in the middle. In particular, the transition between E_\top and E_\top is a transition between two differential characteristics, while the transition between E_\top and E_\perp 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_\top , and 1 round in E_\perp . The optimal differences and masks are:

⁶ We also consider pairs of adjacent bits, following the analysis of [15].

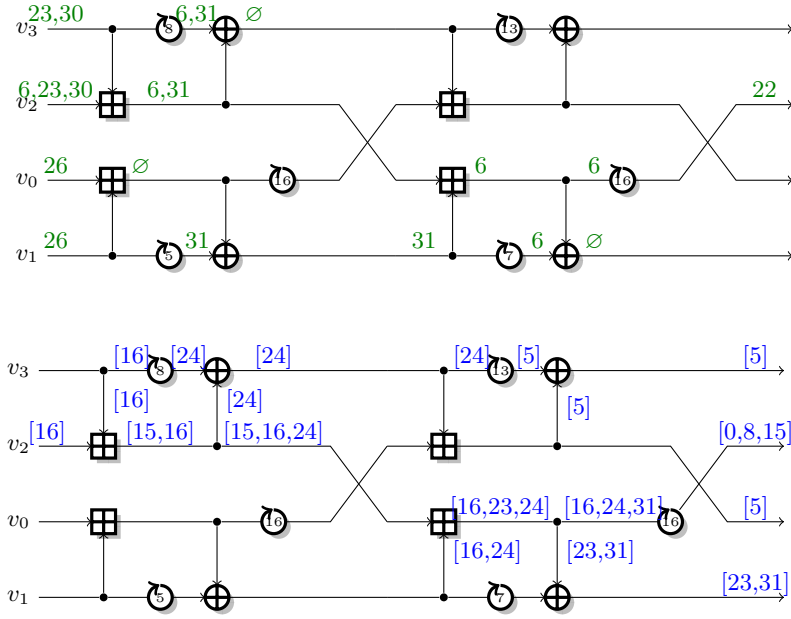


Fig. 7. 6-round attack: differential characteristic, and linear trail.

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

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

- 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, but we can improve it using the results of Sections 2 and 3.

Analysis of linear approximations with partitioning. To make the description easier, we first invert 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:

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

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 gain the following factor

$$G_{\text{lin}} = 2^{-8} \times 2^{6.84} / 2^{-2 \cdot 2.6} \approx 2^4 \quad (9)$$

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:

$$\begin{array}{ccc} v_2[24] & v_2[31] & v_0[27] \end{array}$$

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:

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

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 gain:

$$G_{\text{diff}} = 18/2^8 \times 2^4/2 \times 2^{-2}/2^{-5} = 9/2$$

$$G_{\text{diff-2}} = 18/2^8 \times 2^4/2 \times (2^{-2}/2^{-5})^2 = 36$$

This corresponds to the analysis of Section 3: we have a gain of $3/2$ for bits $v_2[23]$ and $v_0[27]$, and a gain of 2 for $v_2[31]$ in the simple linear case. In the differential-linear case, we have respectively gains of 3 and 2.

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

$$D/G_{\text{lin}}^2 G_{\text{diff-2}} \approx 2^{20.3}.$$

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 $2^2 D$ 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, Corrolary 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 $\hat{\varepsilon}[s]$. Each structure of 2^4 plaintexts provides 2^6 pairs, so that we need $2^2 D$ 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.⁷ 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 $p_{\top} = 2^{-17}$:
 $v_0[8, 18, 21, 30], v_1[8, 13, 21, 26, 30], v_2[3, 21, 26], v_3[21, 26, 27] \xrightarrow{E_{\top}} v_0[31]$
- Biased bit for E_{\perp} with imbalance $\varepsilon_{\perp} = 2^{-6.1}$:
 $v_0[31] \xrightarrow{E_{\perp}} v_2[20]$

⁷ Haswell microarchitecture running at 3.4 GHz

– 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 $D = 2/p_{\perp}^2 \varepsilon_{\perp}^2 \varepsilon_{\perp}^4 \approx 2^{77.6}$. 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 following control bits:

$$\begin{array}{lll} v_1[3] \oplus v_1[11] \oplus v_3[10] & v_1[3] \oplus v_1[11] \oplus v_3[11] & v_0[15] \oplus v_3[14] \\ v_0[15] \oplus v_3[15] & v_1[11] \oplus v_1[18] \oplus v_3[17] & v_1[11] \oplus v_1[18] \oplus v_3[18] \\ v_1[3] \oplus v_2[2] & v_1[3] \oplus v_2[3] & v_1[11] \oplus v_2[9] \\ v_1[11] \oplus v_2[10] & v_1[11] \oplus v_2[11] & v_1[18] \oplus v_2[17] \\ v_1[18] \oplus v_2[18] & v_1[2] \oplus v_1[3] & v_1[9] \oplus v_1[11] \\ v_1[10] \oplus v_1[11] & v_1[17] \oplus v_1[18] & v_0[14] \oplus v_0[15] \\ v_0[15] \oplus v_1[3] \oplus v_1[11] \oplus v_1[18] & & \end{array}$$

