Improving the key recovery in Linear Cryptanalysis: An application to PRESENT

Antonio Florez Gutierrez

Improving the key recovery in Linear Cryptanalysis: An application to PRESENT

M2 Internship report by
Antonio Flórez Gutiérrez

Supervised by
María Naya-Plasencia (Inria Paris)

Abstract

Linear cryptanalysis is widely known as one of the fundamental tools for the crypanalysis of block ciphers. Over the decades following its first introduction by Matsui in [Ma94a], many different extensions and improvements have been proposed. One of them is [CSQ07], where Collard et al. use the Fast Fourier Transform (FFT) to accelerate the parity computations which are required to perform a linear key recovery attack. Modified versions of this technique have been introduced in order to adapt it to the requirements of several dedicated linear attacks. This work provides a model which extends and improves these different contributions and allows for a general expression of the time and memory complexities that are achieved. The potential of this general approach will then be illustrated with new linear attacks on reduced-round PRESENT, which is a very popular and widely studied lightweight cryptography standard. In particular, we show an attack on 26 or 27-round PRESENT-80 which has better time and data complexity than any previously known attacks, as well as the first attack on 28-round PRESENT-128.

Contents

1 Introduction

Background and motivation. The constant cryptanalysis effort on symmetric primitives is one of the fundamental arguments which support the security of secret key cryptography: an algorithm is considered secure if a long-enough analysis by the community has not led to the discovery of any significant flaws. For this reason, a toolbox of known cryptanalysis techniques must exist, as this allows the security of new cryptographic candidates to be tested against previous attacks. Furthermore, a good understanding of the nature of the flaws in the design of broken primitives can help designers create new ciphers whose resistance to these attacks is optimal.

The effect that the introduction of quantum computers might have on this system is unknown: there are many questions about quantum symmetric cryptanalysis which either remain unresolved or have not even been studied in detail. It is known that in order to keep the same level of security in a post-quantum setting, the length of the private key must be doubled to deal with the speedup of the brute-force attack due to Grover's algorithm. However, little is known about non-generic attacks in a quantum setting. One potential avenue for development in this area is studying how quantum computing can improve the performance of classical attacks, such as differential or linear cryptanalysis. This was already studied in [KLLN15], though several open problems are still unresolved.

Research problem. Linear cryptanalysis was first introduced by Matsui in 1994 ([Ma94a]), and soon joined differential cryptanalysis as one of the two essential tools in the cryptanalysis of symmetric primitives. In 2008, Collard et al. ([CSQ07]) introduced a new algorithmic technique which uses the Fast Fourier Transform to greatly reduce the time complexity of some linear attacks based on Matsui's Algorithm 2. The aim of this internship, which took place in the context of the ERC project QUASYModo at Inria Paris, was to attempt to answer two questions. In the first place, we wanted to know whether the approach of Collard et al. could be improved and/or generalized in the classical setting. The second question asks whether these FFT-based linear attacks could have improved performance in the quantum setting, and, if so, to which extent.

Contribution. Due to the limited duration of the internship and the amount of findings about the classical setting, almost all of the duration was spent researching the first question. During my internship the approach of [CSQ07] could be transformed into a more general version for multiple rounds of key recovery. Despite several publications ([NWW11],[ZZ15],[BTV18]) mentioning extended versions of the algorithm, it seems that a generalised algorithm has never been fully described in the literature. I have also started to work on an analysis of how the structure of the individual cipher might be exploited (such as the key dependencies induced by the key schedule, for example). These theoretical findings led to some new attacks on the popular ISO-standard lightweight construction PRESENT which are only possible due to the new algorithm. These include the first cryptanalysis of the 28-round, 128-bit key variant of the cipher.

Future directions in research. In the short term, we would like to complete some parts of our generalised algorithm: in particular, we feel that there is a lot of room for improvement in our analysis of the use of dependencies between the key bits. It is also possible that we attempt to improve our time complexity estimates for the attacks on PRESENT by using Bogdanov et al.'s multivariate profiling technique ([BTV18]), which would lighten the theoretical assumptions.

Another obvious avenue for continuing the research is attempting to answer question 2. We are also interested on the possible quantum speedup of Fast Correlation Attacks on stream ciphers, as they share similarities with linear cryptanalysis of block ciphers and can also be accelerated using the FFT.

Structure of this document. The present report is organised as follows: Section 2 offers a condensed overview of the development of the theory of linear cryptanalysis, with an emphasis on some of the prerequisites for our attacks on PRESENT. This section is complemented by appendix B, which contains further detail on some topics, as well as the proofs of the results.

Section 3 deals with the FFT version of Matsui's Algortithm 2. Subsection 3.1 describes the original

improved attack that was described in [CSQ07]. Subsection 3.2 is our first contribution, and provides a description of a generalised version of the algorithm which can be applied to the key recovery over multiple rounds. Subsection 3.3 discusses the possibility of using the key schedule of the attacked cipher to reduce the time complexity of key-recovery linear attacks using the FFT.

Section 4 is devoted to PRESENT and our attacks on reduced-round variants. It includes a brief description of the cipher, a thorough rundown of the design of our two attacks (from the choice of the linear distinguisher to the key recovery algorithm) and a short discussion about the experimental verification of our time complexity claims.

Appendix A contains the notations that are used throughout the report, some of which are non-standard. It also includes some short introductions to a few mathematical concepts, such as the correlation of boolean functions and the Walsh-Hadamard Transform.

Observations and acknowledgements. This report is presented in English due to my limited command of the French language, and would like to apologise for any inconvenience to the reader.

I would wish to thank the *Fondation Mathématique Jacques Hadamard* and its staff, whose Sophie Germain scholarship allowed me to study the M2 Algèbre Appliquée at the UVSQ, as well as Ana José Reguera López and Olivier Piltant for helping me find and apply for this scholarship. I also want to acknowledge all the professors at the University of Valladolid and UVSQ who have allowed me to reach this point with their unconditional support. My utmost gratitude is with María Naya-Plasencia for her constant assistance while supervising this internship, and the whole SECRET team at Inria Paris for making me feel right at home.

1.1 Some generalities on block ciphers

Throughout the report, we will use some general concepts about symmetric cryptology.

Definition 1.1. *A* block cipher *of length n and key length κ is a map $E : \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^n$ for which $E_K = E(\cdot, K)$ is bijective for any fixed* secret key $K \in \mathbb{F}_2^\kappa$. *Given a* plaintext x *and a key K, we will say $y = E_K(x)$ is the associated* ciphertext. *For any fixed key K, the map E_K is called the* encryption map. *Meanwhile, given a fixed key K there exists an inverse map E_K^{-1}, which is called the* decryption map. *We also write $E^{-1}(y, K) = E_K^{-1}(y)$ as a slight abuse of notation.*

Definition 1.2. *A block cipher E is said to be a* key-alternating cipher *if there exists a decomposition*

$$E_K = G_r^K \circ F \circ G_{r-1}^K \circ \ldots \circ F \circ G_1^K \circ F \circ G_0^K \tag{1.1}$$

where F is a fixed bijective map called internal permutation *and $G_i^K(x) = x \oplus K_i$ is called the* round subkey addition. *The maps $F \circ G_i^K$ are* rounds. *There is usually another subkey addition at the end. Each* subkey $K_i \in \mathbb{F}_2^n, i = 0, ..., r$ *is obtained from the master key K by means of an extension map $\mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^{n(r+1)}$ which is called the* key schedule.
Given a plaintext x and a key K, the key schedule is used to compute all the subkeys, and then the round function is applied on the plaintext r times. The intermediate results are usually called states.

A *key recovery attack* is an algorithm which allows a potential adversary to obtain the secret key K (or part of it) from a selection of pairs $(x, y = E_K(x))$ (the *data*) with high probability. There are several kinds of attacks depending on how these pairs are generated: if any set of pairs can be used, the attack is said to be a *known-plaintext attack*. If the attack demands a set of pairs with a specific structure, then the attack is said to be a *chosen-plaintext attack* (if the adversary chooses the values of x) or even a *chosen-ciphertext attack* (if the adversary chooses the values of y).

A known plaintext key recovery attack using one known plaintext-ciphertext pair is always possible: it suffices to test all possible values of K until one is found for which $y = E_K(x)$. This is called the *generic attack*, *brute-force attack*, or simply *exhaustive search*. This attack has a time complexity equivalent to 2^κ encryptions. The aim of cryptanalysts is finding attacks with lower time complexities.

2 An introduction to linear cryptanalysis

This section consists of an overview of the fundamental aspects of linear cryptanalysis and has the purpose of aiding the understanding of the rest of this report. It begins with an introduction to linear cryptanalysis which follows the approach introduced by Matsui in [Ma94a], and then proceeds to more advanced techniques like linear hulls ([Ny95]) and the use of multiple linear approximations ([BCQ04]). Subsection 2.2.1 contains an introduction to correlation matrices and how to use them to estimate the linear potential of a linear hull as shown in [AES02]. Subsection 2.4 describes a statistical model ([Se08],[BN15]) which will allow us to estimate the time complexities of our attacks. These topics are developed in more detail in appendix B, which also includes proofs.

2.1 Matsui's linear cryptanalysis

Matsui and Yamaguchi introduced an early version of linear cryptanalysis in [MaY93] for building an attack on the block cipher FEAL. The two fundamental algorithms used in linear cryptanalysis, Algorithms 1 and 2, were described in [Ma94a]. Algorithm 2 permitted the first experimental attack on the widely-used Data Encryption Standard (DES) in [Ma94b]. This subsection contains the essential results of these papers, as they constitute the foundation of linear cryptanalysis.

Matsui's linear cryptanalysis is a known plaintext attack which exploits a statistical correlation between some bits of the plaintext, the ciphertext and the secret key of a block cipher. The first step of any linear attack is finding linear approximations of (part of) the cipher.

Definition 2.1. *A* linear approximation *of the block cipher E is an expression of the form*

$$\nu : \alpha \cdot x \oplus \beta \cdot y \oplus \gamma(K) \tag{2.1}$$

The vectors $\alpha, \beta \in \mathbb{F}_2^n$ are called input *and* output masks*, while the map $\gamma : \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2$ is often called the* key mask*. Matsui's attacks use a biased linear approximation, which means that, if the approximation is regarded as a random variable over the key and the plaintext, then the probability that it is equal to 0 is significantly different from $1/2$.*

$$\varepsilon = Pr_{x,K}(\alpha \cdot x \oplus \beta \cdot E_K(x) \oplus \gamma(K) = 0) - 1/2 \neq 0 \tag{2.2}$$

The quantity ε is called bias *or* imbalance*. A related magnitude is the* correlation

$$c = Pr_{x,K}(\alpha \cdot x \oplus \beta \cdot E_K(x) \oplus \gamma(K) = 0) - Pr_{x,K}(\alpha \cdot x \oplus \beta \cdot E_K(x) \oplus \gamma(K) = 1) \neq 0 \tag{2.3}$$

These two values are linked by the relationship $c = 2\varepsilon$.

The attacker wants approximations with high bias/correlation (in absolute value), since they lead to a better data complexity. High-bias approximations can be found for each round of a key-alternating cipher by looking at the internal permutation F, which is usually a composition of other simpler maps. These one-round approximations can be combined to construct a composite approximation of the whole cipher whose correlation is given by the following result:

Corollary 2.2 (Piling-up lemma, lemma 3 in [Ma94a])**.** *Let us consider r linear approximations of the internal permutation F of a key-alternating cipher E,*

$$\alpha_i \cdot x \oplus \beta_i \cdot F(x),\ i = 0, \ldots, r-1 \tag{2.4}$$

with bias ε_i and correlation $c_i = 2\varepsilon_i$. We can deduce approximations of each round with the same bias:

$$\alpha_i \cdot x \oplus \beta_i \cdot F(x \oplus K_i) \oplus \alpha_i \cdot K_i$$

If we suppose that $\alpha_{i+1} = \beta_i$, then by addition, we obtain an approximation of E:

$$\alpha_0 \cdot x \oplus \beta_{r-1} \cdot E_K(x) \oplus \alpha_0 \cdot K_0 \oplus \ldots \oplus \alpha_{r-1} \cdot K_{r-1} \oplus \beta_{r-1} \cdot K_r \tag{2.5}$$

If we assume the statistical independence of the binary variables, then its bias and correlation are

$$\varepsilon = 2^{r-1} \prod_{i=0}^{r-1} \varepsilon_i, \quad c = \prod_{i=0}^{r-1} c_i \tag{2.6}$$

Matsui proposed two different key recovery attack algorithms which make use of a linear approximation. Algorithm 1 uses an approximation of the whole cipher to recover one bit of information about the key, while Algorithm 2 uses an approximation of $r-1$ rounds of the cipher to recover multiple bits.

2.1.1 Matsui's Algorithm 1

Suppose that the attacker has access to a collection $\mathcal{D} = \{(x, y = E_K(x))\} \subset \mathbb{F}_2^n \times \mathbb{F}_2^n$ of N known plaintexts which have been encrypted with a fixed secret key K, and knows a linear approximation $\alpha \cdot x \oplus \beta \cdot y \oplus \gamma(K)$ with bias ε. If N is large enough to detect the bias ε (we will detail how large later), then the following algorithm recovers the value of $\gamma(K)$:

Algorithm 1: Matsui's Algorithm 1

Input: A collection $\mathcal{D} = \{(x, y = E_K(x))\}$ of N plaintext-ciphertext pairs (possibly on-the-fly).
Output: A probable value for $\gamma(K)$.
$T \leftarrow 0$;
forall $(x, y) \in \mathcal{D}$ **do** `// Compute the counter` T
  if $\alpha \cdot x \oplus \beta \cdot y = 0$ **then** $T \leftarrow T + 1$;
end
switch (T, ε) **do** `// Compare` T `and` $N/2$
  case $T > N/2, \varepsilon > 0$ **do return** *0*;
  case $T > N/2, \varepsilon < 0$ **do return** *1*;
  case $T < N/2, \varepsilon > 0$ **do return** *1*;
  case $T < N/2, \varepsilon < 0$ **do return** *0*;
end

In other words, the attacker first computes the counter

$$T = \#\{(x, y) \in \mathcal{D} : \alpha \cdot x \oplus \beta \cdot y = 0\} \tag{2.7}$$

which is expected to be approximately $N/2 \pm N\varepsilon$, with the sign depending on the value of $\gamma(K)$. By comparing this number to $N/2$, this bit of information about the key can be obtained with high probability. The effectiveness of this algorithm depends on the following assumption:

Assumption 2.3. (Right-key equivalence hypothesis) *The key K has no influence on the value of the probability $p(K) = Pr_x\left(\alpha \cdot x \oplus \beta \cdot E_K(x) \oplus \gamma(K) = 0 \mid K\right)$, where $Pr_x\left(A \mid K\right)$ denotes the probability of the event A assuming that the plaintexts x are uniformly distributed and the key is fixed to K.*

Under the right-key equivalence hypothesis, we can estimate the success probability (that is, the probability that the guess for $\gamma(K)$ is correct) as a function of the number of known plaintexts N.

Proposition 2.4 (Lemma 2 in [Ma94a])**.** *The probability of success of Algorithm 1 is*

$$P_S \simeq \Phi\left(2\sqrt{N}|\varepsilon|\right) = \Phi\left(\sqrt{Nc^2}\right) = \int_{-\infty}^{\sqrt{Nc^2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \tag{2.8}$$

This means that around $N \sim 1/4\varepsilon^2 = 1/c^2$ known plaintexts are needed for the attack to succeed with reasonable probability. The attacker can recover more information about the key by using other approximations or by finding the rest of the key by exhaustive search.

2.1.2 Matsui's Algorithm 2

Matsui also proposed an improved attack which recovers a larger section of the key. The idea is to use a partial approximation of the cipher. If $E_K(x) = (F \circ E'_K)(x) \oplus K_r$, then we suppose that the attacker knows an approximation $\alpha \cdot x \oplus \beta \cdot \hat{y} \oplus \gamma(K)$ with bias ε of the first $r-1$ rounds.

We suppose that computing $\beta \cdot F^{-1}(y \oplus K_r)$ from y only requires using a few bits k of K_r, which can be extracted with the mask χ. For any vector $x \in \mathbb{F}_2^n$, $x|_\chi$ denotes the vector of length $HW(\chi)$ whose components are the coordinates of x which correspond to the non-zero entries of χ, so that $k = K_r|_\chi$. We can substitute the term associated to the decryption of the last round for a map $f : \mathbb{F}_2^{|k|} \longrightarrow \mathbb{F}_2$:

$$f\left(y|_\chi \oplus K_r|_\chi\right) = \beta \cdot F^{-1}(y \oplus K_r) \text{ for all } y \in \mathbb{F}_2^n, K_r \in \mathbb{F}_2^n \tag{2.9}$$

Given N plaintext-ciphertext pairs, the partial subkey k can be retrieved as follows:

Algorithm 2: Matsui's Algorithm 2

Input: A collection $\mathcal{D} = \{(x, y = E_K(x))\}$ of N plaintext-ciphertext pairs (possibly on-the-fly).
Output: A probable guess for k.

```
T ← 0;
forall (x, y) ∈ D do                                   // Compute the counters T_k
    for k ← 0 to 2^|k| − 1 do
        if α · x ⊕ f(y|_χ ⊕ k) = 0 then T_k ← T_k + 1;
    end
end
return argmax_k(|T_k − N/2|);                  // Find the T_k that's most different to N/2
```

The attacker *guesses* all the possible values of k and computes the counter

$$T_k = \#\left\{(x, y) \in \mathcal{D} : \alpha \cdot x \oplus f(y|_\chi \oplus k) = 0\right\} \tag{2.10}$$

and then keeps the guess whose counter is farthest from $N/2$. This is based on the assumption

Assumption 2.5. (Wrong-key randomisation hypothesis) *For all the wrong guesses of k, the correlation of the linear approximation is approximately zero. That is, if $\tilde{k} \neq k$ is a wrong guess,*

$$Pr_x\left(\alpha \cdot x \oplus f(y|_\chi \oplus \tilde{k}) = 0\right) \simeq 1/2 \tag{2.11}$$

We also assume that $\alpha \cdot x \oplus f(y|_\chi \oplus \tilde{k})$ are independent variables for all the wrong key guesses.

This assumption seems reasonable, since decrypting the last round of the cipher with the wrong key should have a similar effect to adding an additional round of encryption when it comes to the correlation of the approximation. This leads to the result

Corollary 2.6 (Lemma 4 in [Ma94a])**.** *The probability of success of Matsui's Algorithm 2 depends solely on the quantities $|k|$ and $4N\varepsilon^2 = Nc^2$.*

It takes $N2^{|k|}$ one-round decryptions and $2^{|k|}$ memory registers of up to $\log N$ bits to compute the counters T_k. The search phase of a full key-recovery attack is faster than in Algorithm 1, as we already know $|k|$ bits of the key K. The time complexity of Matsui's Algorithm 2 is $O\left(N2^{|k|}\right) + O\left(2^{\kappa-|k|}\right)$.

In [Ma94b], Matsui noted that since the only information about each (x, y) pair that is required is the values of $\alpha \cdot x$ and $y|_\chi$, it is possible to first count the number of occurrences of each $(\alpha \cdot x \parallel y|_\chi)$ in the data and then compute T_k using these. With this technique the attack takes N parity evaluations and $2^{2|k|}$ one-round decryptions, which reduces the complexity if $2^{|k|} < N$, which happens very often.

Algorithm 3: Matsui's Algorithm 2, modified

Input: A collection $\mathcal{D} = \{(x, y = E_K(x))\}$ of N plaintext-ciphertext pairs (possibly on-the-fly).
Output: A probable guess for k.
$\mathbf{a}^0 \leftarrow \underline{\mathbf{0}}$; $\mathbf{a}^1 \leftarrow \underline{\mathbf{0}}; \mathbf{T} \leftarrow \underline{\mathbf{0}}$;
forall $(x, y) \in \mathcal{D}$ **do** $a^{\alpha\cdot x}_{y|_\chi} \leftarrow a^{\alpha\cdot x}_{y|_\chi} + 1$; `// Extract the information from the data`
for $k \leftarrow 0$ **to** $2^{|k|} - 1$, $j \leftarrow 0$ **to** $2^{|k|} - 1$ **do** `// Compute the counters` T_k
  if $f(j \oplus k) = 0$ **then** $T_k \leftarrow T_k + a^0_j$ **else** $T_k \leftarrow T_k + a^1_j$;
end
return $argmax_k(|T_k - N/2|)$; `// Find the` T_k `that's most different to` $N/2$

Algorithm 2 can also be used with an approximation over even less rounds of the cipher by skipping several rounds at the beginning and/or the end. The limitation is that the number $|k|$ of involved subkey bits increases with the number of "outer" rounds.

When compared with Algorithm 1, Algorithm 2 has several advantages. Since it uses an approximation over a smaller number of rounds of the cipher, its correlation should be larger, and the data complexity is smaller than for Algorithm 1. Furthermore, recovering more bits of information about the key reduces the time cost of the final exhaustive search. Finally, we will be show in the next subsection that Algorithm 2 can still be used when the right-key equivalence hypothesis is not satisfied. For these reasons, nearly all linear attacks in the literature are variations of Algorithm 2.

2.2 Linear trails and linear hulls

The right-key equivalence hypothesis (assumption 2.3) is reasonable as long as all linear approximations which share the same input and output masks α, β but differ in the key mask γ have a much smaller correlation. This is not true in the case of many modern ciphers.