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:

$$G_{\text{lin}} = 2^{-19} \times 2^{14.38} / 2^{-2 \cdot 7.6} \approx 2^{10.5} \quad (10)$$

This example clearly shows the power of the partitioning technique: using a few key guesses, we 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:

$$\begin{array}{cccccc} v_0[9] & v_0[22] & v_0[31] & v_0[19] & & \\ v_0[14] & v_0[27] & v_2[22] & v_2[27] & v_2[4] & \end{array}$$

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:

$$\begin{array}{cccc} v_0[4] \oplus v_2[3] & v_2[22] \oplus v_3[21] & v_2[27] \oplus v_3[26] & v_2[27] \oplus v_3[27] \\ v_2[3] \oplus v_2[4] & v_2[21] \oplus v_2[22] & v_2[26] \oplus v_2[27] & v_0[9] \oplus v_1[8] \\ v_0[14] \oplus v_1[13] & v_0[27] \oplus v_1[26] & v_0[30] \oplus v_1[30] & v_0[8] \oplus v_0[9] \\ v_0[18] \oplus v_0[19] & v_0[21] \oplus v_0[22] & & \end{array}$$

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 gain:

$$G_{\text{diff-2}} = 17496/2^{23} \times 2^{10}/2 \times (2^{-11}/2^{-17})^2 = 4374 \approx 2^{12.1}$$

Finally, the improved attack requires a data complexity of:

$$D/G_{\text{lin}}^2 G_{\text{diff-2}} \approx 2^{44.5}.$$

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, Corrolary 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 $\hat{\epsilon}[s]$. 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, as demonstrated by our applications.

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 SPN ciphers.

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)
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 and Youssef [21], pp. 59–76
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 and Vanstone [32], pp. 2–21
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: FSE 2014 (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. Desmedt, Y. (ed.): Advances in Cryptology - CRYPTO ’94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings, Lecture Notes in Computer Science, vol. 839. Springer (1994)
18. Gilbert, H., Chassé, G.: A Statistical Attack of the FEAL-8 Cryptosystem. In: Menezes and Vanstone [32], pp. 22–33

19. 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)
20. Huang, T., Tjuawinata, I., Wu, H.: Differential-linear cryptanalysis of icepole. In: *FSE 2015* (2015)
21. Joux, A., Youssef, A.M. (eds.): *Selected Areas in Cryptography - SAC 2014 - 21st International Conference*, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers, Lecture Notes in Computer Science, vol. 8781. Springer (2014)
22. Lamberger, M., Mendel, F.: Higher-Order Differential Attack on Reduced SHA-256. *IACR Cryptology ePrint Archive*, Report 2011/37 (2011)
23. Langford, S.K., Hellman, M.E.: Differential-Linear Cryptanalysis. In: Desmedt [17], pp. 17–25
24. 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)
25. 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)
26. 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)
27. 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)
28. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseht, T. (ed.) *EUROCRYPT*. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993)
29. 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)
30. Mavromati, C.: Key-recovery attacks against the mac algorithm chaskey. In: *SAC 2015* (2015)
31. 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)
32. Menezes, A., Vanstone, S.A. (eds.): *Advances in Cryptology - CRYPTO '90*, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings, Lecture Notes in Computer Science, vol. 537. Springer (1991)
33. 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)
34. Mouha, N.: Chaskey. <http://mouha.be/chaskey/> (November 2014)
35. Mouha, N., Mennink, B., Herrewewe, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In: Joux and Youssef [21], pp. 306–323
36. 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)

37. Murphy, S.: Overestimates for the Gain of Multiple Linear Approximations in Symmetric Cryptology. *IEEE Transactions on Information Theory* 57(7), 4794–4797 (2011)
38. 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)
39. Ohta, K., Aoki, K.: Linear Cryptanalysis of the Fast Data Encipherment Algorithm. In: Desmedt [17], pp. 12–16
40. 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)
41. 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)
42. Tardy-Corffdir, 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)
43. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) *FSE. Lecture Notes in Computer Science*, vol. 1636, pp. 156–170. Springer (1999)
44. 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)
45. 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)
46. 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)
47. 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)