This possibility was first considered by Nyberg in [Ny95]. If the right-key equivalence hypothesis is not satisfied, then the value of $\alpha \cdot x \oplus \beta \cdot E_K(x)$ is influenced by several different $\gamma(K)$, and Algorithm 1 cannot distinguish between them. In the case of Algorithm 2, however, the probability of success can increase if there are multiple biased linear approximations sharing the same input and output masks.

This is substantiated by the following theorem, which applies to ciphers with a simplified key schedule.

Theorem 2.7 (Theorem 1 in [Ny95])**.** *Let $\nu : \alpha\cdot x\oplus\beta\cdot y$ be a linear approximation of a key-alternating block cipher E for which all the subkeys are independent from each other (so $\kappa = n(r+1)$). Then*

$$\begin{aligned}\frac{1}{2^{n(r+1)}} \sum_{K\in\mathbb{F}_2^\kappa} \left(Pr_x\left(\alpha\cdot x\oplus\beta\cdot E_K(x) = 0|K\right) - \frac{1}{2}\right)^2 \\ = \sum_{\gamma_0,\ldots,\gamma_r\in\mathbb{F}_2^n} \left(Pr_{x,K}\left(\alpha\cdot x\oplus\beta\cdot E_K(x)\oplus\gamma_0\cdot K_0\oplus\ldots\oplus\gamma_r\cdot K_r = 0\right) - \frac{1}{2}\right)^2\end{aligned} \tag{2.12}$$

Let us now consider what this means for the probability of success of Matsui's Algorithm 2. The value of $p(K) = Pr_x\left(\alpha\cdot x\oplus\beta\cdot E_K(x) = 0|K\right)$ varies over the keyspace, so the attack will succeed with a higher probability for some keys (which we'll call *weak keys*), and with a lower probability for others. We know that the probability of success for the key K depends on $4N(p(K)-1/2)^2$, and the previous theorem gives us the expected value of this magnitude over the keyspace. This result usually provides reasonable predictions even when the hypothesis about the key schedule is not satisfied.

Definition 2.8. *Let $\nu : \alpha\cdot x\oplus\beta\cdot y$ be a linear approximation of a key-alternating block cipher E. A* linear trail *is any linear approximation of the cipher of the form*

$$\nu_{\boldsymbol{\gamma}} : \alpha\cdot x\oplus\beta\cdot y\oplus\gamma_0\cdot K_0\oplus\ldots\oplus\gamma_r\cdot K_r, \quad \boldsymbol{\gamma} = (\gamma_0 = \alpha, \ldots, \gamma_r = \beta) \in (\mathbb{F}_2^n)^{(r+1)} \tag{2.13}$$

The set of all the linear trails of the approximation is called the approximate linear hull*, $ALH(\alpha, \beta)$. The probability of success of an Algorithm 2 attack using the approximation ν depends on the value*

$N \cdot ELP(\alpha, \beta)$, *where* $ELP(\alpha, \beta)$ *is the* estimated linear potential *of the linear hull*

$$ELP(\alpha, \beta) = \sum_{\substack{\boldsymbol{\gamma} \in \mathbb{F}_2^{n(r+1)} \\ \gamma_0 = \alpha, \gamma_r = \beta}} \left(Pr_{x,K_0,\ldots,K_r}(\nu_{\boldsymbol{\gamma}} = 0) - Pr_{x,K_0,\ldots,K_r}(\nu_{\boldsymbol{\gamma}} = 1) \right)^2 \tag{2.14}$$

2.2.1 Computing the ELP with orrelation matrices

The aim of this subsection is to illustrate an efficient way of estimating the ELP of linear approximations whose input and output masks have low Hamming weight by using correlation matrices. This is the technique that will be used to compute the capacity of our linear distinguishers for the PRESENT block cipher in section 4. We begin by defining the correlation matrix of a boolean function:

Definition 2.9. *Let* $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ *be a boolean function. Its* correlation matrix $C^{(f)} \in \mathbb{R}^{2^n \times 2^n}$ *is defined as the matrix whose entries are the correlations of all possible linear approximations of* f*:*

$$C_{ij}^{(f)} = \frac{1}{2^n} \left(\#\left\{ x \in \mathbb{F}_2^n : i \cdot x \oplus j \cdot f(x) = 0 \right\} - \#\left\{ x \in \mathbb{F}_2^n : i \cdot x \oplus j \cdot f(x) = 1 \right\} \right), \quad 0 \leq i, j < 2^n \tag{2.15}$$

Another common way of expressing the correlation of all the linear approximations of a boolean function is the linear approximation table *or LAT, whose entries are* $\#\left\{ x \in \mathbb{F}_2^n : i \cdot x \oplus j \cdot f(x) = 0 \right\} - 2^{n-1}$.

In many block ciphers the inner permutation is the composition of a linear map and the parallel application of several smaller non-linear maps called *S-boxes* on parts of the state. In this case the linear approximation tables or the correlation matrices of the S-boxes can be used to construct the correlation matrix of the whole round function using a variant of the piling-up lemma. Given an approximation of the round function, it decomposes into a sum of approximations of some S-boxes. These are often called *active S-boxes* of the approximation. This notion extends to linear trails, as all linear trails have a unique decomposition as approximations of each round.

The following result is deduced from the piling-up lemma and the definition of the ELP:

Theorem 2.10. *Let* E *be a key-alternating cipher of* r *rounds with internal permutation* F*, and let* $\hat{C}^{(F)}$ *be the element-wise square of the correlation matrix of the round function. Then the* ELP *of a linear approximation of* E *with input mask* α *and output mask* β *is*

$$ELP(\alpha, \beta) = \left(\left(\hat{C}^{(F)} \right)^r \right)_{\alpha\beta} \tag{2.16}$$

In most practical ciphers n is too large to construct the correlation matrix of the round function in full, but in many cases a good estimation of the ELP can still be obtained by considering a submatrix of $C^{(F)}$ containing a carefully picked selection of masks, such as those with a bound Hamming weight or those which lead to a limited amount of active S-boxes in the previous and the next rounds. This limits the amount of linear trails which are considered to those whose intermediate masks belong to the selected family. If all other linear trails have a very small correlation (in other words, if the submatrix of $C^{(F)}$ is chosen well enough), then the obtained estimation of the ELP can be very accurate.

2.3 Multiple linear cryptanalysis

Linear cryptanalysis can also be extended by using more than one linear approximation. This possibility was first discussed by Kaliski and Robshaw in [KR94], where the information obtained from several linear approximations using a single key mask γ but different input and output masks α_i, β_i was combined by the attacker to achieve a better success probability.

An approach allowing for the use of any set of linear approximations was introduced by Biryukov et al. in [BCQ04], and is commonly referred to as *multiple linear cryptanalysis*. However, the analysis of the probability of success relies on the assumption that all the approximations are statistically independent, and this is not always the case.

For this reason, Hermelin et al. proposed *multidimensional linear cryptanalysis* in [HCN08] and [HCN09]. Multidimensional linear cryptanalysis uses linear approximations whose input and output masks constitute a linear subspace of $\mathbb{F}_2^n \times \mathbb{F}_2^n$, which means that the estimation of the probability of success takes into account the joint distribution of all these approximations and doesn't require the assumption of statistical independence. Multidimensional attacks are covered in appendix B.

We will now describe a simple multiple version of Matsui's Algorithm 2. Let ν_i be M linear approximations of the $r-1$ first rounds of the cipher,

$$\nu_i : \alpha_i \cdot x \oplus \beta_i \cdot \hat{y}, \quad i = 1, \ldots, M \tag{2.17}$$

We suppose that $\beta_i \cdot F^{-1}(y \oplus K_r)$ can be replaced by $f_i(y|_\chi \oplus k)$ for each approximation as in the simple version of Algorithm 2. Then an attack using N known plaintext-ciphertext pairs (x, y) can be performed as follows:

Algorithm 4: Multiple version of Matsui's Algorithm 2

Input: A collection $\mathcal{D} = \{(x, y = E_K(x))\}$ of N plaintext-ciphertext pairs (possibly on-the-fly).
Output: A probable guess for k.
for $i \leftarrow 1$ **to** M **do** $\mathbf{q}^i \leftarrow \underline{\mathbf{0}}$;
forall $(x, y) \in \mathcal{D}$ **do** `// Compute the counters` q_k^i `for all keys and approximations`
  for $i \leftarrow 1$ **to** M, $k \leftarrow 0$ **to** $2^{|k|} - 1$ **do**
    if $\alpha_i \cdot x \oplus f_i(y|_\chi \oplus k) = 0$ **then** $q_k^i \leftarrow q_k^i + 1$ **else** $q_k^i \leftarrow q_k^i - 1$;
  end
end
for $k \leftarrow 0$ **to** $2^{|k|} - 1$ **do** $Q_k \leftarrow \frac{1}{N} \sum_{i=1}^{M} (q_k^i)^2$; `// Aggregate the information`
return $argmax_k(Q_k)$; `// Find the largest value of` Q_k

For each guess of k, the attacker computes the empirical correlation for all the approximations:

$$q_k^i = \#\{(x, y) \in \mathcal{D} : \alpha_i \cdot x \oplus f_i(y|_\chi \oplus k) = 0\} - \#\{(x, y) \in \mathcal{D} : \alpha_i \cdot x \oplus f_i(y|_\chi \oplus k) = 1\} \tag{2.18}$$

which are then aggregated into the multiple linear cryptanalysis statistic

$$Q_k = \frac{1}{N} \sum_{i=1}^{M} \left(q_k^i\right)^2 \tag{2.19}$$

The guess with the largest associated value of Q_k is probably correct. Under the assumption that all the linear approximations are statistically independent, the probability of success of the attack can be estimated using the notion of capacity of the set of approximations:

Proposition 2.11. *The probability of success of a multiple linear attack using M statistically independent linear approximations depends on the number of available plaintexts N, the number of approximations M and the* capacity *of the set of linear approximations. If $c_i(K)$ is the correlation of the i-th approximation for the key K, then the capacity for the key K and the overall capacity are defined as*

$$C(K) = \sum_{i=1}^{M} \left(c_i(K)\right)^2, \quad C = Exp_K\left(C(K)\right) = \sum_{i=1}^{M} \left(ELP(\alpha_i, \beta_i)\right)^2 \tag{2.20}$$

2.4 Estimating the probability of success

Another "branch" in the theory of linear cryptanalysis which has been developed in the last few decades deals with the analysis of the probabilistic behaviour of linear approximations and how it can be used to better estimate the probability of success of a linear attack.

In an attack based on Matsui's Algorithm 2, it is possible to keep more than one key candidate: for example, we can keep a fixed number of the highest ranked key candidates, or all candidates which

surpass a certain correlation threshold. This increases the probability of success of the attack. Selçuk introduced the notion of advantage ([Se08]) in order to measure the effectiveness of this type of attack:

Definition 2.12. *An attack that ranks the partial key guesses k according to a statistic X_k achieves an* advantage *of a bits if the right key ranks among the best $2^{|k|-a}$ key candidates. Given a desired advantage, the probability of success is the probability that the actual advantage surpasses the desired value. If an advantage of a bits is achieved, then the time complexity of the exhaustive search phase of the attack is $2^{\kappa-a}$ full encryptions.*

The probability of success given a target advantage can be computed using the following result:

Theorem 2.13 ([Se08])**.** *Supposing that the key-ranking statistic X_k has the cumulative distribution function F_R for the right key guess and F_W for any wrong guess, then the success probability of the associated statistical attack for a given desired advantage a is*

$$P_S = 1 - F_R\left(F_W^{-1}(1-2^{-a})\right) \tag{2.21}$$

If the right-key distribution can be approximated by a normal distribution $\mathcal{N}(\mu_R, \sigma_R^2)$, then

$$P_S \simeq \Phi\left(\frac{\mu_R - F_W^{-1}(1-2^{-a})}{\sigma_R}\right) \tag{2.22}$$

For multiple linear cryptanalysis, the distributions of the test statistics which are required for the previous result can be approximated by the widely-accepted theorem from [BN17]:

Theorem 2.14 (Theorem 6 in [BN17])**.** *In a multiple linear attack using M linear approximations and N available plaintexts, the right-key statistic Q_k approximately follows a normal distribution:*

$$\begin{array}{l} Q_k \sim \mathcal{N}(\mu_R, \sigma_R), \text{ where} \\ \quad \left\{ \begin{array}{rcccl} \mu_R & = & Exp_{\mathcal{D},K}(Q_k) & = & BM + NExp_K\left(C(K)\right) \\ \sigma_R^2 & = & Var_{\mathcal{D},K}(Q_k) & = & 2B^2M + 4BNExp_K\left(C(K)\right) + N^2Var_K\left(C(K)\right) \end{array} \right. \\ B = \left\{ \begin{array}{ll} 1 & \textit{if repeated plaintexts are allowed} \\ \frac{2^n-N}{2^n-1} & \textit{if the plaintexts are all different} \text{ (distinct known plaintext)} \end{array} \right. \end{array} \tag{2.23}$$

$Exp_K\left(C(K)\right)$ and $Var_K\left(C(K)\right)$ can be estimated using a subset $\mathcal{S}$ of significant linear trails,

$$Exp_K\left(C(K)\right) \simeq \sum_{i=1}^{M}\sum_{\gamma\in\mathcal{S}}\left(Pr_{x,\widehat{K}}(\nu_i \oplus \gamma\cdot\widehat{K}=0) - Pr_{x,\widehat{K}}(\nu_i \oplus \gamma\cdot\widehat{K}=1)\right)^2 + M2^{-n} \tag{2.24}$$

$$Var_K\left(C(K)\right) \simeq 2\sum_{i=1}^{M}\left(\sum_{\gamma\in\mathcal{S}}\left(Pr_{x,\widehat{K}}(\nu_i \oplus \gamma\cdot\widehat{K}=0) - Pr_{x,\widehat{K}}(\nu_i \oplus \gamma\cdot\widehat{K}=1)\right)^2 + 2^{-n}\right)^2 \tag{2.25}$$

Meanwhile, if the key guess $\tilde{k} \neq k$ is different from the right one, a multiple of the wrong key statistic follows a χ^2 distribution with M degrees of freedom:

$$\frac{1}{B+N2^{-n}}Q_{\tilde{k}} \sim \chi^2_M, \text{ so } \left\{ \begin{array}{rcccl} \mu_W & = & Exp_{\mathcal{D},K}(Q_{\tilde{k}}) & = & BM + NM2^{-n} \\ \sigma_W^2 & = & Var_{\mathcal{D},K}(Q_{\tilde{k}}) & = & 2M(B+N2^{-n})^2 \end{array} \right. \tag{2.26}$$

3 Matsui's Algorithm 2 using FFT

As was shown in the previous section, the naïve implementation of Matsui's Algorithm 2 requires $N2^{|k|}$ one-round decryptions in order to compute the test statistic for all possible guesses of the subkey. This number can be reduced to $O(N) + 2^{2|k|}$ if the attacker divides the statistical analysis into two phases and $2^{|k|} < N$. Since we are only interested in some bits of information about each plaintext-ciphertext pair, we begin by counting the number of appearances of each possibility for these relevant bits. Then the calculations are only performed once for each one of these possibilities, and then the result is multiplied by the number of appearances that was stored in the first table. This is a recurring strategy in statistical attacks, which can often be divided into the following phases:

1. **Distillation phase:** The relevant information for the attack is extracted from the data by partitioning the space of plaintexts and ciphertexts and counting the number of occurrences of each category. This can be done on-the-fly, which means that each plaintext-ciphertext pair can be processed separately as it is received by the attacker without storing the whole collection.

2. **Analysis phase:** The information that was extracted during the analysis phase is manipulated in order to obtain one or more candidates for a part of the key K.

3. **Search phase:** The rest of the key is found by testing all the possibilities for the remainder of the key exhaustively until the correct key is found.

Collard, Standaert and Quisquater showed in [CSQ07] that the Generalised Fast Fourier Transform (and, in particular, the Fast Walsh-Hadamard Transform) can speed up the time complexity of the analysis phase of the modified Algorithm 2 from $O\left(2^{2|k|}\right)$ to $O\left(|k|2^{|k|}\right)$ in last-round attacks (the original paper also briefly discusses first and last-round attacks, as well as attacks on ciphers in which the round subkey is added modulo 2^n instead of being XORed). During this internship we described an extended version of the algorithm which can be used for any number of rounds at the beginning and/or the end of the cipher, and is inspired by several applications of the technique in subsequent papers ([NWW11],[ZZ15],[BTV18]). This section contains both the original approach of [CSQ07] as well as our new generalised algorithm, and provides more accurate expressions of the time complexity. We also discuss how the key schedule might be exploited in generalised FFT-accelerated linear attacks.

3.1 The original algorithm

In [CSQ07] it was shown that the parity calculations for Matsui's Algorithm 2 can be performed more efficiently by using the FFT. This improved algorithm can be used when the cipher is of the form $E_K(x) = (F \circ E'_K)(x) \oplus K_r$ and the attacker uses one approximation of the form $\alpha \cdot x \oplus \beta \cdot \hat{y}$. As usual, we will suppose that the last round of the cipher can be truncated so that the partial decryption which obtains $\beta \cdot F^{-1}(y \oplus K_r)$ from y only requires using a few bits of K_r.

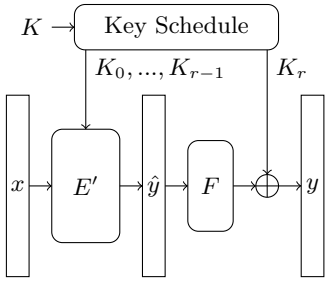

Figure 1: Attack on last round of a cipher.

The partial subkey guess will be denoted by k, and there are $2^{|k|}$ possibilities for this segment of the key. Let χ be the mask which extracts these relevant bits, so that $k = K_r|_\chi$. Let $f(y|_\chi \oplus k) = \beta \cdot F^{-1}(y \oplus K_r)$ denote the term of the approximation associated to the partial last round decryption. The attacker wants to compute the vector of experimental correlations whose entries are

$$q_k = \#\{(x,y) \in \mathcal{D} : \alpha \cdot x \oplus f(y|_\chi \oplus k) = 0\} - \#\{(x,y) \in \mathcal{D} : \alpha \cdot x \oplus f(y|_\chi \oplus k) = 1\} \tag{3.1}$$

with the aim of extracting the key candidate(s) with the largest value(s) of $|q_k|$. The experimental correlation can be rewritten as a sum in the following manner:

$$q_k = \sum_{(x,y)\in\mathcal{D}} (-1)^{\alpha\cdot x \oplus f(y|_\chi \oplus k)} = \sum_{j=0}^{2^{|k|}-1} (-1)^{f(j\oplus k)} \sum_{\substack{(x,y)\in\mathcal{D} \\ y|_\chi = j}} (-1)^{\alpha\cdot x}, \quad 0 \le k \le 2^{|k|}-1 \tag{3.2}$$

so that j represents the relevant $|k|$ bits of the ciphertext. This suggests that the attack should begin by computing the integer vector $\mathbf{a} \in \mathbb{Z}^{2^{|k|}}$ with coordinates

$$a_j = \sum_{\substack{(x,y)\in\mathcal{D} \\ y|_\chi = j}} (-1)^{\alpha\cdot x}, \quad 0 \le j \le 2^{|k|}-1 \tag{3.3}$$

This constitutes the distillation phase of the algorithm of [CSQ07]. We can also define the matrix $C \in \mathbb{Z}^{2^{|k|}\times 2^{|k|}}$ with entries

$$c_{jk} = (-1)^{f(j\oplus k)}, 0 \le j,k \le 2^{|k|}-1 \tag{3.4}$$

The vector $\mathbf{q} = (q_0, \ldots, q_{2^{|k|}-1})$ can thus be calculated as the matrix-vector product

$$\mathbf{q}^T = \mathbf{a}^T C \tag{3.5}$$

However, the time complexity of constructing C and computing the matrix-vector product is still $O\left(2^{2|k|}\right)$. The product can be computed in a much more efficient manner by making use of the following result on spectral analysis:

Proposition 3.1. *Let $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ be a boolean function. We consider a matrix of 1s and -1s $C \in \mathbb{Z}^{2^m\times 2^m}$ whose entries are of the form*

$$c_{ij} = (-1)^{f(i\oplus j)}, \quad 0 \le i,j \le 2^m - 1 \tag{3.6}$$

This matrix diagonalizes as

$$2^m C = H_{2^m} \Delta H_{2^m} \tag{3.7}$$

where H_{2^m} is the Walsh-Hadamard matrix of size 2^m whose entries are $h_{ij} = (-1)^{i\cdot j}$, and $\Delta = \mathrm{diag}(\boldsymbol{\lambda})$, $\boldsymbol{\lambda} \in \mathbb{Z}^{2^m}$ is a diagonal matrix.
The eigenvalue vector $\boldsymbol{\lambda}$ is the matrix-vector product $H_{2^m} C_{\cdot 1}$, where $C_{\cdot 1}$ denotes the first column of C.

Proof. We will show that the columns of the Walsh-Hadamard matrix (often called Walsh functions) are eigenvectors of C. In the process we will also obtain an expression for the associated eigenvalues. Let $h_{\cdot j}$ denote the j-th column of H_{2^l}. Its coordinates are therefore $(h_{\cdot j})_i = h_{ij} = (-1)^{i\cdot j}$. The i-th component of the matrix-vector product $Ch_{\cdot j}$ is equal to:

$$\sum_{k=0}^{2^m-1} c_{ik} h_{kj} = \sum_{k=0}^{2^m-1} (-1)^{f(i\oplus k)} (-1)^{k\cdot j} = \sum_{k=0}^{2^m-1} (-1)^{f(i\oplus k)\oplus k\cdot j}$$

Since the sum is taken over all the possible values of k, we can change the indices $k \oplus i$ for k to obtain

$$\sum_{k=0}^{2^m-1} (-1)^{f(k)\oplus(k\oplus i)\cdot j} = \sum_{k=0}^{2^m-1} (-1)^{f(k)\oplus k\cdot j\oplus i\cdot j} = \left(\sum_{k=0}^{2^m-1} (-1)^{f(k)\oplus k\cdot j}\right)(-1)^{i\cdot j},$$

which is a multiple of the i-th component of $h_{\cdot j}$. The associated eigenvalue is

$$\lambda_j = \sum_{k=0}^{2^m-1} (-1)^{k\cdot j}(-1)^{f(k)},$$

which is indeed the j-th component of the product of H_{2^m} and the first column of C. □

The matrix-vector product $\mathbf{a}^T C$ can then be further decomposed into:

$$2^{|k|}\mathbf{q}^T = \mathbf{a}^T H_{2^{|k|}} \mathrm{diag}(H_{2^{|k|}} C_{\cdot 1}) H_{2^{|k|}} \tag{3.8}$$

The cost of computing $\mathbf{q}$ is now three products of the form $H_{2^{|k|}}\mathbf{v}$, which can in turn be evaluated efficiently with the Fast Walsh-Hadamard transform (see appendix A) with $|k|2^{|k|}$ additions/substractions. The decomposition of C justifies the following algorithm:

1. **Distillation phase:** Construct a list of zeros $\mathbf{a}$ of length $2^{|k|}$. For each plaintext-ciphertext pair (x, y), calculate $\alpha \cdot x$. If it is equal to zero (resp. one), then increment (resp. decrement) the entry of $\mathbf{a}$ corresponding to the $|k|$ relevant bits of $y|_\chi$ by one.

   The time complexity of this phase is $\rho_D N$,where ρ_D is the cost (in binary operations) of evaluating $\alpha \cdot x$ and $y|_\chi$ and incrementing one of the counters.

2. **Analysis phase:** Compute the vector $\mathbf{q}$ of experimental correlations:

   (a) Compute the first column of C, and apply the Fast Walsh-Hadamard Transform to obtain $\boldsymbol{\lambda}$, the vector containing the eigenvalues of C.

   (b) Apply the Fast Walsh-Hadamard transform to $\mathbf{a}$.

   (c) Multiply the previous vector by $\boldsymbol{\lambda}$ elementwise.

   (d) Apply the Fast Walsh-Hadamard transform to the resulting vector to find $2^{|k|}\mathbf{q}$.

   Select the subkey guesses k with the largest values for $|q_k|$.

   The time complexity of this phase is $\rho_f 2^{|k|} + 3\rho_A|k|2^{|k|} + \rho_M 2^{|k|} + \rho_C 2^{|k|}$. The constant ρ_f denotes the cost in binary operations of evaluating $f(j)$, and ρ_A, ρ_M, ρ_C denote the cost of adding, multiplying and comparing two integers whose length is bound by n.

3. **Search phase:** For each of the remaining candidate guesses, try all the possibilities for the remainder of the key until either the complete key K is found or all the candidates have been tested without success.

   The time complexity of this phase is $\rho_E 2^{\kappa-a}$, where ρ_E is the cost of one encryption with E and a is the achieved advantage of the attack.

Proposition 3.2. *The previous algorithm for a key recovery linear attack using a linear approximation between the first and penultimate rounds of a block cipher has a time complexity*

$$\underbrace{\rho_D N}_{\substack{distillation\\phase}} + \underbrace{3\rho_A|k|2^{|k|} + (\rho_f + \rho_M + \rho_C)2^{|k|}}_{analysis\ phase} + \underbrace{\rho_E 2^{\kappa-a}}_{\substack{search\\phase}} \tag{3.9}$$

The memory requirement is $2 \cdot 2^{|k|}n$ *bits.*

Algorithm 5: The algorithm of [CSQ07] (without the final phase)

Input: A collection $\mathcal{D} = \{(x, y = E_K(x))\}$ of N plaintext-ciphertext pairs (possibly on-the-fly).
Output: The experimental correlations q_k (multiplied by a constant factor).

```
// DISTILLATION PHASE
```
$\mathbf{a} \leftarrow \mathbf{0}$;
forall $(x, y) \in \mathcal{D}$ **do**
  if $\alpha \cdot x = 0$ **then** $a_{y|_\chi} \leftarrow a_{y|_\chi} + 1$ **else** $a_{y|_\chi} \leftarrow a_{y|_\chi} - 1$;
end
```
// ANALYSIS PHASE
```
for $j \leftarrow 0$ **to** $2^{|k|} - 1$ **do** $\lambda_j \leftarrow f(j)$; `// Compute the first column of` C
$\boldsymbol{\lambda} \leftarrow \mathrm{FWHT}(\boldsymbol{\lambda})$; `// Compute the eigenvalues of` C
$\mathbf{a} \leftarrow \mathrm{FWHT}(\mathbf{a})$; `// Apply the FWHT to a`
for $j \leftarrow 0$ **to** $2^{|k|} - 1$ **do** $a_j \leftarrow a_j \cdot \lambda_j$; `// Multiply a by` $\boldsymbol{\lambda}$ `elementwise`
$\mathbf{q} \leftarrow \mathrm{FWHT}(\mathbf{a})$; `// Apply the FWHT to a`
return $\boldsymbol{q}$;

3.2 The extended algorithm

After the publication of [CSQ07], several linear attacks have made use of this FFT technique. Although it was originally only described for the case in which the adversary only wants to guess bits of the last round subkey, it is possible to apply the same approach in broader scenarios, as showcased in [CSQ07], [NWW11], [ZZ15], and [BTV18]. The main original result of this internship is the description of an algorithm which allows for an efficient partial key recovery on an arbitrary number of rounds at both the beginning and the end of the cipher, as well as a generic analysis of the time complexity.

Consider a block cipher E of block size n and key size κ which can be decomposed as in figure 2. The outer ciphers E_1 and E_2 represent the first and last few rounds of E. These ciphers take some *inner keys* $K_1 \in \mathbb{F}_2^{\kappa_1}$ and $K_2 \in \mathbb{F}_2^{\kappa_2}$ as input. In a key-alternating cipher, these inner keys correspond to the round subkeys involved in the key recovery attack, excluding the first and the last. The first and last round subkeys correspond to the *outer keys* K_0 and K_3, and will be treated separately.

We will suppose that the inner cipher E_M has a linear linear approximation of the form

$$\nu : \ \alpha \cdot \hat{x} \oplus \beta \cdot \hat{y} \tag{3.10}$$

As with the previous algorithms, we assume that the values of $\alpha \cdot E_1(x \oplus K_0, K_1)$ (resp. $\beta \cdot E_2^{-1}(y \oplus K_3, K_2)$) can be obtained from a part of x (resp. y) by guessing some bits of the keys K_0 and K_1 (resp. K_3 and K_2). We will denote the necessary part of the plaintext by i (resp. ciphertext, j), while the guessed parts of the subkeys will be denoted by k_0, k_1 (resp. k_3, k_2). We can consider masks

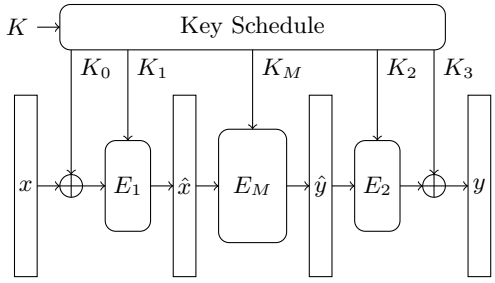

Figure 2: The general description of the cipher.

$\chi_0, \chi_1, \chi_2, \chi_3$, so that

$$i = x|_{\chi_0}, \qquad k_0 = K_0|_{\chi_0}, \qquad k_1 = K_1|_{\chi_1}, \qquad k_2 = K_2|_{\chi_2}, \qquad y = y|_{\chi_3}, \qquad k_3 = K_3|_{\chi_3} \tag{3.11}$$

Let $f_1 : \mathbb{F}_2^{2^{|k_0|}} \times \mathbb{F}_2^{2^{|k_1|}} \longrightarrow \mathbb{F}_2$ and $f_2 : \mathbb{F}_2^{2^{|k_3|}} \times \mathbb{F}_2^{2^{|k_2|}} \longrightarrow \mathbb{F}_2$ denote the maps verifying

$$f_1\left(x|_{\chi_0} \oplus K_0|_{\chi_0}, K_1|_{\chi_1}\right) = \alpha \cdot E_1\left(x \oplus K_0, K_1\right) \text{ for all possible } x, K_0, K_1 \tag{3.12}$$

$$f_2\left(y|_{\chi_3} \oplus K_3|_{\chi_3}, K_2|_{\chi_2}\right) = \beta \cdot E_2^{-1}\left(y \oplus K_3, K_2\right) \text{ for all possible } y, K_3, K_2 \tag{3.13}$$

The attacker is given a set $\mathcal{D}$ of N plaintext-ciphertext pairs $(x, y = E_K(x))$ which have been encrypted with a fixed key K. In order to perform a linear key recovery attack, the adversary needs to compute the experimental correlation for each possible guess of the subkeys:

$$\begin{aligned} q(k_0, k_1, k_2, k_3) = &\#\left\{(x, y) \in \mathcal{D} : f_1(i \oplus k_0, k_1) \oplus f_2(j \oplus k_3, k_2) = 0\right\} \\ &- \#\left\{(x, y) \in \mathcal{D} : f_1(i \oplus k_0, k_1) \oplus f_2(j \oplus k_3, k_2) = 1\right\}, \\ &(k_0, k_1, k_2, k_3) \in \mathbb{F}_2^{|k_0|} \times \mathbb{F}_2^{|k_1|} \times \mathbb{F}_2^{|k_2|} \times \mathbb{F}_2^{|k_3|} \end{aligned} \tag{3.14}$$

The attack begins with a modified distillation phase which is inspired by the work of Nguyen et al. in [NWW11]. A matrix $A \in \mathbb{Z}^{2^{|k_0|} \times 2^{|k_3|}}$, whose entries a_{ij} are the number of plaintext-ciphertext pairs for which the relevant bits are i and j, respectively, is constructed.

$$a_{ij} = \#\left\{(x, y) \in \mathcal{D} : x|_{\chi_0} = i, y|_{\chi_3} = j\right\}, \quad 0 \le i \le 2^{|k_0|} - 1, \quad 0 \le j \le 2^{|k_3|} - 1 \tag{3.15}$$

We can thus rewrite the experimental correlation for any key guess as the following sum:

$$q(k_0, k_1, k_2, k_3) = \sum_{i=0}^{2^{|k_0|}-1} \sum_{j=0}^{2^{|k_3|}-1} a_{ij} (-1)^{f_1(i \oplus k_0, k_1)} (-1)^{f_2(j \oplus k_3, k_2)} \tag{3.16}$$

Let us now consider that the values of $0 \le k_1 \le 2^{|k_1|} - 1$ and $0 \le k_2 \le 2^{|k_2|} - 1$ are fixed. All the associated experimental correlations (which now only depend on the choice of k_0 and k_3) form a matrix $Q^{k_1,k_2} \in \mathbb{Z}^{2^{|k_0|} \times 2^{|k_3|}}$ with entries

$$q^{k_1,k_2}_{k_0,k_3} = q(k_P, k_1, k_2, k_C), \quad 0 \le k_0 \le 2^{|k_0|} - 1, \quad 0 \le k_3 \le 2^{|k_3|} - 1 \tag{3.17}$$

We can see that $Q^{k_1,k_2} = B^{k_1} A C^{k_2}$, where $B^{k_1} \in \mathbb{Z}^{2^{|k_0|} \times 2^{|k_0|}}$ and $C^{k_2} \in \mathbb{Z}^{2^{|k_3|} \times 2^{|k_3|}}$, and the elements of these matrices are defined as

$$b^{k_1}_{k_0,i} = (-1)^{f_1(i \oplus k_0, k_1)}, \quad 0 \le k_0, i \le 2^{|k_0|} - 1 \tag{3.18}$$

$$c^{k_2}_{j,k_3} = (-1)^{f_2(j \oplus k_3, k_2)}, \quad 0 \le j, k_3 \le 2^{|k_3|} - 1 \tag{3.19}$$

All the matrices B^{k_1}, C^{k_2} adhere to the structure described in Proposition 3.1 and thus the product of a vector by one of these matrices can be computed by using tree Fast Walsh-Hadamard Transforms (FWHT). More precisely, these matrices decompose as

$$2^{|k_0|} B^{k_1} = H_{2^{|k_0|}} \mathrm{diag}\left(\boldsymbol{\lambda}_1^{k_1}\right) H_{2^{|k_0|}}, \text{ where } \boldsymbol{\lambda}_1^{k_1} = H_{2^{|k_0|}} B^{k_1}_{\cdot 1} \tag{3.20}$$

$$2^{|k_3|} C^{k_2} = H_{2^{|k_3|}} \mathrm{diag}\left(\boldsymbol{\lambda}_2^{k_2}\right) H_{2^{|k_3|}}, \text{ where } \boldsymbol{\lambda}_2^{k_2} = H_{2^{|k_3|}} C^{k_2}_{\cdot 1} \tag{3.21}$$

The matrices $Q^{k_1 k_2}$ can therefore be calculated as

$$2^{|k_0|+|k_3|} Q^{k_1 k_2} = H_{2^{|k_0|}} \mathrm{diag}\left(H_{2^{|k_0|}} B^{k_1}_{\cdot 1}\right) H_{2^{|k_0|}} A H_{2^{|k_3|}} \mathrm{diag}\left(H_{2^{|k_3|}} C^{k_2}_{\cdot 1}\right) H_{2^{|k_3|}} \tag{3.22}$$

The end result is that the attack can be performed efficiently as follows:

1. **Distillation phase:** Construct the matrix A by looking at each plaintext-ciphertext pair (x, y), finding the associated values of i and j and incrementing the corresponding a_{ij} by one.

   The time complexity of the distillation phase is $\rho_D N$ binary operations, where ρ_D is the cost of checking one pair. The distilled data needs $2^{|k_0|+|k_3|}$ memory registers of up to n bits.

2. **Analysis phase:** Compute all the experimental correlations $q(k_0, k_1, k_2, k_3)$:

   (a) Apply the FWHT on all rows and columns of A to obtain a matrix $\widehat{A}$.
   The cost of applying the transform on all the rows is $\rho_A \cdot 2^{|k_0|} \cdot |k_3| 2^{|k_3|}$ and the cost of applying it on all the columns is $\rho_A \cdot 2^{|k_3|} \cdot |k_0| 2^{|k_0|}$, so the total cost is $\rho_A \left(|k_0|+|k_3|\right) 2^{|k_0|+|k_3|}$. No additional memory is required.

   (b) Construct all the eigenvalue vectors $\boldsymbol{\lambda}_1^{k_1}$ and $\boldsymbol{\lambda}_2^{k_2}$. This is done by calculating the first column of B^{k_1} or C^{k_2} and then applying the FWHT on this vector.
   This step can be precomputed before the distillation phase, and requires $\rho_{f_1} 2^{|k_0|+|k_1|} + \rho_A |k_0| 2^{|k_0|+|k_1|}$ operations to compute all the vectors $\boldsymbol{\lambda}_1^{k_1}$ (where ρ_{f_1} is the cost of evaluating f_1) and $\rho_{f_2} 2^{|k_2|+|k_3|} + \rho_A |k_3| 2^{|k_2|+|k_3|}$ time to compute the $\boldsymbol{\lambda}_2^{k_2}$. The memory requirement is $2^{|k_0|+|k_1|} + 2^{|k_2|+|k_3|}$ registers of n bits.

   (c) Compute Q^{k_1,k_2} for all the values of k_1 and k_2:

      i. Multiply each column of a copy of $\widehat{A}$ by $\boldsymbol{\lambda}_1^{k_1}$ and each row by $\boldsymbol{\lambda}_2^{k_2}$ elementwise.
      ii. Apply the FWHT on all the rows and columns of this matrix to obtain Q^{k_1,k_2}.

   Since these calculations need to be done for each one of the $2^{|k_1|+|k_2|}$ guesses for the inner keys, the overall time cost is $2\rho_M 2^{|k_0|+|k_1|+|k_2|+|k_3|} + \rho_A \left(|k_0|+|k_3|\right) 2^{|k_0|+|k_1|+|k_2|+|k_3|}$, while $2^{|k_0|+|k_1|+|k_2|+|k_3|}$ memory registers of n bits are required.

   (d) Select the subkey guesses with the largest values for $|q(k_P, k_1, k_2, k_C)|$.
   The cost is $\rho_C 2^{|k_0|+|k_1|+|k_2|+|k_3|}$.

3. **Search phase:** The selected candidate subkeys are tested by searching for the rest of the key exhaustively. The cost is $\rho_E 2^{\kappa-a}$.

Algorithm 6: General key recovery algorithm using FFT (without the final phase)

Input: A collection $\mathcal{D} = \{(x, y = E_K(x))\}$ of N plaintext-ciphertext pairs (possibly on-the-fly).
Output: The experimental correlations $Q^{k_1,k_2}_{k_0,k_3}$.

```
// DISTILLATION PHASE
```
$A \leftarrow 0$;
forall $(x, y) \in \mathcal{D}$ **do** $a_{x|_{\chi_0}, y|_{\chi_3}} \leftarrow a_{x|_{\chi_0}, y|_{\chi_3}} + 1$;
```
// ANALYSIS PHASE
```
for $i \leftarrow 0$ **to** $2^{|k_0|} - 1$ **do** $A_{i\cdot} \leftarrow \mathrm{FWHT}(A_{i\cdot})$; `// Apply the FWHT to the rows of A`
for $j \leftarrow 0$ **to** $2^{|k_3|} - 1$ **do** $A_{\cdot j} \leftarrow \mathrm{FWHT}(A_{\cdot j})$; `// Apply the FWHT to the columns of A`
for $k_1 \leftarrow 0$ **to** $2^{|k_1|} - 1$; $i \leftarrow 0$ **to** $2^{|k_0|} - 1$ **do** $(\lambda_1^{k_1})_i \leftarrow f_1(i, k_1)$; `// First column of` B^{k_1}
for $k_2 \leftarrow 0$ **to** $2^{|k_2|} - 1$; $j \leftarrow 0$ **to** $2^{|k_3|} - 1$ **do** $(\lambda_2^{k_2})_j \leftarrow f_2(j, k_2)$; `// First column of` C^{k_2}
for $k_1 \leftarrow 0$ **to** $2^{|k_1|} - 1$ **do** $\boldsymbol{\lambda}_1^{k_1} \leftarrow \mathrm{FWHT}(\boldsymbol{\lambda}_1^{k_1})$; `// Compute` $\boldsymbol{\lambda}_1^{k_1}$
for $k_2 \leftarrow 0$ **to** $2^{|k_2|} - 1$ **do** $\boldsymbol{\lambda}_2^{k_2} \leftarrow \mathrm{FWHT}(\boldsymbol{\lambda}_2^{k_2})$; `// Compute` $\boldsymbol{\lambda}_2^{k_2}$
for $k_1 \leftarrow 0$ **to** $2^{|k_1|} - 1$; $k_2 \leftarrow 0$ **to** $2^{|k_2|} - 1$ **do** `// Compute` $Q^{k_1,k_2}_{k_0,k_3}$
  for $k_0 \leftarrow 0$ **to** $2^{|k_0|} - 1$; $k_3 \leftarrow 0$ **to** $2^{|k_3|} - 1$ **do** $Q^{k_1,k_2}_{k_0 k_3} \leftarrow A_{k_0 k_3} \cdot (\lambda_1^{k_1})_{k_0} \cdot (\lambda_2^{k_2})_{k_3}$;
  for $k_0 \leftarrow 0$ **to** $2^{|k_0|} - 1$ **do** $Q^{k_1,k_2}_{k_0\cdot} \leftarrow \mathrm{FWHT}(Q^{k_1,k_2}_{k_0\cdot})$;
  for $k_3 \leftarrow 0$ **to** $2^{|k_3|} - 1$ **do** $Q^{k_1,k_2}_{\cdot k_3} \leftarrow \mathrm{FWHT}(Q^{k_1,k_2}_{\cdot k_3})$;
end
return $\left\{Q^{k_1,k_2}\right\}_{k_1,k_2}$;

Result 3.1. *The time complexity of the general algorithm is (ignoring the significantly smaller terms)*

$$\underbrace{\rho_D N}_{\substack{distillation\\phase}} + \underbrace{2\rho_M 2^{|k_0|+|k_1|+|k_2|+|k_3|} + \rho_A\left(|k_0|+|k_3|\right)2^{|k_0|+|k_1|+|k_2|+|k_3|}}_{analysis\ phase} + \underbrace{\rho_E 2^{\kappa-a}}_{search\ phase} \tag{3.23}$$

The memory use is $2^{|k_0|+|k_1|+|k_2|+|k_3|} + o\left(2^{|k_0|+|k_1|+|k_2|+|k_3|}\right)$ *memory registers of* n *bits.*

A look at the structure of the analysis phase shows that a time-memory tradeoff in the lower order terms can be achieved if some of the steps are mixed: for example, slightly less memory is required if the eigenvalue vectors are only computed as needed (this has a small additional time cost, however). Furthermore, if the attacker doesn't need to store all the experimental correlations, it is possible to compute the matrices Q^{k_1,k_2} one by one without storing them, so that only $2 \cdot 2^{|k_0|+|k_3|}$ memory registers of n bits are used.

This algorithm also generalises the case in which no part of the key is guessed at the beginning of the cipher (for example, the original attack of [CSQ07]): it suffices to only compute the experimental correlations corresponding to the case $k_0 = 0$. This means that instead of applying the FWHT on all the rows of A, it suffices to right-multiply A by the first column of C to obtain $\mathbf{a}$. This is different from the original algorithm that was shown in [CSQ07], as the calculation of the vector $\mathbf{a}$, which was originally done in one step, is now separated into two different computations. This is slightly more costly in memory, but has the advantage that, in the case of multiple linear cryptanalysis, only one distillation phase needs to be performed, instead of one distillation for each possibility for the input mask α.

We also compared the computational costs $\rho_D, \rho_{f_1}, \rho_{f_2}, \rho_C, \rho_A, \rho_M$ and ρ_E. In general, ρ_D, ρ_{f_1} and ρ_{f_2} should be negligible when compared to the others, as they are much simpler operations. For most cases $\rho_C \simeq \rho_A$ and ρ_M should be comparable to or smaller than ρ_E (this depends on the implementations of the cipher and the operations).

The adaptability of this algorithm to multiple and multidimensional linear attacks was also considered in [NWW11], [ZZ15] and [BTV18]. For the general algorithm, since the distillation phase only needs to be performed once, we arrive at the following complexity:

Result 3.2. *A multiple/multidimensional Algorithm 2 attack using* M *approximations with the extended FFT algorithm has complexity (ignoring the significantly smaller terms)*

$$\underbrace{\rho_D N}_{\substack{distillation\\phase}} + \underbrace{2M\rho_M 2^{|k_0|+|k_1|+|k_2|+|k_3|} + M\rho_A(|k_0|+|k_3|)2^{|k_0|+|k_1|+|k_2|+|k_3|}}_{analysis\ phase} + \underbrace{\rho_E 2^{\kappa-a}}_{search\ phase} \tag{3.24}$$

If there are several approximations which share the same input mask α but differ in their output masks (or the other way around), then it is possible to reuse some partial results such as $B^{k_1}\widehat{A}$, which only need to be computed once. This can lead to a further reduction of the time complexity.

3.3 Exploiting the key schedule

In the algorithm of the previous subsection, the attacker has no prior knowledge of the relationship between the different round subkeys: they guess $|k_0|+|k_1|+|k_2|+|k_3|$ bits of subkey independently. However, real ciphers have a key schedule, which means that relationships between the different subkeys will exist. In a "classical" implementation of Matsui's Algorithm 2 over multiple rounds (that is, without using the FWHT), these relationships are easily exploited: if k_0, k_1, k_2 and k_3 can all be deduced from $|k_T|$ bits of information about the subkey, then the time complexity of the attack can be reduced to $N2^{|k_T|}$ or $O(N) + 2^{2|k_T|}$, depending on the chosen variant of the algorithm.

This subsection deals with how the known dependencies between subkey bits which are induced by the key schedule can be used to improve the time complexity of our algorithm, especially in the

case of multiple linear cryptanalysis. Two different but compatible strategies will be introduced: the first exploits bit dependencies when computing the experimental correlations for each one of the approximations. The second computes the experimental correlations for each approximation using only the subkey bits which are strictly necessary, and then combines all the information to obtain the Q_k statistics.

3.3.1 Top-down strategy

The first approach consists of directly trying to use the dependency relationships between the subkey bits in order to reduce the time complexity of the general algorithm, so it can be used for simple, multiple and multidimensional linear cryptanalysis. We have so far been unable to provide an efficient general algorithm which takes account of all dependency relationships between k_0, k_1, k_2, k_3. However, we have considered several possible instances in which the attacker can obtain a time complexity gain by using their knowledge of the subkeys that's derived from the key schedule:

- *Dependency within k_1 or within k_2:* This case is already covered by the general algorithm, as it is sufficient to redefine k_1 and k_2 with a smaller number of bits.
- *Dependency between k_1 and k_2:* If k_1 and k_2 can be deduced from $|k_T| < |k_1| + |k_2|$ bits of information about the master key, then it is possible to only consider the matrices Q^{k_1,k_2} corresponding to all possible values of k_T while running the algorithm. The time complexity of the analysis phase is thus reduced to

$$2\rho_M 2^{|k_0|+|k_T|+|k_3|} + \rho_A(|k_0|+|k_3|)2^{|k_0|+|k_T|+|k_3|} \tag{3.25}$$

- *Dependency within k_0 or within k_3:* If the key schedule means that k_0 (or k_3) can't take all the $2^{|k_0|}$ possible values (one particular case is when the round subkey is only XORed with a part of the state, that is, when some bits of k_0 are always zero), then the attacker doesn't need to compute $q(k_0, k_1, k_2, k_3)$ for the impossible values of k_0. This can be done by ignoring these output positions when computing the FWHTs of step 2c of the algorithm (this is often called a "pruned" transform, see [JSD01],[HW04]). This can lead to a gain in time complexity which depends on the set of required outputs.
- *Dependency between k_0 and k_3:* If some bits of k_3 can be deduced from k_0, then the last set of FWHTs can be pruned according to the possible values of k_0 which are associated to the row of $Q^{k_1 k_2}$ currently being calculated.
- *Dependency between k_0 and $(k_1 \| k_2)$ (or $(k_1 \| k_2)$ and k_3):* If some bits of k_0 (or k_3) can be deduced from k_1 and k_2, then the attacker can use the same strategy of pruning the FWHT as in the previous cases. This time, however, the set of possible values of k_0 and k_3 changes according to the value of k_1 and k_2 currently under consideration.

3.3.2 Bottom-up strategy: multiple linear cryptanalysis

Another approach which is specific to multiple attacks is to use the FWHT algorithm to compute the experimental correlation for each approximation but considering only the subkey bits which are strictly necessary, and then combining the information from all the approximations to obtain Q_k more quickly. This is similar to the approach of [ZZ15] and [BTV18] and seems to allow for a greater flexibility than the top-down strategy. Let $\nu_i : \alpha_i \cdot \hat{x} \oplus \beta_i \cdot \hat{y}, i = 1, \ldots, M$ be linear approximations of the inner cipher E_M. Multiple linear cryptanalysis requires the attacker to compute

$$Q(k_0, k_1, k_2, k_3) = \frac{1}{N} \sum_{i=1}^{m} \left(q^i(k_0, k_1, k_2, k_3)\right)^2$$

In order to calculate one particular $q^i(k_0, k_1, k_2, k_3)$ some subkey bits might be unnecessary: some part of the subkey might be necessary for one approximation but not for a different one. Let us suppose

that $q^i(k_0, k_1, k_2, k_3)$ can be calculated as $\hat{q}^i(k_0^i, k_1^i, k_2^i, k_3^i)$ (where $k_0^i = k_0|_{\chi_0^i}$ is a part of k_0, and so on), and that all the k_0^i, k_1^i, k_2^i and k_3^i (for all values of i) can be extracted from a part k_T of the master key K, so $Q(k_0, k_1, k_2, k_3) = \hat{Q}(k_T)$. We will also suppose that the sets of masks (χ_0^i, χ_3^i) and $(\chi_0^i, \chi_1^i, \chi_2^i, \chi_3^i)$ take l_1 and l_2 different values over the set of M approximations, respectively. In this situation, the attacker can perform a modified algorithm:

1. Perform a modified distillation phase in which l_1 tables instead of one are constructed: one for each plaintext-ciphertext mask pair (χ_0^i, χ_3^i).

2. For each approximation ν_i, compute a table of length $2^{|k_0^i|+|k_1^i|+|k_2^i|+|k_3^i|}$ containing all the possible values of $q^i(k_0^i, k_1^i, k_2^i, k_3^i)$ by using the algorithm from the previous subsection and the appropiate table from the distillation phase.

3. The M tables from the previous step are combined into l_2 "condensed" tables by adding the square correlations of approximations corresponding to the same choice of subkey bits, that is, one table for each possible value of $(\chi_0^i, \chi_1^i, \chi_2^i, \chi_3^i)$. In other words, given a fixed set of masks (X_0, X_1, X_2, X_3), the associated condensed table contains the coefficients:

$$\sum_{\substack{(\chi_0^i, \chi_1^i, \chi_2^i, \chi_3^i) \\ = (X_0, X_1, X_2, X_3)}} \left(q^i(k_0^i, k_1^i, k_2^i, k_3^i)\right)^2 \quad \text{for all} \quad (k_0^i, k_1^i, k_2^i, k_3^i) \tag{3.26}$$

4. For each possible guess of the partial master key k_T, use the key schedule to compute the associated values of $k_0^i, k_1^i, k_2^i, k_3^i$. Use the tables from the previous step to compute $Q(k_T)$. Keep the key candidates with a large enough value of the statistic for the search phase.

Result 3.3. *The improved linear attack algorithm using the bottom-up strategy has the following time complexity (ignoring significantly smaller terms):*

$$\underbrace{l_1 \rho_D N}_{\substack{\text{distillation} \\ \text{phase}}} + \underbrace{\sum_{i=1}^{M} \left(3\rho_M + (|k_0^i| + |k_3^i|)\rho_A\right) 2^{|k_0^i|+|k_1^i|+|k_2^i|+|k_3^i|}}_{\text{analysis phase I}} + \underbrace{l_2 \rho_A 2^{|k_T|}}_{\substack{\text{analysis} \\ \text{phase II}}} + \underbrace{\rho_E 2^{\kappa - a}}_{\text{search phase}} \tag{3.27}$$

and requires $\sum_{i=1}^{M} 2^{|k_0^i|+|k_1^i|+|k_2^i|+|k_3^i|}$ *memory registers.*

This algorithm can produce large gains in the case of multiple linear cryptanalysis, but its success is more limited in multidimensional attacks, as there is always a linear approximation of maximum Hamming weight masks for which $|k_0^i| = |k_0|, \ldots, |k_3^i| = |k_3|$.

4 Application to the block cipher PRESENT

In order to showcase the potential of our key recovery techniques, we will describe some new attacks on reduced-round variants of the block cipher PRESENT, which surpass any previously known attacks, linear or otherwise. Our new results on PRESENT are of independent interest by themselves: in particular, we will describe our attacks on the 26 and 27-round variants of PRESENT-80 with lower data and time complexities than those of previous attacks, and what, to the best of our knowledge, is the first attack on 28-round PRESENT-128.

PRESENT is a lightweight block cipher which was proposed by Bogdanov et al. in CHES 2007 (see [PRS07]) and was made an ISO standard in 2012. It is a very popular and thoroughly analysed cipher, and is also the inspiration of several recent lightweight ciphers such as GIFT ([GFT17]) and TRIFLE-BC ([TRF19]). It has a block size of 64 bits, and there are two variants with 80 and 128 bit key lenghs (which we'll refer to as PRESENT-80 and PRESENT-128, respectively). Ever since its proposal, PRESENT has been subject to substantial cryptanalisis efforts, which have slowly eroded its security margin. The best known attacks on the PRESENT construction are linear ([ZZ15],[BTV18]), and are effective on up to 27 rounds (out of 31).

We will describe two new linear attacks on reduced-round variants of PRESENT which make use of different sets of linear approximations (or *linear distinguishers*). The first one only uses approximations with input and output masks of Hamming weight one, which makes the key recovery part of the attack light enough to allow attacks on up to 27-rounds of PRESENT-80. Later we will describe a more effective (in the sense that it has a larger capacity for the same number of rounds) linear distinguisher using masks of Hamming weight 2: this allows to reach 28 rounds, but also means that the key recovery is more expensive and the attack only applies to PRESENT-128.

This section is structured as follows: the first subsection contains a description of the PRESENT block cipher specifications for both key sizes. The next subsections explain the development of our linear attacks: from the selection of sets of suitable linear approximations to the key recovery algorithm. Finally, we provide some experimental verification of our claims on the data and time complexity of our attacks by implementing computer simulations on 10-round PRESENT.

4.1 Description of PRESENT

PRESENT is a key-alternating block cipher which takes a 64-bit plaintext $x = x_{63} \dots x_0$ and an 80-bit (or 128-bit) key $K = \kappa_{79} \dots \kappa_0$ (or $K = \kappa_{127} \dots \kappa_0$) and returns a 64-bit ciphertext $y = y_{63} \dots y_0$. The encryption is performed by iteratively applying a round transformation to the state $b = b_{63} \dots b_0 = w_{15} \| \dots \| w_0$, where each of the w_i represents a 4-bit nibble, $w_i = b_{4i+3} b_{4i+2} b_{4i+1} b_{4i}$.

Both variants of PRESENT consist of 31 rounds, plus the addition of a final whitening key at the output. Each round is the composition of the following three transformations:

- **addRoundKey:** Given the round key $K_i = \kappa^i_{63} \dots \kappa^i_0,\ 0 \le i \le 31$ and the state b, the round key is XORed bitwise to the state.
- **sBoxLayer:** A fixed 4-bit to 4-bit S-box $S : \mathbb{F}_2^4 \longrightarrow \mathbb{F}_2^4$ is applied to each 4-bit nibble of the state w_i. The S-box S is given as a lookup table (here in hexadecimal notation):

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

- **pLayer:** A fixed bitwise permutation P is applied to the state b. The permutation P is

$$\begin{array}{rccc} P: & \{0, \dots 63\} & \longrightarrow & \{0, \dots, 63\} \\ & j \neq 63 & \longmapsto & 16j \bmod 63 \\ & 63 & \longmapsto & 63 \end{array} \tag{4.1}$$

  Alternatively, P can be given in the form of a lookup table:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

All the components of a PRESENT round are invertible, so it is easy to invert each round and therefore the whole cipher. The following algorithm summarises the structure of PRESENT:

Algorithm 7: The PRESENT block cipher

Input: A 64-bit plaintext x, an 80 or 128-bit key K.
Output: A 64-bit ciphertext y.
$b \leftarrow x$;
$\mathbf{K} \leftarrow \text{KeySchedule}(K,31)$; // Compute all the round subkeys
for $i \leftarrow 0$ **to** 30 **do**
  $b_{63}, \ldots, b_0 \longleftarrow b_{63} \oplus \kappa^i_{63}, \ldots, b_0 \oplus \kappa^i_0$; // addRoundKey
  $w_{15} \| \ldots \| w_0 \longleftarrow S(w_{15}) \| \ldots \| S(w_0)$; // sBoxLayer
  $b_{63}, \ldots, b_0 \longleftarrow b_{P^{-1}(63)}, \ldots, b_{P^{-1}(0)}$; // pLayer
end
$b_{63}, \ldots, b_0 \longleftarrow b_{63} \oplus \kappa^{31}_{63}, \ldots, b_0 \oplus \kappa^{31}_0$; // Whitening key
return b

The only element left to describe is the key schedule, which is the only difference between both variants:

Algorithm 8: Key schedule of PRESENT-80

Input: A master key K of 80 bits, a number of rounds r.
Output: $r+1$ round subkeys K_i of 64 bits.
$\kappa^0_{63} \ldots \kappa^0_0 \longleftarrow \kappa_{79} \ldots \kappa_{16}$; // Extract the first round subkey
for $i \leftarrow 1$ **to** r **do**
  $\kappa_{79} \ldots \kappa_0 \longleftarrow \kappa_{18} \ldots \kappa_{19}$; // Rotate key register 19 bits to the right
  $\kappa_{79}\kappa_{78}\kappa_{77}\kappa_{76} \longleftarrow S(\kappa_{79}\kappa_{78}\kappa_{77}\kappa_{76})$; // Apply S-box on leftmost nibble
  $\kappa_{19}\kappa_{18}\kappa_{17}\kappa_{16}\kappa_{15} \longleftarrow \kappa_{19}\kappa_{18}\kappa_{17}\kappa_{16}\kappa_{15} \oplus i$; // Add 5-bit round counter
  $\kappa^i_{63} \ldots \kappa^i_0 \longleftarrow \kappa_{79} \ldots \kappa_{16}$; // Extract round subkey
end
return $\{K_i\}_{i=0}^r$;

Algorithm 9: Key schedule of PRESENT-128

Input: A master key K of 128 bits, a number of rounds r.
Output: $r+1$ round subkeys K_i of 64 bits.
$\kappa^0_{63} \ldots \kappa^0_0 \longleftarrow \kappa_{127} \ldots \kappa_{64}$; // Extract the first round subkey
for $i \leftarrow 1$ **to** r **do**
  $\kappa_{127} \ldots \kappa_0 \longleftarrow \kappa_{66} \ldots \kappa_{67}$; // Rotate key register 61 bits to the left
  $\kappa_{127}\kappa_{126}\kappa_{125}\kappa_{124} \longleftarrow S(\kappa_{127}\kappa_{126}\kappa_{125}\kappa_{124})$;
  $\kappa_{123}\kappa_{122}\kappa_{121}\kappa_{120} \longleftarrow S(\kappa_{123}\kappa_{122}\kappa_{121}\kappa_{120})$; // Apply S-box on 2 leftmost nibbles
  $\kappa_{66}\kappa_{65}\kappa_{64}\kappa_{63}\kappa_{62} \longleftarrow \kappa_{66}\kappa_{65}\kappa_{64}\kappa_{63}\kappa_{62} \oplus i$; // Add 5-bit round counter
  $\kappa^i_{63} \ldots \kappa^i_0 \longleftarrow \kappa_{127} \ldots \kappa_{64}$; // Extract i-th round subkey
end
return $\{K_i\}_{i=0}^r$;

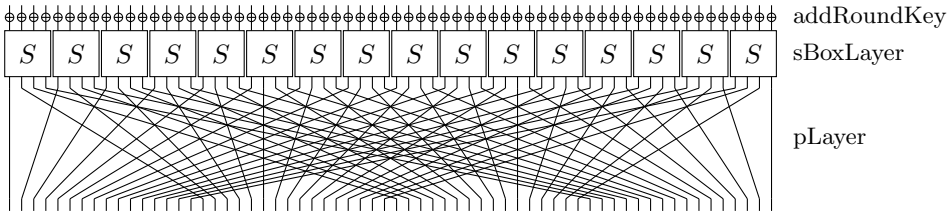

Figure 3: Graphic representation of one round of PRESENT.

4.2 The 1-bit linear distinguisher for up to 23 rounds

This subsection provides a multiple linear distinguisher which can be used on up to 23 rounds of PRESENT (thus allowing for an Algorithm 2 attack on up to 27 rounds by adding two rounds at the beginning and two at the end), and only uses approximations with input and output masks of Hamming weight 1 (which will be called 1-bit approximations). We begin our analysis by looking at the Linear Approximation Table of the PRESENT S-box (table 1).

This table shows that the S-box has eight (out of the total sixteen) biased linear approximations for which both the input and output masks have Hamming weight 1. These approximations can be used to construct linear trails which only have one active S-box on each round and involve one bit of each round subkey. These trails lead to the presence of approximations for multiple rounds with one-bit input and output masks and a large ELP. This phenomenon is exploited by all the linear attacks on PRESENT that we are aware of, such as [Oh09], [NSZ09], [Ch10], [ZZ15] and [BTV18].

We will now select a few high-bias one-bit approximations of 22/23 rounds of PRESENT which we will use on the 26/27-round attack of the next subsection. Since there are only $64 \times 64 = 2^{12}$ possible one-bit approximations, we can just estimate the ELP of all of them and keep the ones with the largest values. To this end we used the correlation matrix method considering all linear trails with up to two active S-boxes on each intermediate round (this means that the correlation submatrix is of size 2800×2800). The results of this computation have been condensed into table 2.

		Output mask β															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Input mask α	**0**	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	1	-	-	-	-	-	-4	-	-4	-	-	-	-	-	-4	-	4
	2	-	-	2	2	-2	-2	-	-	2	-2	-	4	-	4	-2	2
	3	-	-	2	2	2	-2	-4	-	-2	2	-4	-	-	-	-2	-2
	4	-	-	-2	2	-2	-2	-	4	-2	-2	-	-4	-	-	-2	2
	5	-	-	-2	2	-2	2	-	-	2	2	-4	-	4	-	2	2
	6	-	-	-	-4	-	-	-4	-	-	-4	-	-	4	-	-	-
	7	-	-	-	4	4	-	-	-	-	-4	-	-	-	-	4	-
	8	-	-	2	-2	-	-	-2	2	-2	2	-	-	-2	2	4	4
	9	-	4	-2	-2	-	-	2	-2	-2	-2	-4	-	-2	2	-	-
	10	-	-	4	-	2	2	2	-2	-	-	-	-4	2	2	-2	2
	11	-	-4	-	-	-2	-2	2	-2	-4	-	-	-	2	2	2	-2
	12	-	-	-	-	-2	-2	-2	-2	4	-	-	-4	-2	2	2	-2
	13	-	4	4	-	-2	-2	2	2	-	-	-	-	2	-2	2	-2
	14	-	-	2	2	-4	4	-2	-2	-2	-2	-	-	-2	-2	-	-
	15	-	4	-2	2	-	-	-2	-2	-2	2	4	-	2	2	-	-

Table 1: Linear Approximation Table (LAT) for the S-box of PRESENT. The entries with value 0 have been indicated with a - to facilitate reading. Furthermore, the entries corresponding to input and output masks of Hamming weight 1 have been highlighted.

Class	Input bits	Output bits	Qty.	ELP 22	ELP 23
A	**21**,**22**,**25**,**26**,**37**,**38**,**41**,**42**	**21**,**23**,**29**,**31**,**53**,**55**,**61**,**63**	64	$2^{-61.5}$	$2^{-63.4}$
B1	**21**,**22**,**25**,**26**,**37**,**38**,**41**,**42**	**22**,25,27,**30**,37,39,45,47,**54**,57,59,**62**	96	$2^{-62.2}$	$2^{-64.1}$
B2	23,27,29,30,39,43,45,46,**53**,**54**,**57**,**58**	**21**,**23**,**29**,**31**,**53**,**55**,**61**,**63**	96		
C1	21,22,25,26,37,38,41,42	26,38,41,43,46,58	48	$2^{-62.9}$	$2^{-64.8}$
C2	23,27,29,30,39,43,45,46,53,54,57,58	22,25,27,30,37,39,45,47,54,57,59,62	144		
C3	31,47,55,59,61,62	21,23,29,31,53,55,61,63	48		
D1	21,22,25,26,37,38,41,42	42	8	$2^{-63.6}$	$2^{-65.5}$
D2	23,27,29,30,39,43,45,46,53,54,57,58	26,38,41,43,46,58	72		
D3	31,47,55,59,61,62	22,25,27,30,37,39,45,47,54,57,59,62	72		
D4	63	21,23,29,31,53,55,61,63	8		
E1	23,27,29,30,39,43,45,46,53,54,57,58	42	12	$2^{-64.3}$	$2^{-66.2}$
E2	31,47,55,59,61,62	26,38,41,43,46,58	36		
E2	63	22,25,27,30,37,39,45,47,54,57,59,62	12		
F1	31,47,55,59,61,62	42	6	$2^{-64.3}$	$2^{-66.9}$
F2	63	26,38,41,43,46,58	6		
G	63	42	1	$2^{-65.0}$	$2^{-67.6}$

Table 2: Classification of the most correlated 1-bit approximations for 22 and 23 rounds of PRESENT. For each row of the table, an approximation can be constructed by masking one of the possible input bits and one of the possible output bits. The bits in bold represent the approximations which are used in our linear distinguisher.

There seems to be a structure in these approximations: the values of the ELP cluster around a few discrete values which differ from each other by a factor of $2^{-0.7}$, and the input and output bits can be classified in groups according to the correlation of their associated approximations. This might be caused by the interaction between the Linear Approximation Table of the PRESENT S-box and the permutation P, and it would be interesting to find a thorough explanation of this phenomenon.

For our attack we have chosen 128 linear approximations with high correlation: we use all 64 approximations from group A, 32 from group B1 (those corresponding to output bits 22,30,54,62) and 32 from group B2 (with input bits 53,54,57,58). If the reader looks at the figures in the next subsection, the choice of the approximations from groups B1 and B2 should become clearer: they involve the same subkey bits as the approximations in group A, thus minimizing the time cost of the key recovery. This set of linear approximations has a total capacity of $2^{-54.81}$ (for 22 rounds) and $2^{-56.71}$ (for 23 rounds).

It should be noted that these approximations are not statistically independent: they are not even linearly independent. One possible solution would be the application of multidimensional linear cryptanalysis, which would make the key recovery too costly as it would be necessary to consider all the linear combinations of our set of approximations, so the benefits of using approximations of Hamming weight 1 would be lost. Instead, the multiple linear cryptanalysis statistic will be used for these attacks, and we will estimate the probability of success under the assumption that the approximations are independent. In order to justify the validity of the resulting estimation, we will provide experimental results which conform to the theoretical predictions for a reduced number of rounds at the end of this section.

We have used the statistical model of [BN17] to compute the achievable advantage depending on the amount of data when the 22 and 23-round distinguishers are used to mount attacks on 26 and 27-round PRESENT, as well as comparing it to other previously known attacks. The results can be found on figure 4. For all the attacks, the probability of success has been fixed to 0.95, while a distinct known plaintext scenario is supposed (in other words, N is the number of strictly different available plaintexts). In both cases, our new linear distinguisher can either permit a lower data complexity or a larger advantage when using the same data when compared to previous attacks.

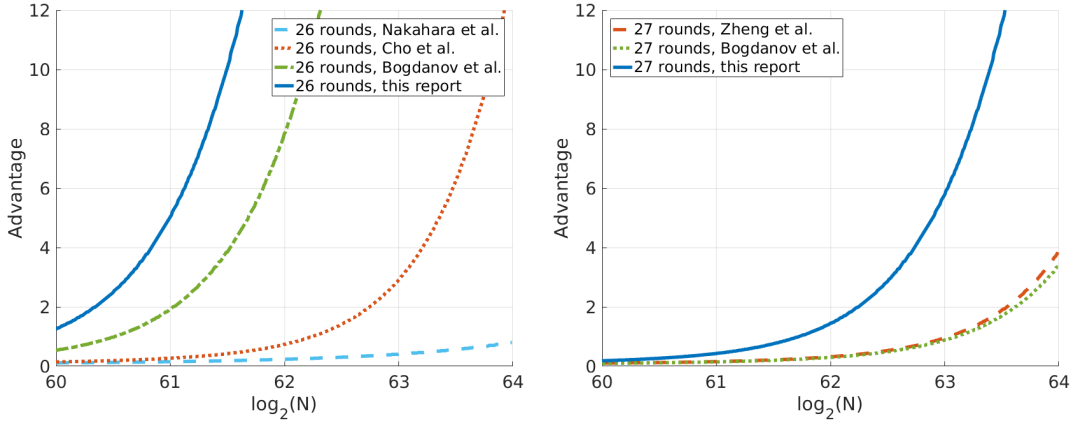

Figure 4: Comparison of the expected advantage (with 0.95 probability) of some known linear attack-sattacks on 26 and 27-round PRESENT for a given amount of data (in a DKP scenario).

4.3 Attacks on 26 and 27-round PRESENT-80

We will now proceed to use the linear distinguisher from the previous subsection to construct linear attacks on 26 and 27-round PRESENT-80. This is done by utilising Matsui's Algorithm 2 on the first two and the last two rounds of the cipher.

We begin by studying the first and last two rounds of PRESENT.

Proposition 4.1. (Key recovery on the first two rounds) *Let $\hat{x}$ be the state at the beginning of the second round of PRESENT. Given two fixed values of i, j between 0 and 3, the value of the four bits $\hat{x}_{48+4i+j}, \hat{x}_{32+4i+j}, \hat{x}_{16+4i+j}, \hat{x}_{4i+j}$ can be obtained from the 16 bits of the plaintext $x_{16j+15} \dots x_{16j}$, as well as the 16 bits of the first round subkey $\kappa^0_{16j+15} \dots \kappa^0_{16j}$ and the 4 bits of the second round subkey $\kappa^1_{16i+4j+3}\kappa^1_{16i+4j+2}\kappa^1_{16i+4j+1}\kappa^1_{16i+4j}$.*
In particular, for a fixed j the 16 bits of the state $\hat{x}_{60+j}, \hat{x}_{56+j}, \dots, \hat{x}_{4+j}, \hat{x}_j$ can be obtained from the 16 bits of the plaintext $x_{16j+15} \dots x_{16j}$, the 16 bits of the first round subkey $\kappa^0_{16j+15} \dots \kappa^0_{16j}$, and the 16 bits of the second round subkey

$$\kappa^1_{48+4j+3} \dots \kappa^1_{48+4j},\ \kappa^1_{32+4j+3} \dots \kappa^1_{32+4j},\ \kappa^1_{16+4j+3} \dots \kappa^1_{16+4j},\ \kappa^1_{4j+3}\kappa^1_{4j+2}\kappa^1_{4j+1}\kappa^1_{4j}$$

Proof. The reader is invited to convince themselves by looking at figure 5. We can also prove this formally by noting that the inverse permutation P^{-1} is of the form

$$P^{-1}(j) = 4j \bmod 63 \text{ if } j \neq 63,\ P^{-1}(63) = 63$$

This means that $\hat{x}_{16l+4i+j}$ is equal to the bit of position $P^{-1}(16l+4i+j) = 64l+16i+4j \bmod 63 = 16i+4j+l$ before the second application of pLayer. If we then undo sBoxLayer, we find that we need bits $\kappa^1_{16i+4j+3} \dots \kappa^1_{16i+4j}$ of the second round subkey K_1. If we continue by undoing the first round, we find that we need bits $P^{-1}(16i+4j+3) = 16j+12+i, P^{-1}(16i+4j+2) = 16j+8+i, P^{-1}(16i+4j+1) = 16j+4+i, P^{-1}(16i+4j) = 16j+i$ of the state before the first application of pLayer. Finally, by undoing the first sBoxLayer, we find that we require bits $x_{16j+15} \dots x_{16j}$ of the plaintext x and bits $\kappa_{16j+15} \dots \kappa_{16j}$ of the first round subkey K_0. □

Proposition 4.2. (Key recovery on the last two rounds) *Let $\hat{y}$ be the state after the application of addRoundKey in the second to last round of PRESENT. Given two fixed values of i, j between 0 and 3, the value of all the four bits $\hat{y}_{16i+4j+3}, \hat{y}_{16i+4j+2}, \hat{y}_{16i+4j+1}, \hat{y}_{16i+4j}$ can be obtained from*

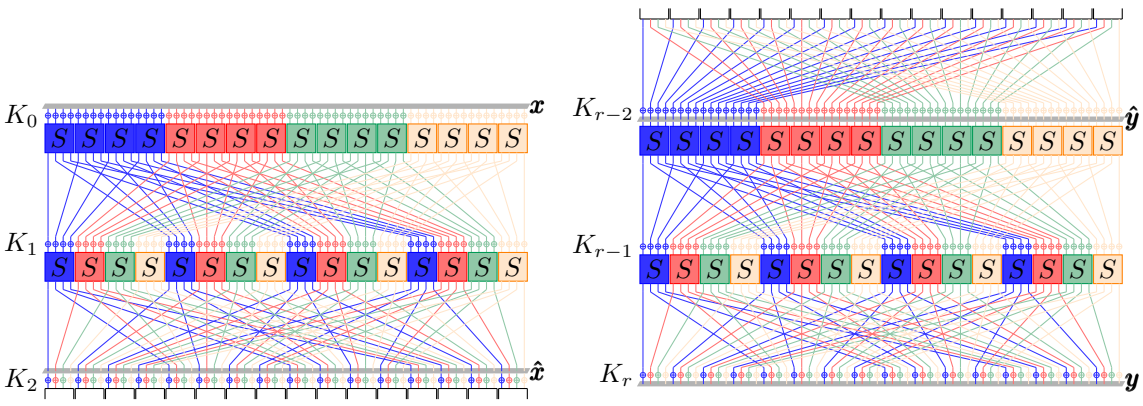

Figure 5: The four groups of bits for the key recovery on last two rounds.

the 16 bits of the ciphertext $y_{60+i}, y_{56+i}, \ldots, y_{4+i}, y_i$*, as well as the 16 bits of the last round subkey* $\kappa^r_{60+i}, \kappa^r_{56+i}, \ldots, \kappa^r_{4+i}, \kappa^r_i$ *and 4 bits of the second-to-last round subkey* $\kappa^{r-1}_{48+4i+j}\kappa^{r-1}_{32+4i+j}\kappa^{r-1}_{16+4i+j}\kappa^{r-1}_{4i+j}$*. In particular, for a fixed* i *the 16 bits of the state* $\hat{y}_{16i+15} \ldots \hat{y}_{16i}$ *can be obtained from the 16 bits of the ciphertext* $y_{60+i} \ldots y_{4+i}, y_i$*, the 16 bits of the last round subkey* $\kappa^r_{60+i} \ldots \kappa^r_{4+i}, \kappa^r_i$*, and the 16 bits of the second-to-last round subkey*

$$\kappa^{r-1}_{48+4i+3} \ldots \kappa^{r-1}_{48+4i},\ \kappa^{r-1}_{32+4i+3} \ldots \kappa^{r-1}_{32+4i},\ \kappa^{r-1}_{16+4i+3} \ldots \kappa^{r-1}_{16+4i},\ \kappa^{r-1}_{4i+3}\kappa^{r-1}_{4i+2}\kappa^{r-1}_{4i+1}\kappa^1_{4i}$$

Proof. The proof is analogous to that of the previous result. Figure 5 also illustrates this. □

With these observations and the bottom-up strategy from the previous section, we can describe an attack on 26-round PRESENT which uses the 22-round one-bit linear distinguisher between the third and the 24th rounds. If $\hat{x}$ denotes the state after the second round and $\hat{y}$ denotes the state before the 25th round, in order to evaluate the experimental correlation for all the approximations it is necessary to compute

$$\begin{aligned}&\hat{x}_{58}, \hat{x}_{57}, \hat{x}_{54}, \hat{x}_{53}, \hat{x}_{42}, \hat{x}_{41}, \hat{x}_{38}, \hat{x}_{37}, \hat{x}_{26}, \hat{x}_{25}, \hat{x}_{22}, \hat{x}_{21}\\ &\quad \in \{\hat{x}_{16r+4\cdot 2+2}, \hat{x}_{16r+4\cdot 2+1}, \hat{x}_{16r+4\cdot 1+2}, \hat{x}_{16r+4\cdot 1+1}, r = 0, \ldots, 3\}\end{aligned} \tag{4.2}$$

$$\begin{aligned}&\hat{y}_{63}, \hat{y}_{62}, \hat{y}_{61}, \hat{y}_{55}, \hat{y}_{54}, \hat{y}_{53}, \hat{y}_{31}, \hat{y}_{30}, \hat{y}_{29}, \hat{y}_{23}, \hat{y}_{22}, \hat{y}_{21}\\ &\quad \in \{\hat{y}_{16\cdot 3+4\cdot 3+s}, \hat{y}_{16\cdot 3+4\cdot 1+s}, \hat{y}_{16\cdot 1+4\cdot 3+s}, \hat{y}_{16\cdot 1+4\cdot 1+s}, s = 0, \ldots, 3\}\end{aligned} \tag{4.3}$$

For any individual one-bit to one-bit approximation, because of the previous results, we need to guess 16 bits of K_0, 4 bits of K_1, 4 bits of K_{25} and 16 bits of K_{26}. That means that for all approximations $|k^i_0| = |k^i_3| = 16$ and $|k^i_1| = |k^i_2| = 4$. We also know that the total number of subkey bits which require guessing (without taking the key schedule into account) in the key recovery is $|k_0| = |k_3| = 32$ and $|k_1| = |k_2| = 16$.

We will now consider the set of approximations as a whole in order to compute l_1 (the number of different plaintext-ciphertext masks) and l_2 (the number of different sets of subkey bits). Since all the input mask bits are of the form $\hat{x}_{4r+1}$ or $\hat{x}_{4r+2}$, there are two groups of bits of the plaintext which are relevant ($x_{31} \ldots x_{16}$ and $x_{47} \ldots x_{32}$). Analogously, since all the output bits are of the form $\hat{y}_{16+s}$ or $\hat{y}_{48+s}$, we conclude that only the groups of ciphertext bits $y_{61}, \ldots, y_5, y_1$ and $y_{63}, \ldots, y_7, y_3$ are necessary. This means that $l_1 \leq 2 \cdot 2 = 4$. A look at the approximations confirms that indeed $l_1 = 4$. In a similar manner, we can show that $l_2 = 4 \cdot 4 = 16$.

We also need to compute the number of master key bits which are actually involved in the key recovery, $|k_T|$. In order to do this, we must consider the key schedule in detail. We found that guessing the

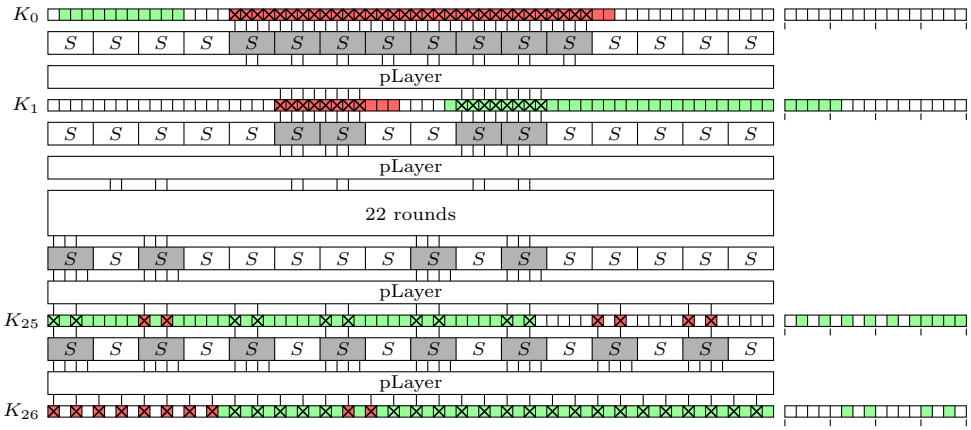

Figure 6: Our attack on 26-round PRESENT using 128 approximations with the analysis of the key schedule. In total there are 96 bits of the subkeys which need to be guessed, which have been indicated by a cross in the figure. However, they can all be deduced from the $|k_T|= 61$ bits of key which have been highlighted in (dark) red. From these bits of key, all the bits in (light) green can be deduced, which includes all the necessary bits for the attack. The 16 bits to the right represent the fraction of the key register which is not used for the current round subkey.

32 necessary bits of K_0 and 8 bits of K_1 already allows us to deduce 6 necessary bits of K_{25} and 16 necessary bits of K_{26}. Furthermore, by guessing two additional bits of K_0 and three additional bits of K_1, it is possible to compute 10 additional bits of K_{25} and K_{26}. This means that there are only 6 bits of K_{25} and 10 bits of K_{26} which need to be guessed separately. This makes for a total of $|k_T|= 61$ total information bits of the master key. The reader can study these considerations in further detail by looking at figure 6.

Finally, we must compare ρ_M and ρ_A to ρ_E. We can find a lower bound for ρ_E by noticing that in a PRESENT encryption 64 bit additions are required for the addition of each round subkey, while at least 64 bit operations are required for the application of sBoxLayer (at the very least, every bit of the new state must be computed). This means that we can assume that $\rho_E \geq 64 \cdot (27 + 26) = 3392$. On the other hand, we only expect to multipl and add integers of up to 64 bits. This means that $\rho_A \simeq 64$ and $\rho_M \simeq 3 \cdot 64^{1.58} \simeq 2142$ (using Karatsuba's algorithm). We conclude that the complexity of the attack (in 26-round encryptions) is $N + 2^{65} + 2^{80-a}$.

The fact that the time complexity of the analysis phase is "only" 2^{65} (when compared to the 80 bits of key) allows for a trade-off between the data complexity and the time complexity of the linear attack. As an extreme example, we can consider an attack with $N = 2^{61.9}$ (distinct) data complexity. This provides an advantage of 17 bits with probability 0.95, so that the search phase has a significantly smaller time complexity than the analysis phase and the whole time complexity of the attack is thus 2^{65}. On the other hand, a smaller (distinct) data complexity of $N = 2^{60.8}$ provides an advantage of 8 bits, thus increasing the time complexity to 2^{72} full round encryptions. The attack requires storing 2^{44} integers in memory.

In a similar fashion, we can design an attack on 27 rounds by using the same approximations between rounds 3 and 25. All the parameters for the key recovery algorithm are the same as on the 26-round attack, except for $|k_T|$, which in this case is 68 (see figure 7 for details). This means that an attack can be mounted with time complexity $2^{72}+2^{80-a}$. Since the time complexity of the analysis phase is larger in this attack because of the larger value of $|k_T|$, it is not possible to obtain better time complexities by incrementing the amount of data. When $N = 2^{63.4}$ distinct known plaintexts are available, the advantage is 8 bits with probability 0.95, and the overall time complexity of the attack is 2^{72} full round

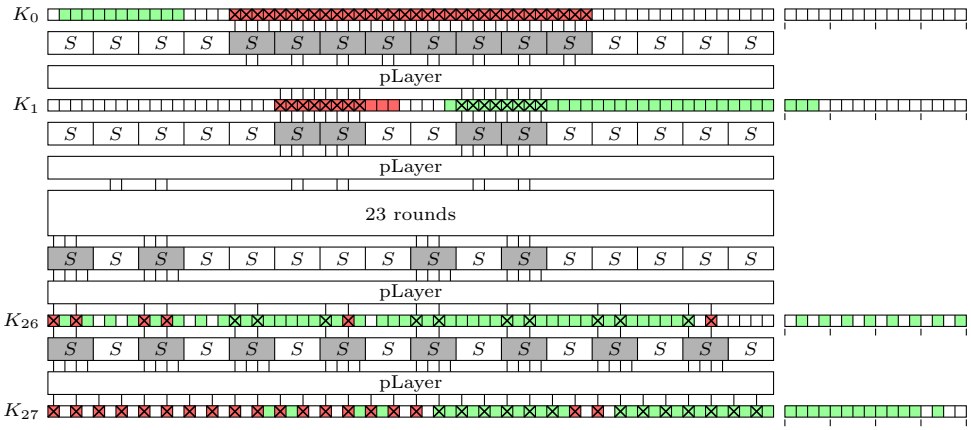

Figure 7: Our attack on 27-round PRESENT using 128 approximations and taking account of the key schedule. All the necessary subkey bits can be deduced from the $|k_T|= 68$ bits which have been highlighted in (dark) red.

encryptions. The memory requirement to perform this attack is again 2^{44} memory positions.

These attacks can be extended to the 128-bit key variant, since there is no need to exploit the key schedule because $|k_0|+|k_1|+|k_2|+|k_3|= 96 < 128$. The data complexities and the time complexities of the distillation and analysis phases of these attacks are the same, while the cost of the search phase increases with the key size.

4.4 The 2-bit linear distinguisher for up to 24 rounds

The capacity of our linear distinguisher using only approximations with masks of Hamming weight one diminishes as the number of rounds increases, so that the achievable advantage becomes too small when the number of rounds reaches 24 for the internal cipher (that is, when we attempt to attack 28-round PRESENT with the key recovery algorithm from the previous subsection). We will now construct a linear distinguisher for 24 rounds which uses approximations with masks of Hamming weight 1 or 2: this leads to a larger capacity because of the larger amount of available approximations, but it also means that the key recovery becomes more computationally expensive, so that the resulting attack will only compare favorably with a brute-force attack in the case of PRESENT-128.

In order to construct the new distinguisher, we start by looking for highly biased approximations over a larger search space than before. In particular, we considered approximations with up to two active S-boxes in their first and last rounds, and with input and output masks so that prolonging the approximation for an additional round at the beginning or the end would only require two additional active S-boxes (this is in fact equivalent to imposing that the mask for each of the active S-boxes in either the input or the output rounds has Hamming weight 1 or 2). Approximations with two active S-boxes in the first and last round but with a larger amount of active S-boxes when they are extended lead to a too expensive key recovery, so it makes sense to ignore them.

There are 2800 input masks and 2800 output masks which verify this property. A correlation matrix was constructed for one round of PRESENT, and by elevating the element-wise square of this matrix to the 24th power we obtained estimations of the ELPs of all the approximations with these input and output masks for up to 24 rounds. The results showed that no candidate approximations with two active S-boxes in either the first or the last round exhibit large biases (at least when compared to approximations where the first and last round only have one active S-box). We decided to classify the best approximations according to the active S-box in the first and the last round, as well as the input/output mask of that S-box (note that the final output mask is obtained after applying pLayer

Group	Input mask	Input S-box	Output mask	Output S-box	Qty.	Quantity (in distinguisher)	ELP (24)
A	**A**	**5,6,9,10**	**2,8,3,9**	**5,7,13,15**	64	64	$2^{-65.1}$
B	**C**	**5,6,9,10**	**2,8,3,9**	**5,7,13,15**	64	64	$2^{-65.6}$
C1	**A**	**5,6,9,10**	**2,8,3,9**	**6**,9,11,**14**	64	32	$2^{-65.8}$
C2	A	5,6,9,10	4,5	5,7,13,15	32	-	
C3	**A**	**7,11,13,14**	**2,8,3,9**	**5,7,13,15**	64	64	
D	**2,4**,3,5	**5,6,9,10**	**2,8,3,9**	**5,7,13,15**	256	128	$2^{-66.0}$
E1	**C**	**5,6,9,10**	**2,8,3,9**	**6**,9,11,**14**	64	32	$2^{-66.3}$
E2	C	5,6,9,10	4,5	5,7,13,15	32	-	
E3	**C**	**7,11,13,14**	**2,8,3,9**	**5,7,13,15**	64	64	
F1	A	5,6,9,10	2,8,3,9	10	16	-	$2^{-66.5}$
F2	A	5,6,9,10	4,5	6,9,11,14	32	-	
F3	A	5,6,9,10	6,C	5,7,13,15	32	-	
F4	A	7,11,13,14	2,8,3,9	6,9,11,14	64	-	
F5	A	7,11,13,14	4,5	5,7,13,15	32	-	
F6	A	15	2,8,3,9	5,7,13,15	16	-	
G1	2,4,3,5	5,6,9,10	2,8,3,9	6,9,11,14	256	-	$2^{-66.7}$
G2	2,4,3,5	5,6,9,10	4,5	5,7,13,15	128	-	
G3	8,9	5,6,9,10	2,8,3,9	5,7,13,15	64	-	
G4	2,4,3,5	7,11,13,14	2,8,3,9	5,7,13,15	256	-	

Table 3: Linear approximations for 24 rounds of PRESENT with input and output masks of Hamming weight 1 or 2 and an ELP larger than 2^{-67}. Note that the approximations from the one-bit distinguisher are all in groups D,G1 and G4. The approximations which have been chosen for the two-bit linear distinguisher are highlighted in a bold typeface.

on the mask of the S-box). The results are condensed in table 3. From these candidates, we have selected 448 approximations which lead to a total capacity of $2^{-56.98}$, and have been indicated on the table. As before, the choice of the approximations obeys both their ELP and their suitability for a reasonably expensive key recovery.

Once again, these approximations are not linearly independent, and we will justify our predictions about the advantage by providing experimental results. As before, we have estimated the advantage that would be provided by these approximations by using the model of [BN17]. If the attacker has access to the full codebook (that is, $N = 2^{64}$ distinct known plaintext-ciphertext pairs), then the advantage will be of 6 bits or more with probability 0.95.

4.5 Attack on 28-round PRESENT-128

The linear distinguisher from the previous subsection will now be used to construct a multiple linear attack on 28-round PRESENT-128, which to the best of our knowledge is the first to reach this number of rounds. We begin by slightly updating proposition 5 to account for the mask on the S-box of the last round instead of the output mask:

Proposition 4.3. (Key recovery on the last two rounds, revisited) *Let $\tilde{y}$ be the state after the application of sBoxLayer in the $(r-2)$-th round of PRESENT. Given two fixed values of i, j between 0 and 3, the value of all the four bits*

$$\tilde{y}_{16j+12+i} \oplus \kappa^{r-2}_{P(16j+12+i)},\ \tilde{y}_{16j+8+i} \oplus \kappa^{r-2}_{P(16j+8+i)},\ \tilde{y}_{16j+4+i} \oplus \kappa^{r-2}_{P(16j+4+i)},\ \tilde{y}_{16j+i} \oplus \kappa^{r-2}_{P(16j+i)} \quad (4.4)$$

can be obtained from the 16 bits of the ciphertext $y_{60+i}, y_{56+i}, \ldots, y_{4+i}, y_i$, as well as the 16 bits of the last round subkey $\kappa^r_{60+i}, \kappa^r_{56+i}, \ldots, \kappa^r_{4+i}, \kappa^r_i$ and the 4 bits of the second-to-last round subkey $\kappa^{r-1}_{48+4i+j}\kappa^{r-1}_{32+4i+j}\kappa^{r-1}_{16+4i+j}\kappa^{r-1}_{4i+j}$.

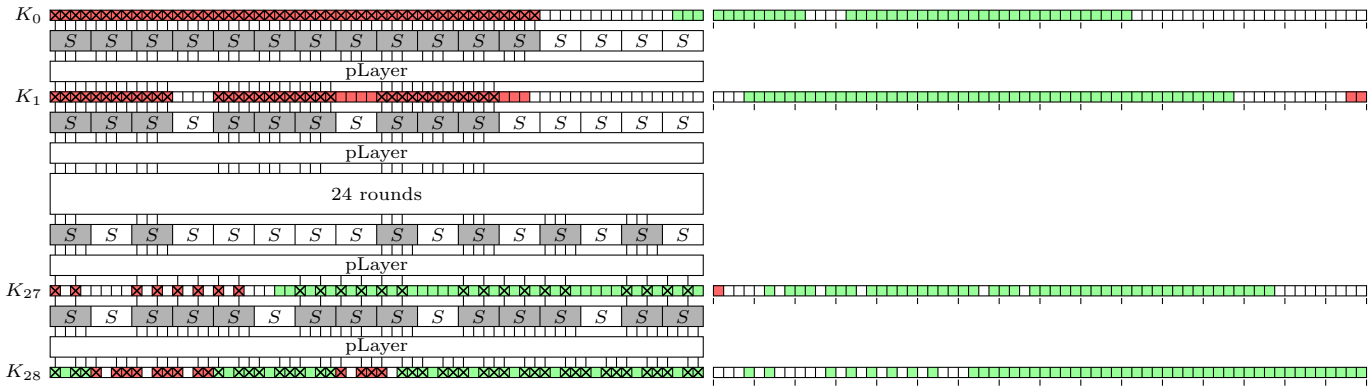

Figure 8: Our attack on 28-round PRESENT-128. In total there are 156 bits of the subkeys which need to be guessed, which have been indicated by a cross. However, if the $|k_T|= 114$ bits of key which have been highlighted in (dark) red are known, then all the bits in (light) green can be deduced, which include all the necessary bits for the attack. The 64 bits to the right represent the half of the key register which is not used for the current round subkey.

We can mount an attack using the bottom-up algorithm from the previous section, we just need to compute the parameters using the previous result as well as proposition 4.1. Since there are 4 different possibilities for the input mask in the active S-box for the first round and 4 possibilities for the output mask of the active S-box of the last round, we conclude that $l_2 = 4 \cdot 4 = 16$. With a similar argument but also taking the choice of the active S-box into account, we deduce that $l_1 = 12 \cdot 8 = 96$. It can also be shown that $|k_0^i| \leq 32$, $|k_1^i| \leq 8$, $|k_2^i| \leq 8$ and $|k_3^i| \leq 32$ for all approximations, and $|k_0| = 48, |k_1| = 36, |k_2| = 36, |k_3| = 48$. Figure 8 illustrates how $|k_T| = 114$. As discussed in the previous subsection, the advantage of the linear distinguisher is at least 6 bits with probability 0.95 when the full codebook of $N = 2^{64}$ plaintext-ciphertext pairs is available. The time complexity of the attack in number of 28-round PRESENT encryptions is thus bounded (with probability 0.95) by 2^{122} full encryptions. The attack requires $448 \cdot 2^{80} = 2^{89}$ memory positions, which is the bottleneck of the attack as it is, and more than is required to just store the 2^{64} plaintext-ciphertext pairs. We will try to reduce the memory requirements of this attack in the future.

4.6 Experimental verification

As was previously stated, our analysis of the achievable advantage of these attacks made some assumptions which do not completely hold. Apart from the linear dependence of the approximations conforming the linear distinguisher, there are other possible sources of problems: such as PRESENT not being a long-key cipher (which means that the ELP might not correspond to the real correlation of the approximations) and the fact that the wrong-key randomisation hypothesis doesn't hold for all approximations and all wrong keys (as not all bits of key influence the value of all approximations). For this reason it is necessary to provide some additional justification that these theoretical simplifications of the analysis still lead to reasonably accurate predictions.

Since it's practically impossible to simulate our attacks because of the high time complexity, we performed computer simulations of the same attacks on 10 rounds of PRESENT instead of 26/27/28, using the linear distinguishers between rounds 3 and 8. These distinguishers have a much higher capacity than those over a larger number of rounds, which means that a manageable number of plaintext-ciphertext pairs needs to be generated. Since it is also unfeasible to perform the full key recovery algorithm on a real computer, we decided to estimate the advantage instead by comparing the right key against a large number of random wrong keys. More specifically, for each randomly chosen

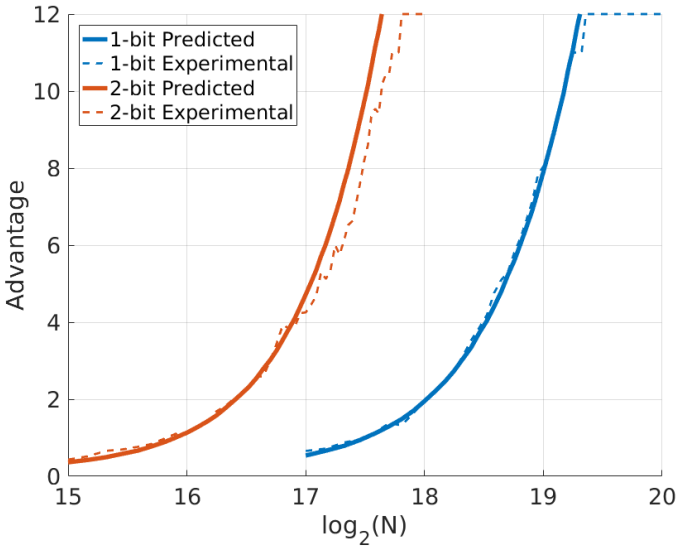

Figure 9: The results of our experiments testing the 1-bit and 2-bit distinguishers on 10-round PRESENT. The correlation for the right key was compared with 2^{12} other random keys, thus obtaining an estimation of the advantage for up to 12 bits. The experiment was repeated with 20 different master keys and 20 data samples for each key and value of N.

key and each randomly drawn sample of plaintexts, the right-key χ^2 statistic was computed. Then, 2^{12} random keys were drawn, and these keys were used to perform partial encryption and decryption of the first and last two rounds, in order to create a sample of 2^{12} wrong-key χ^2 statistics. The position of the right-key statistic among these provides an estimation of the advantage of up to 12 bits. This process was repeated for 20 different random right keys and 20 different random data samples, which provides a sample of 400 values of the advantage for a given value of N. The 5th percentile of these was used as an estimation of the advantage that's achieved with probability 0.95.

Figure 9 compares the observed results to the theoretical predictions for these attacks. Given the proximity of these results, we are confident that our predictions for attacks on a larger number of rounds will also be accurate, especially in the case of the 1-bit distinguisher for the attacks on 26 and 27 rounds.

5 Conclusion

Our analysis of the FFT/FWHT technique for linear cryptanalysis which was introduced in [CSQ07] has shown that the acceleration applies to most Algorithm 2 attacks over any number of rounds of key recovery. This algorithm also introduces a new distillation phase algorithm which extends the ideas of [NWW11] and allows the reuse of the distilled data in multiple and multidimensional linear cryptanalysis. We have also commenced the study of the compatibility of these accelerated attacks with the exploitation of the key schedule of the cipher. In particular, we have found that this can lead to effective results in the case of some multiple (but not multidimensional) attacks. However, we believe that there are many possible improvements on this area.

We have been able to showcase the real use of these theoretical developments by designing new attacks on reduced-round variants of PRESENT, which surpass any previously known attacks in terms of data and time complexity, and we have provided the first (bar biclique cryptanalysis) shortcut attack on 28-round PRESENT, albeit only for the 128-bit key variant, and with a high memory requirement. We hope to refine this attack in the future to make it compatible with the 80-bit variant, as well as reducing the memory requirement. Table 4 compares our attacks on PRESENT with other attacks found in the literature.

Our hope is that further developments in the algorithmic theory of linear cryptanalysis can make it more comparable with differential cryptanalysis, in the sense that some differential attacks showcase very intricate key recovery algorithms, which is fairly uncommon in linear attacks. We also believe that some block ciphers intended for lightweight cryptography with limited diffusion properties and simple key schedules might find their security eroded because of the possibility of linear attacks using multiple approximations: during this internship, I found a linear attack on the full-round primitive of the NIST Lightweight candidate TRIFLE ([TRF19])[1], which did not pass to the second round of the contest. Our intention is to publish a refined version of this attack with smaller data and time complexity in the near future.

Rounds	**Appr.**	**Capacity**	**Data**	**Time** ($\kappa = 80$)	**Time** ($\kappa = 128$)	**Memory**	P_S	**Source**
26	2995†	$2^{-55.58}$	$2^{63.8}$	2^{70}	2^{118}	2^{34}	0.95	[Ch10]
	135	$2^{-55.47}$	2^{62}	2^{72}	2^{120}	2^{48}	0.95	[BTV18]
	128	$2^{-54.81}$	$2^{61.9}$	2^{65}	2^{113}	2^{44}	0.95	This
	128	$2^{-54.81}$	$2^{60.8}$	2^{72}	2^{120}	2^{44}	0.95	This
27	405†	$2^{-55.33}$	$2^{63.8}$‡	2^{77}	2^{125}	2^{70}	0.95	[ZZ15]
	135	$2^{-58.06}$	$2^{63.8}$	$2^{77.5}$	$2^{125.5}$	2^{48}	0.95	[BTV18]
	128	$2^{-56.71}$	$2^{63.4}$	2^{72}	2^{120}	2^{44}	0.95	This
28	448	$2^{-56.98}$	2^{64}	-	2^{122}	2^{89}	0.95	This

†: Multidimensional linear cryptanalysis is used.

‡: Only one fourth of the known plaintexts is effectively used for each key guess.

Table 4: Comparison of linear attacks on PRESENT. All the time complexities have been evaluated using the same model assuming in a distinct known plaintext scenario.

[1] Our attack can be found on `https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/` as an official comment on the TRIFLE candidate or through **this link**.

6 References

[AABL12] Abdelraheem M.A., Ågren M., Beelen P., Leander G. (2012) *On the Distribution of Linear Biases: Three Instructive Examples.* In: Advances in Cryptology – CRYPTO 2012. CRYPTO 2012. Lecture Notes in Computer Science, vol 7417, pp 50-67. Springer.

[AM05] Åhlander K., Munthe-Kaas H. (2005) *Applications of the Generalized Fourier Transform on Numerical Linear Algebra.* In: Bit Numerical Mathematics, vol 45, pp 819-850. Springer.

[AR16] Ashur T., Rijmen V. (2016) *On Linear Hulls and Trails.* In: Progress in Cryptology - INDOCRYPT 2016. INDOCRYPT 2016. Lecture Notes in Computer Science, vol 10095, pp 269-286. Springer.

[GFT17] Banik S., Pandey S.K., Peyrin T, Sasaki Y., Sim S.M., Todo Y. (2017) *GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption.* In: Cryptographic Hardware and Embedded Systems – CHES 2017. CHES 2017. Lecture Notes in Computer Science, vol 10529, pp 321-345. Springer.

[BCQ04] Biryukov A., De Cannière C., Quisquater M. (2004) *On Multiple Linear Approximations.* In: Advances in Cryptology – CRYPTO 2004. CRYPTO 2004. Lecture Notes in Computer Science, vol 3152, pp 1-22. Springer.

[BN15] Blondeau C., Nyberg K. (2015) *Joint Data and Key Distribution of the Linear Cryptanalysis Test Statistic and its Impact to Data Complexity Estimates of Multiple/Multidimensional Linear and Truncated Differential Attacks.* In: IACR Cryptology ePrint Archive 2015, 935. International Association for Cryptologic Research.

[BN17] Blondeau C., Nyberg K. (2017) *Improved Parameter Estimates for Correlation and Capacity Deviates in Linear Cryptanalysis.* In: IACR Transactions on Symmetric Cryptology, 2016 (2), pp 162-191. International Association for Cryptologic Research.

[PRS07] Bogdanov A., Knudsen L.R., Leander G., Paar C., Poschmann A., Robshaw M.J.B., Seurin Y., Vikkelsoe C. (2007) *PRESENT: An Ultra-Lightweight Block Cipher.* In: Cryptographic Hardware and Embedded Systems - CHES 2007. CHES 2007. Lecture Notes in Computer Science, vol 4727, pp 450-466. Springer.

[BW12] Bogdanov A., Wang M. (2012) *Zero Correlation Linear Cryptanalysis with Reduced Data Complexity.* In: Fast Software Encryption. FSE 2012. Lecture Notes in Computer Science, vol 7549, pp 29-48. Springer.

[BGW14] Bogdanov A., Geng H., Wang M., Wen L., Collard B. (2014) *Zero-Correlation Linear Cryptanalysis with FFT and Improved Attacks on ISO Standards Camellia and CLEFIA.* In: Selected Areas in Cryptography - SAC 2013. SAC 2013. Lecture Notes on Computer Science, vol 8282, pp 306-323. Springer, Berlin, Heidelberg.

[BTV18] Bogdanov A., Tischhauser E., Vejre P. (2018) *Multivariate Profiling of Hulls for Linear Cryptanalysis.* In: IACR Transactions on Symmetric Cryptology, 2018 (1), pp 101-125. International Association for Cryptologic Research.

[Ca10] Carlet C. (2010) *Vectorial Boolean Functions for Cryptography.* In: Boolean Models and Methods in Mathematics, Computer Science, and Engineering (Encyclopedia of Mathematics and its Applications), pp. 398-470. Cambridge University Press.

[Ch10] Cho J.Y. (2010) *Linear Cryptanalysis of Reduced-Round PRESENT.* In: Topics in Cryptology - CT-RSA 2010. CT-RSA 2010. Lecture Notes in Computer Science, vol 5985, pp 302-317. Springer.

[CT65] Cooley J.W., Tukey, J.W. (1965) *An Algorithm for the Machine Calculation of Complex Fourier Series.* In: Mathematics of Computation, vol 19, pp 297-301. American Mathematical Society.

[CJM02] Chose P., Joux A., Mitton M. (2002) *Fast Correlation Attacks: An Algorithmic Point of View.* In: Advances in Cryptology — EUROCRYPT 2002. EUROCRYPT 2002. Lecture Notes in Computer Science, vol 2332, pp 209-221. Springer.

[CSQ07] Collard B., Standaert F.X., Quisquater JJ. (2007) *Improving the Time Complexity of Matsui's Linear Cryptanalysis.* In: Information Security and Cryptology - ICISC 2007. ICISC 2007. Lecture Notes in Computer Science, vol 4817, pp 77-88. Springer.

[CS09] Collard B., Standaert F.X. (2009) *A Statistical Saturation Attack against the Block Cipher PRESENT.* In: Topics in Cryptology – CT-RSA 2009. CT-RSA 2009. Lecture Notes in Computer Science, vol 5473, pp 195-210. Springer.

[AES02] Daemen J., Rijmen V. (2002) *The Design of Rijndael.* Springer.

[DR07] Daemen J., Rijmen V. (2007) *Probability distributions of correlation and differentials in block ciphers.* In: Journal of Mathematical Cryptology, vol 1, issue 3, pp 221-242. De Gruyter.

[TRF19] Datta N., Ghoshal A., Mukhopadhyay D., Patranabis S., Picek S., Sadhukhan R. (2019) *TRIFLE.* Submission to the NIST Lightweight Cryptography Competition.

[HCN08] Hermelin M., Cho J.Y., Nyberg K. (2008) *Multidimensional Linear Cryptanalysis of Reduced Round Serpent.* In: Information Security and Privacy. ACISP 2008. Lecture Notes in Computer Science, vol 5107, pp 203-215. Springer.

[HCN09] Hermelin M., Cho J.Y., Nyberg K. (2009) *Multidimensional Extension of Matsui's Algorithm 2.* In: Fast Software Encryption. FSE 2009. Lecture Notes in Computer Science, vol 5665, pp 209-227. Springer.

[HCN19] Hermelin M., Cho J.Y., Nyberg K. (2019) *Multidimensional Linear Cryptanalysis.* In: Journal of Cryptology, vol 32, pp 1-34. Springer.

[HW04] Hu Z., Wan H. (2004) *A novel generic Fast Fourier Transform pruning technique and complexity analysis.* In: IEEE Transactions on Signal Processing, vol 53, issue 1, pp 274-282. Institute of Electrical and Electronics Engineers.

[JSD01] Jankovic D., Stankovic R.S., Drechsler R. (2001) *Decision Diagram Method for Calculation of Pruned Walsh Transform.* In: IEEE Transactions on Computers, vol 50, issue 2, pp 147-157. Institute of Electrical and Electronics Engineers.

[KR94] Kaliski B.S., Robshaw M.J.B. (1994) *Linear Cryptanalysis Using Multiple Approximations.* In: Advances in Cryptology — CRYPTO 1994. CRYPTO 1994. Lecture Notes in Computer Science, vol 839, pp 26-39. Springer.

[KLLN15] Kaplan M., Laurent G., Leverrier A., Naya-Plasencia M. (2015) *Quantum differential and linear cryptanalysis.* In: IACR Transactions on Symmetric Cryptology 2016 (1), pp 71-94. International Association for Cryptologic Research.

[MaY93] Matsui M., Yamagishi A. (1993) *A New Method for Known Plaintext Attack of FEAL Cipher.* In: Advances in Cryptology — EUROCRYPT 1992. EUROCRYPT 1992. Lecture Notes in Computer Science, vol 658, pp 81-91. Springer.

[Ma94a] Matsui M. (1994) *Linear Cryptanalysis Method for DES Cipher.* In: Advances in Cryptology - EUROCRYPT 1993. EUROCRYPT 1993. Lecture Notes in Computer Science, vol 765, pp 386-397. Springer.

[Ma94b] Matsui M. (1994) *The First Experimental Cryptanalysis of the Data Encryption Standard.* In: Advances in Cryptology — CRYPTO 1994. CRYPTO 1994. Lecture Notes in Computer Science, vol 839, pp 1-11. Springer.

[Ma95] Matsui M. (1995) *On Correlation between the Order of S-boxes and the Strength of DES.* In: Advances in Cryptology — EUROCRYPT 1994. EUROCRYPT 1994. Lecture Notes in Computer Science, vol 950, pp 366-375. Springer.

[NWW11] Nguyen P.H., Wu H., Wang H. (2011) *Improving the Algorithm 2 in Multidimensional Linear Cryptanalysis.* In: Information Security and Privacy. ACISP 2011. Lecture Notes in Computer Science, vol 6812, pp 61-74. Springer.

[Ny95] Nyberg K. (1995) *Linear Approximation of Block Ciphers.* In: Advances in Cryptology — EUROCRYPT 1994. EUROCRYPT 1994. Lecture Notes in Computer Science, vol 950, pp 439-444. Springer.

[Ny18] Nyberg K. (2018) *Statistical and Linear Independence of Binary Random Variables.* In: IACR Cryptology ePrint Archive 2017, 432. International Association for Cryptologic Research.

[NSZ09] Nakahara J., Sepehrdad P., Zhang B., Wang M. (2009) *Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT.* In: Cryptology and Network Security. CANS 2009. Lecture Notes in Computer Science, vol 5888, pp 58-75. Springer.

[Oh09] Ohkuma K. (2009) *Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis.* In: Selected Areas in Cryptography. SAC 2009. Lecture Notes in Computer Science, vol 5867, pp 249-265. Springer.

[Le11] Leander G. (2011) *On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN.* In: Advances in Cryptology – EUROCRYPT 2011. EUROCRYPT 2011. Lecture Notes in Computer Science, vol 6632, pp 303-322. Springer.

[Se08] Selçuk A.A. (2008) *On Probability of Success in Linear and Differential Cryptanalysis.* In: Journal of Cryptology, vol 21, pp 131-147. Springer.

[ZZ15] Zheng L., Zhang S. (2015) *FFT-based Multidimensional Linear Attack on PRESENT using the 2-bit-Fixed Characteristic.* In: Security and Communication Networks, vol 8, pp 3535-3545. John Wiley and Sons.

A Mathematical background and notation

The purpose of this appendix is to provide an overview of some mathematical concepts which are used in the report, as well as establishing some notations, some of which are non-standard.

Binary vector spaces. We will begin by describing some properties of boolean vector spaces.

Definition A.1. *Let $\mathbb{F}_2^n$ be the vector space of dimension n over the finite field $\mathbb{F}_2 = \{0, 1\}$. There is an identification between the elements of this vector space and the natural numbers between 0 and $2^n - 1$, which takes the form of the following bijective map:*

$$\begin{array}{rcl} \mathbb{F}_2^n & \longleftrightarrow & \{0, \ldots, 2^n - 1\} \\ (x_{n-1}, \ldots, x_0) & \longmapsto & x_{n-1} \cdot 2^{n-1} + \ldots + x_1 \cdot 2 + x_0 \end{array} \tag{A.1}$$

These two descriptions of $\mathbb{F}_2^n$ are used without distinction in this report. We will always consider that the rightmost bit of a binary vector is the least significant one. Additionally, sometimes binary vectors will be represented without parentheses as is common in Computer Science literature. All real vectors and matrices in this report are considered with coordinates starting at 0 (instead of the usual 1).

We can define several operations on and between elements of $\mathbb{F}_2^n$:

Definition A.2. *Given $x \in \mathbb{F}_2^n$, its* Hamming Weight *$HW(x)$ is its number of non-zero coordinates. Given a binary vector x of any length, we will denote its length by $|x|$, that is, $|x| = n$ if $x \in \mathbb{F}_2^n$. We say that two vectors $x, y \in \mathbb{F}_2^n$ are* disjoint *if they don't share any non-zero coordinates. Given $x, y \in \mathbb{F}_2^n$, then $x \oplus y \in \mathbb{F}_2^n$ denotes the* addition *of x and y as vectors over $\mathbb{F}_2$. We use the symbol $\oplus$ instead of $+$ to distinguish this operation from addition modulo 2^n when describing the elements of $\mathbb{F}_2^n$ as integers, as well as to keep consistency with the previous literature on linear cryptanalysis. Given $x, y \in \mathbb{F}_2^n$, then $x \cdot y \in \mathbb{F}_2$ denotes the* inner product *of both vectors (which should not be confused with their product as integers), that is:*

$$x \cdot y = < x, y > = x_{n-1} \cdot y_{n-1} \oplus \ldots \oplus x_0 \cdot y_0 \in \mathbb{F}_2 \tag{A.2}$$

Given $x \in \mathbb{F}_2^n, y \in \mathbb{F}_2^m$, we can construct the concatenation

$$(x \| y) = (x_{n-1}, \ldots, x_0, y_{m-1}, \ldots, y_0) \in \mathbb{F}_2^{n+m} \tag{A.3}$$

Two maps $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^s$, $g : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2^s$ can also be concatenated:

$$\begin{array}{rccc} (f \| g) : & \mathbb{F}_2^{n+m} & \longrightarrow & \mathbb{F}_2^{r+s} \\ & (x \| y) & \longmapsto & (f(x) \| f(y)) \end{array} \tag{A.4}$$

Given $x, y \in \mathbb{F}_2^n$, we define the restricted vector *$x|_y \in \mathbb{F}_2^{HW(y)}$ as the vector of the components of x corresponding to the non-zero coordinates of y (this is not a standard notation).*

The following is a useful result about the inner product:

Proposition A.3. *For any $x \in \mathbb{F}_2^n$, the following equality holds:*

$$\sum_{y \in \mathbb{F}_2^n} (-1)^{y \cdot x} = \begin{cases} 0 & \text{if } x \neq \mathbf{0} \\ 2^n & \text{if } x = \mathbf{0} \end{cases} \tag{A.5}$$

Proof. The equality is clear when $x = \mathbf{0}$, since $(-1)^0 = 1$. In the case $x \neq \mathbf{0}$, the equality $y \cdot x = 0$ is a non-trivial linear equation over $\mathbb{F}_2^n$ which is satisfied by the elements of a hyperplane, which has dimension $n - 1$ and therefore has 2^{n-1} elements. The sum is thus $2^{n-1} \cdot (-1)^0 + 2^{n-1} \cdot (-1)^1 = 0$. □

Definition A.4. *A* boolean function *is any map* $f : \mathbb{F}_2^n \longmapsto \mathbb{F}_2$.

The correspondence between elements of $\mathbb{F}_2^n$ *and numbers between 0 and* 2^n-1 *induces a correspondence between the space of boolean functions over* $\mathbb{F}_2^n$ *and the vector space* $\mathbb{F}_2^{2^n}$*, as a boolean function can be seen as a vector whose components are the images of each element of* $\mathbb{F}_2^n$.

This identification is compatible with the notion of the addition of maps, in the sense that the addition of two boolean functions corresponds to the addition of their representation as vectors. Additionally, the product of two maps is the same as the element-wise product of their vectorial representations.

The following definition mimics the definitions of bias and correlation in linear cryptanalysis, so that the bias of a linear approximation ν is just the bias of the boolean function $f(x) = \alpha \cdot x \oplus \beta \cdot E(x)$.

Definition A.5. *The Hamming Weight of a boolean function* f *is*

$$HW(f) = \#\left\{x \in \mathbb{F}_2^n : f(x) = 1\right\} \tag{A.6}$$

The bias *of* f *is*

$$\varepsilon(f) = \frac{1}{2^n}\left(\#\left\{x \in \mathbb{F}_2^n : f(x) = 0\right\} - 2^{n-1}\right) \tag{A.7}$$

We also define the correlation *of* f *as*

$$c(f) = \frac{1}{2^n}\left(\#\left\{x \in \mathbb{F}_2^n : f(x) = 0\right\} - \#\left\{x \in \mathbb{F}_2^n : f(x) = 1\right\}\right) \tag{A.8}$$

If X *is a uniformly distributed random vector over* $\mathbb{F}_2^n$ *(or, equivalently, a uniformly distributed random variable over* $\{0, \ldots, 2^n - 1\}$*), then* $Y = f(X)$ *is a random variable in* $\mathbb{F}_2$*, and we have the following alternative definitions of bias and correlation:*

$$\varepsilon(f(X)) = Pr_X\left(f(X) = 0\right) - \frac{1}{2} \tag{A.9}$$

$$c(f(X)) = Pr_X\left(f(X) = 0\right) - Pr_X\left(f(X) = 1\right) \tag{A.10}$$

These magnitudes are all related by

Proposition A.6. *Let* $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ *be a boolean function. Then*

$$c(f) = 2\varepsilon(f) = \frac{1}{2} - \frac{1}{2^n}HW(f) \tag{A.11}$$

Proof. It suffices to look at the definitions and note that

$$\#\left\{x \in \mathbb{F}_2^n : f(x) = 1\right\} = 2^n - \#\left\{x \in \mathbb{F}_2^n : f(x) = 0\right\}$$

The relationship follows from this equality. □

The Fast Walsh-Hadamard Transform. We will now describe the Walsh-Hadamard Transform and its efficient computation algorithm.

Definition A.7. *We define the family of matrices*

$$H_1 = (1), \quad H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_4 = H_2 \otimes H_2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}, \tag{A.12}$$

$$H_{2^m} = H_2 \otimes H_{2^{m-1}} = \left(\begin{array}{c|c} H_{2^{m-1}} & H_{2^{m-1}} \\ \hline H_{2^{m-1}} & -H_{2^{m-1}} \end{array}\right) \in \mathbb{Z}^{2^m \times 2^m}$$

where $\otimes$ denotes the Kronecker (or tensor) product of matrices. These matrices are called Hadamard-Sylvester matrices. *The entry in the i-th row and j-th column of H_{2^m} is*

$$h_{ij}^{2^m} = (-1)^{i \cdot j}, 0 \leq i, j \leq 2^m - 1 \tag{A.13}$$

where $\cdot$ denotes the inner product of binary vectors. H_{2^m} is always a symmetrical matrix of 1s and -1s which verifies the orthogonality property:

$$H_{2^m} H_{2^m}^T = 2^m I_{2^m} \tag{A.14}$$

Matrices of these characteristics (which are not necessarily symmetrical but must be composed of 1s and -1s) are called Hadamard matrices.

Proof. All the properties can be proven by induction on m. □

Given a real vector $x \in \mathbb{R}^{2^m}$ of length 2^m, we say that its Walsh-Hadamard transform is the matrix-vector product $H_{2^m} x$. This product can be computed efficiently using the following result:

Proposition A.8. *The Hadamard-Sylvester matrix H_{2^m} can be obtained by multiplying the matrices $H_m^i = I_{2^{m-i}} \otimes H_2 \otimes H_{2^{i-1}}$. In other words, $H_{2^m} = \prod_{i=1}^m H_m^i$.*

Proof. This result is deduced from the mixed-product property $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$.

$$\prod_{i=1}^{m} H_m^i = \prod_{i=1}^{m} (I_{2^{m-i}} \otimes H_2 \otimes H_{2^{i-1}}) = \prod_{i=1}^{m} \left(\bigotimes_{j=1}^{m-i} I_2 \otimes H_2 \otimes \bigotimes_{j=m-i+1}^{m} I_2 \right) = \bigotimes_{j=1}^{m} H_2 = H_{2^m}$$

□

This means that the Walsh-Hadamard Transform of a given vector of length 2^m can be computed directly on the input registers with exactly $m2^m$ additions and substractions using the following algorithm, which succesively multiplies the vector by the matrices H_m^i:

Algorithm 10: The Fast Walsh-Hadamard Transform

```
Input: A binary vector x of length 2^m.
Output: A binary vector y of length 2^m.
y ← x;
for i ← 1 to m do
    for j ← 0 to 2^{m-i} do
        for k ← 0 to 2^{i-1} do
            tmp ← y[j2^i + 2^{i-1} + k];
            y[j2^i + 2^{i-1} + k] ← y[j2^i + k] − y[j2^i + 2^{i-1} + k];
            y[j2^i + k] ← tmp + y[j2^i + k];
        end
    end
end
return x
```

The Walsh-Hadamard Transform is a particular case of the Generalised Discrete Fourier Transform (in particular, the one corresponding to the abelian group $\mathbb{Z}_2^m$) and the Fast Walsh-Hadamard Transform is a particular case of the Generalised Fast Fourier Transform. The spectral analysis of boolean functions using the Walsh Transform is a very interesting topic in cryptography (see [AES02]) and has a very strong relationship with linear cryptanalysis.

Probability distributions. Finally, table 5 contains a condensed description of the probability distributions which are used in the statistical models for linear cryptanalysis and their properties, as well as the notations that we use for them.

Bernoulli distribution $Be(p)$

A discrete distribution which takes value 1 with probability p and 0 otherwise.

Probability function: $Pr(X = 1) = p, \quad Pr(X = 0) = 1 - p$

Expected value: p **Variance:** $p(p - 1)$

Binomial distribution $\mathcal{B}(p, n)$

The sum of n independent Bernoulli random variables $Be(p)$.

Probability function: $Pr(X = k) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1 - p)^{n-k}, \quad k = 0, \ldots, n$

Expected value: np **Variance:** $np(p - 1)$

For a large enough value of n and a value of p that's not too close to 0 or 1, this distribution can be approximated by the normal distribution of the same expected value and variance.

Hypergeometric distribution $\mathcal{HG}(K, N, n)$

The distribution of the number of successes among $n \leq N$ draws without replacement of a population of size N containing $K \leq N$ marked elements.

Probability function: $Pr(X = k) = \dfrac{\left(\begin{array}{c} N \\ k \end{array} \right) \left(\begin{array}{c} N - K \\ n - k \end{array} \right)}{\left(\begin{array}{c} n \\ k \end{array} \right)}, \quad k = 0, \ldots, n$

Expected value: $n\frac{K}{N}$ **Variance:** $n\frac{K}{N}\frac{N-K}{N}\frac{N-n}{N-1}$

This distribution can be approximated by the normal distribution of the same expected value and variance, although the resulting error depends on the parameters.

Normal distribution $\mathcal{N}(\mu, \sigma^2)$

Probability density function: $\varphi_{\mu,\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad \varphi = \varphi_{0,1}$

Cumulative distribution function: $\Phi_{\mu,\sigma^2}(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad \Phi = \Phi_{0,1}$

Expected value: μ **Variance:** σ^2

Folded normal distribution $\mathcal{FN}(\mu, \sigma^2)$

The distribution of the absolute value of normal random variable of the same parameters.

Probability density function: $\frac{1}{\sqrt{2\pi\sigma^2}} \left(e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} + e^{-\frac{1}{2}\left(\frac{x+\mu}{\sigma}\right)^2} \right)$

Non-central chi-square distribution $\chi^2_m(\lambda), \quad \chi^2_m = \chi^2_m(0)$

The distribution of the sum of m independent variables $\mathcal{N}(\mu_i, 1)$ verifying $\lambda = \sum_{i=1}^{m} \mu_i^2$. The distribution is said to have m degrees of freedom and non-centrality parameter λ.

Cumulative distribution function: $\Psi_{m,\lambda}(x), \quad \Psi_m = \Psi_{m,0}$

Expected value: $m + \lambda$ **Variance:** $2(m + 2\lambda)$

For a large enough value of m, this distribution can be approximated by the normal distribution of the same expected value and variance because of the Central Limit Theorem.

Table 5: Definitions, notations and properties of some probability distributions.

B Linear cryptanalysis revisited

This appendix acts as a complement to section 2.

B.1 Proofs of the results of subsections 2.1 and 2.2

We begin by proving Matsui's piling up lemma, which is a direct consequence of the following general result on the addition of independent binary random variables:

Proposition B.1. *If $X_1, \ldots, X_r$ are independent Bernoulli random variables whose value is 0 with probability $1/2 + \varepsilon_i$ (and 1 with probability $1/2 - \varepsilon_i$), then*

$$Pr(X_1 \oplus X_2 \oplus \ldots \oplus X_r = 0) = \frac{1}{2} + 2^{r-1} \prod_{i=1}^{r} \varepsilon_i \tag{B.1}$$

Proof. The proof will proceed by induction on r.

If $r = 2$, we have $X_1 \oplus X_2 = 0$ if and only if $X_1 = X_2 = 0$, which happens with probability $(1/2 + \varepsilon_1)(1/2 + \varepsilon_2)$, or $X_1 = X_2 = 1$, which happens with probability $(1/2 - \varepsilon_1)(1/2 - \varepsilon_2)$. The sum of these two probabilities is $1/2 + 2\varepsilon_1\varepsilon_2$.

For greater values of r, from the formula for $r - 1$ variables we deduce

$$\begin{aligned} Pr(X_1 \oplus \ldots \oplus X_r = 0) &= Pr(X_1 = 0, X_2 \oplus \ldots \oplus X_r = 0) + Pr(X_1 = 1, X_2 \oplus \ldots \oplus X_r = 1) \\ &= \left(\frac{1}{2} + \varepsilon_1\right)\left(\frac{1}{2} + 2^{r-2}\prod_{i=2}^{r}\varepsilon_i\right) + \left(\frac{1}{2} - \varepsilon_1\right)\left(\frac{1}{2} - 2^{r-2}\prod_{i=2}^{r}\varepsilon_i\right) = \frac{1}{2} + 2^{r-1}\prod_{i=1}^{r}\varepsilon_i \end{aligned}$$

which is the expression we wanted to obtain. □

We now proceed to the proof of proposition 2.4.

Proposition (2.4)**.** *Under the right-key equivalence hypothesis, the probability of success of Matsui's Algorithm 1 is approximately*

$$P_S \simeq \Phi\left(2\sqrt{N}|\varepsilon|\right) = \Phi\left(\sqrt{Nc^2}\right) = \int_{-\infty}^{\sqrt{Nc^2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\, dx \tag{B.2}$$

Proof. Without loss of generality, we can suppose that $\gamma(K) = 0$ and $\varepsilon > 0$. The right-key equivalence hypothesis implies that $Pr_x(\alpha \cdot x \oplus \beta \cdot E_K(x) = 0) = 1/2 + \varepsilon$.

The number T is thus a random variable which follows a binomial distribution $\mathcal{B}(1/2+\varepsilon, N)$ and can be approximated by the normal $\mathcal{N}(N(1/2+\varepsilon), N/4)$. The probability of success is the probability that $T > N/2$. We conclude by expressing T as a linear transformation of a standard normal distribution. □

We can also prove the following proposition, which is an extended version of corollary 2.6:

Proposition B.2 (Lemma 5 in [Ma94a])**.** *Under the right-key equivalence and the wrong-key randomisation hypotheses, he probability of success of Matsui's algorithm 2 is approximately*

$$P_S \simeq \int_{-2\sqrt{N}|\varepsilon|}^{\infty} \left(\int_{-x-2\sqrt{N}|\varepsilon|}^{x+2\sqrt{N}|\varepsilon|} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2}\, dy \right)^{2^{|k|}-1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\, dx \tag{B.3}$$

Proof. Without loss of generality, we can suppose that $\varepsilon > 0$ and $\gamma(K) = 0$. From the statistical assumptions about the linear approximation, we deduce that the distribution of T_k can be approximated by $\mathcal{N}(N/2 + N\varepsilon, N/4)$ if k is the right subkey, while for any wrong subkey $T_{\tilde{k}}$ follows the distribution

$\mathcal{N}(N/2, N/4)$, and they are all independent. The probability of success P_S is the probability that the counter T_k for the right key is larger than $N/2$ and $|T_k - N/2|$ is larger than all the $|T_{\tilde{k}} - N/2|$.

$$\begin{aligned}
P_S =& Pr_x\left(T_k > N/2,\ |T_k - N/2| > |T_{\tilde{k}} - N/2| \text{ for all } \tilde{k}\right)\\
=& Pr_x\left(T_k - N/2 > 0,\ -(T_k - N/2) < T_{\tilde{k}} - N/2 < T_k - N/2 \text{ for all } \tilde{k}\right)\\
=& \int_0^\infty \left(\int_{-x}^{x} \sqrt{\frac{2}{\pi N}} e^{-\frac{2}{N}y^2} dy\right)^{2^{|k|}-1} \sqrt{\frac{2}{\pi N}} e^{-\frac{2}{N}(x-N\varepsilon)^2} dx
\end{aligned}$$

The desired expression is obtained after a change of variables in the previous integral. □

This is Nyberg's proof of theorem 2.7, which is a version of Parseval's identity for the Walsh Transform:

Theorem (2.7)**.** *Let $\nu : \alpha \cdot x \oplus \beta \cdot y$ be a linear approximation of a key-alternating block cipher E for which all the round subkeys are independent (this is often called a long-key cipher, and it implies $\kappa = n(r+1)$). Then, for any key mask $\gamma : \mathbb{F}_2^\kappa \to \mathbb{F}_2$:*

$$\begin{aligned}
&\frac{1}{2^\kappa} \sum_{K \in \mathbb{F}_2^\kappa} \left(Pr_x\left(\alpha \cdot x \oplus \beta \cdot E_K(x) = 0 | K\right) - Pr_x\left(\alpha \cdot x \oplus \beta \cdot E_K(x) = 1 | K\right)\right)^2\\
=& \sum_{\gamma_0,\ldots,\gamma_r \in \mathbb{F}_2^n} \left(Pr_{x,K}\left(\alpha \cdot x \oplus \beta \cdot E_K(x) \oplus \gamma_0 \cdot K_0 \oplus \ldots \oplus \gamma_r \cdot K_r = 0\right)\right.\\
&\qquad \left. - Pr_{x,K}\left(\alpha \cdot x \oplus \beta \cdot E_K(x) \oplus \gamma_0 \cdot K_0 \oplus \ldots \oplus \gamma_r \cdot K_r = 1\right)\right)^2
\end{aligned} \tag{B.4}$$

Proof. The second member of the equality can be reweitten as follows:

$$\begin{aligned}
&\sum_{\boldsymbol{\gamma} \in \mathbb{F}_2^{n(r+1)}} \left(\frac{1}{2^{n+\kappa}} \sum_{x \in \mathbb{F}_2^n} \sum_{K \in \mathbb{F}_2^\kappa} (-1)^{\alpha \cdot x \oplus \beta \cdot E_K(x) \oplus \gamma_0 \cdot K_0 \oplus \ldots \oplus \gamma_r \cdot K_r}\right)^2\\
&= \frac{1}{2^{2n+2\kappa}} \sum_{\boldsymbol{\gamma} \in \mathbb{F}_2^{n(r+1)}} \sum_{x,x' \in \mathbb{F}_2^n} \sum_{K,K' \in \mathbb{F}_2^\kappa} (-1)^{\alpha \cdot x \oplus \alpha \cdot x' \oplus \beta \cdot E_K(x) \oplus \beta \cdot E_{K'}(x') \oplus \gamma_0 \cdot K_0 \oplus \ldots \oplus \gamma_r \cdot K_r \oplus \gamma_0 \cdot K'_0 \oplus \ldots \oplus \gamma_r \cdot K'_r}\\
&= \frac{1}{2^{2n+2\kappa}} \sum_{x,x' \in \mathbb{F}_2^n} \sum_{K,K' \in \mathbb{F}_2^\kappa} (-1)^{\alpha \cdot x \oplus \alpha \cdot x' \oplus \beta \cdot E_K(x) \oplus \beta \cdot E_{K'}(x')} \left(\sum_{\boldsymbol{\gamma} \in \mathbb{F}_2^{n(r+1)}} (-1)^{\gamma_0 \cdot K_0 \oplus \ldots \oplus \gamma_r \cdot K_r \oplus \gamma_0 \cdot K'_0 \oplus \ldots \oplus \gamma_r \cdot K'_r}\right)
\end{aligned}$$

Because E is a long-key cipher, we have

$$K = K' \iff (K_0, \ldots, K_r) = (K'_0, \ldots, K'_r) \iff (K_0 \oplus K'_0, ..., K_r \oplus K'_r) = \mathbf{0}$$

Therefore, by using proposition A.3, we have

$$\sum_{\boldsymbol{\gamma} \in \mathbb{F}_2^{n(r+1)}} (-1)^{\gamma_0 \cdot K_0 \oplus \ldots \oplus \gamma_r \cdot K_r \oplus \gamma_0 \cdot K'_0 \oplus \ldots \oplus \gamma_r \cdot K'_r} = \sum_{\boldsymbol{\gamma} \in \mathbb{F}_2^{n(r+1)}} (-1)^{(\gamma_0,\ldots,\gamma_r) \cdot (K_0 \oplus K'_0, \ldots, K_r \oplus K'_r)} = \begin{cases} 0 & \text{if } K \neq K' \\ 2^\kappa & \text{if } K = K' \end{cases}$$

We can substitute this in the previous expression:

$$\frac{1}{2^{2n+\kappa}} \sum_{K \in \mathbb{F}_2^\kappa} \sum_{x,x' \in \mathbb{F}_2^n} (-1)^{\alpha \cdot (x \oplus x') \oplus \beta \cdot (E_K(x) \oplus E_K(x'))} = \frac{1}{2^\kappa} \sum_{K \in \mathbb{F}_2^\kappa} \left(\frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{\alpha \cdot x \oplus \beta \cdot E_K(x)}\right)^2$$

This concludes the proof. □

In definition 2.8, we can say $\gamma_0 = \alpha, \gamma_r = \beta$ because if, for example, we had $\gamma_0 \neq \alpha$, then $\alpha \cdot x \oplus \gamma_0 \cdot K_0$ is 0 with probability exactly 1/2, so no matter what the rest of the linear trail is, its correlation is 0 because of the piling-up lemma. We will now prove theorem 2.10:

Theorem (2.10)**.** *Let E be a key-alternating cipher of r rounds with internal permutation F, and let $\hat{C}^{(F)}$ be the element-wise square of the correlation matrix of the round function. Then the ELP of a linear approximation of E with input mask α and output mask β is*

$$ELP(\alpha, \beta) = \left(\left(\hat{C}^{(F)}\right)^r\right)_{\alpha\beta} \tag{B.5}$$

Proof. The equality is deduced from the definition of the ELP and the piling-up lemma:

$$\begin{aligned} ELP(\alpha,\beta) &= \sum_{\substack{\boldsymbol{\gamma}\in\mathbb{F}_2^{n(r+1)} \\ \gamma_0=\alpha,\gamma_r=\beta}} \left(Pr_{x,K_0,\ldots,K_r}(\nu_{\boldsymbol{\gamma}}=0) - Pr_{x,K_0,\ldots,K_r}(\nu_{\boldsymbol{\gamma}}=1)\right)^2 \\ &= \sum_{\substack{\boldsymbol{\gamma}\in\mathbb{F}_2^{n(r+1)} \\ \gamma_0=\alpha,\gamma_r=\beta}} \left(\prod_{i=1}^{r} \left(Pr_x(\gamma_{i-1}\cdot x \oplus \gamma_i \cdot F(x) = 0) - Pr_x(\gamma_{i-1}\cdot x \oplus \gamma_i \cdot F(x) = 1)\right)\right)^2 \\ &= \sum_{\substack{\boldsymbol{\gamma}\in\mathbb{F}_2^{n(r+1)} \\ \gamma_0=\alpha,\gamma_r=\beta}} \left(\prod_{i=1}^{r} \left(C^{(F)}_{\gamma_{i-1}\gamma_i}\right)\right)^2 = \sum_{\substack{\boldsymbol{\gamma}\in\mathbb{F}_2^{n(r+1)} \\ \gamma_0=\alpha,\gamma_r=\beta}} \prod_{i=1}^{r} \left(C^{(F)}_{\gamma_{i-1}\gamma_i}\right)^2 = \left(\left(\hat{C}^{(F)}\right)^r\right)_{\alpha\beta} \end{aligned}$$

It is also true that the correlation matrix for the composition of two boolean functions is the matrix product of their respective correlation matrices, and the proof can be found in chapter 7 of [AES02] or deduced from the piling-up lemma. □

B.2 Multidimensional linear cryptanalysis

We now provide a very short description of a multidimensional version of Algorithm 2 using the χ^2 statistic as introduced in [HCN08],[HCN09],[HCN19] (multidimensional attacks using the log-likelihood ratio as well as Algorithm 1-type attacks are also introduced in these publications).

Consider a set of $M = 2^m - 1$ approximations whose masks form an m-dimensional vector subspace of $\mathbb{F}_2^n \times \mathbb{F}_2^n$ (ignoring the approximation with both masks equal to zero). There are m linearly independent linear approximations ν_j given by $\alpha_j, \beta_j \in \mathbb{F}_2^n,\ j = 0, \ldots, m-1$ which span the complete set of approximations. These m approximations follow a joint discrete distribution in $\mathbb{F}_2^m$. In other words, for each K we have the distribution

$$\boldsymbol{\eta}(K) = (\eta_0(K), \ldots, \eta_{2^m-1}(K)), \text{ where } \eta_i(K) = Pr_x\left((\nu_{m-1}, \ldots, \nu_0) = i \mid K\right) \tag{B.6}$$

using the identification between $\mathbb{F}_2^m$ and the integers from 0 to $2^m - 1$.

Hermelin et al. proposed a variation of Algorithm 2 based on a new wrong-key randomisation hypothesis. They assumed that for any wrong guess of the partial subkey k, then $\boldsymbol{\eta}(K)$ will be the uniform distribution over $\mathbb{F}_2^m$. Given k, this distribution can be tested for uniformity using the χ^2 test of fit.

The attacker begins by computing the $2^m \cdot 2^{|k|}$ statistics

$$\xi_k^i = \#\left\{(x,y) \in \mathcal{D} : (\alpha_{m-1}\cdot x \oplus f_{m-1}(y|_\chi \oplus k), \ldots, \alpha_0 \cdot x \oplus f_0(y|_\chi \oplus k)) = i\right\} \tag{B.7}$$

These are then combined into the χ^2 statistic for each key:

$$Q_k = \frac{1}{N2^{-m}} \sum_{i=0}^{2^m-1} \left(\xi_k^i - N2^{-m}\right)^2 \tag{B.8}$$

The attacker then chooses the subkey guesses k corresponding to the largest values of Q_k.

Algorithm 11: Multidimensional Algorithm 2 using the χ^2 statistic

Input: A collection $\mathcal{D} = \{(x, y = E_K(x))\}$ of N plaintext-ciphertext pairs (possibly on-the-fly).
Output: A probable guess of k.
for $i \leftarrow 1$ **to** $2^m - 1$ **do** $\boldsymbol{\xi}^i \leftarrow \mathbf{0}$;
forall $(x, y) \in \mathcal{D}$ **do** `// Compute the counters`
  for $k \leftarrow 0$ **to** $2^{|k|} - 1$ **do**
    $\xi_k^{(\alpha_{m-1}\cdot x \oplus f_{m-1}(y|_\chi \oplus k),\ldots,\alpha_0 \cdot x \oplus f_0(y|_\chi \oplus k))} \leftarrow \xi_k^{(\alpha_{m-1}\cdot x \oplus f_{m-1}(y|_\chi \oplus k),\ldots,\alpha_0 \cdot x \oplus f_0(y|_\chi \oplus k))} + 1$;
  end
end
for $k \leftarrow 0$ **to** $2^{|k|} - 1$ **do** $Q_k \leftarrow \frac{1}{N2^{-m}} \sum_{i=0}^{2^m-1} \left(\xi_k^i - N2^{-m}\right)^2$; `// Compute the` χ^2 `statistic`
return $argmax_k(Q_k)$;

The probability of success depends on the capacity of the set of approximations, and in this case there is no need to suppose statistical independence:

Proposition B.3 ([HCN09])**.** *The probability of success of a multidimensional linear attack using $M = 2^m - 1$ linear approximations depends on the number of available plaintexts N, the number of approximations and the* capacity *of the set of linear approximations, which is*

$$C(K) = \sum_{i=1}^{2^m-1} (c_i(K))^2 = \sum_{i=0}^{2^m-1} \frac{(\eta_i(K) - 2^{-m})^2}{2^{-m}}, \quad C = Exp_K\left(C(K)\right) = \sum_{i=1}^{2^m-1} ELP(\alpha_i, \beta_i) \quad \text{(B.9)}$$

B.3 The statistical model of subsection 2.4

B.3.1 Statistical attacks as a hypothesis testing problem

Selçuk introduced the notion of advantage that was discussed in subsection 2.4 in [Se08], and also provided a way of estimating the probability of success of this type of attack by using a result on order statistics. Subsequent reformulations such as the one found in [BW12] have reframed the problem as a hypothesis testing scenario, which leads to the same estimate. Here we will discuss the latter approach.

Given a guess for the partial subkey k, we want to test whether it corresponds to the actual secret key K (alternative hypothesis H_1) or whether it is a wrong guess (null hypothesis H_0). We are interested in the significance level α and the power $1-\beta$ of the test (not to be confused with the input and output masks of the linear approximations). The significance level α is the probability that the null hypothesis is rejected erroneously (probability of false alarm), in this case, the probability that a wrong guess is marked as a candidate for the search phase. Since we expect $\alpha 2^{|k|}$ wrong key guesses to be marked as candidates, we deduce that $\alpha 2^{|k|} \simeq 2^{|k|-a}$. This means that, if we want to achieve an advantage of a, we should use a test with $\alpha = 2^{-a}$. On the other hand, the power $1-\beta$ is the probability that the null hypothesis is rejected when the alternative hypothesis is true (β is the probability of non-detection), in other words, the probability that the right guess for the subkey is marked as a candidate for the search phase. This means that the success probability of the attack is $P_S = 1 - \beta$.

To distinguish between both hypotheses, we use a test statistic X_k. In Matsui's Algorithm 2, this statistic is $|T_k - N/2|$ while in multiple or multidimensional attacks it's Q_k. Let us suppose that, for the right guess of k, this statistic has a cumulative distribution function F_R. Analogously, we will suppose that for any wrong guess, the distribution function is F_W. We will consider a threshold Θ:

$$F_W^{-1}(1-\alpha) \leq \Theta \leq F_R^{-1}(\beta) \quad \text{(B.10)}$$

We reject the null hypothesis (keep the subkey guess as a candidate) if $X_k > \Theta$ and accept it (discard the subkey guess) if $X_k < \Theta$. Usually, we will choose the significance level $\alpha = 2^{-a}$ according to the desired advantage and then take $\Theta = F_W^{-1}(1-\alpha)$. The power of the test will then be $\beta = F_R(\Theta)$. We deduce the following result about the probability of success:

Theorem (2.13)**.** *Under the previous hypothesis testing model for a statistical attack, the success probability for a given desired advantage a is*

$$P_S = 1 - \beta = 1 - F_R\left(F_W^{-1}(1-\alpha)\right) = 1 - F_R\left(F_W^{-1}(1-2^{-a})\right) \tag{B.11}$$

If the right-key distribution can be approximated by a normal distribution $\mathcal{N}(\mu_R, \sigma_R^2)$, then

$$P_S \simeq \Phi\left(\frac{\mu_R - F_W^{-1}(1-2^{-a})}{\sigma_R}\right) \tag{B.12}$$

If the wrong-key distribution can also be approximated by a normal distribution $\mathcal{N}(\mu_W, \sigma_W^2)$, then

$$P_S \approx \Phi\left(\frac{\mu_R - \mu_W - \sigma_W\Phi^{-1}(1-2^{-a})}{\sigma_R}\right) \tag{B.13}$$

Since $1-2^{-a}$ is close to 1, the normal approximation of F_W^{-1} might induce a non-negligible error. For this reason we only recommend using this approximation when there is no other option.

Corollary B.4. ([Se08]) *The probability of success of Algorithm 2 with target advantage a is*

$$P_S \simeq \Phi\left(2\sqrt{N}|\varepsilon| - \Phi^{-1}(1-2^{-a-1})\right) \tag{B.14}$$

Proof. We will suppose that $\varepsilon > 0$ and $\gamma(K) = 0$. From the right-key equivalence and wrong-key randomisation hypotheses, we know that $T_k - N/2$ follows the normal distribution $\mathcal{N}(N\varepsilon, N/4)$ if k is the right guess, and $\mathcal{N}(0, N/4)$ if it is a wrong guess. Therefore $|T_k - N/2|$ follows the folded normal distribution $\mathcal{FN}(N\varepsilon, N/4)$ for the right guess and $\mathcal{FN}(0, N/4)$ for any wrong guess. If X is a random variable with distribution $\mathcal{N}(0, N/4)$, we have

$$F_W(x) = Pr(|X| < x) = 1 - Pr(X > x) - Pr(X < -x) = 1 - 2\left(1 - \Phi\left(\frac{x}{\sqrt{N/4}}\right)\right) = 2\Phi\left(\frac{2x}{\sqrt{N}}\right) - 1$$

So $F_W^{-1}(1-2^{-a}) = \frac{\sqrt{N}}{2}\Phi^{-1}(1-2^{-a-1})$. We finish by substituting this in equation 2.22/B.12. □

B.3.2 Distribution of the multiple linear cryptanalysis statistic

We will replicate the reasoning by Blondeau and Nyberg in [BN17] to deduce the distribution of the multiple linear cryptanalysis statistics to plug into Selçuk's formula. We begin with updated versions of the right-key and wrong-key hipotheses. As usual, let $\nu_i : \alpha_i \cdot x \oplus \beta_i \cdot \hat{y}$ be M linear approximations of the first $r-1$ rounds of the cipher, so that $\beta_i \cdot F^{-1}(y \oplus K_r)$ can be substituted by $f_i(y|_\chi \oplus k)$.

Assumption B.5. (Right-key randomisation) *If K is uniformly distributed over $\mathbb{F}_2^\kappa$, then*

$$c_i(K) = Pr_x\left(\nu_i = 0|K\right) - Pr_x\left(\nu_i = 1|K\right) \tag{B.15}$$

follow the normal distribution $\mathcal{N}(0, ELP(\alpha_i, \beta_i))$ and are statistically independent.

Since the ELP is the expected value of the square of the correlation, it is clear that it must be equal to the variance if the expected value of the correlation is zero.

Assumption B.6. (Wrong-key randomisation) *The wrong-key empirical correlations ($\tilde{k} \neq k$)*

$$c_i^W(K, \tilde{k}) = Pr_x\left(\alpha_i \cdot x \oplus f_i(E_K(x)|_\chi \oplus \tilde{k}) = 0|K, \tilde{k}\right) - Pr_x\left(\alpha_i \cdot x \oplus f_i(E_K(x)|_\chi \oplus \tilde{k}) = 1|K, \tilde{k}\right) \tag{B.16}$$

follow the normal distribution $\mathcal{N}(0, 2^{-n})$ and are statistically independent.

This assumption can be justified if we imagine that the value of $\alpha_i \cdot x \oplus f_i(E_K(x)|_\chi \oplus \tilde{k}) = 0$ is random.

$$\#\left\{x : \alpha_i \cdot x \oplus f_i(E_K(x)|_\chi \oplus \tilde{k}) = 0\right\} = 2^n Pr_x\left(\alpha_i \cdot x \oplus f_i(E_K(x)|_\chi \oplus \tilde{k}) = 0\right) = 2^{n-1}(c_i^W(K, \hat{k}) + 1)$$

is a random variable which follows the binomial distribution $\mathcal{B}(1/2, 2^n)$, and thus has expected value 2^{n-1} and variance 2^{n-2}. This means that the expected value of $c_i^W(K, \hat{k})$ is 0 and its variance is 2^{-n}.

Distribution of the capacity. To calculate the estimated linear potentials the M linear approximations, we identify a set of "dominant" trails, as discussed in 2.2.1. In other words, we consider a set $\mathcal{S}$ of key masks γ so that the approximations $\nu_i \oplus \gamma \cdot \widehat{K}$ are highly biased, while the rest have a much smaller correlation, and are modeled as random noise.

Theorem B.7 (Theorem 3 in [BN17])**.** *For each approximation $\nu_i : \alpha_i \cdot x \oplus \beta_i \cdot \hat{y}$, we can estimate the linear potential by choosing a set $\mathcal{S}$ containing the linear trails with the largest correlation contribution, and using the formula:*

$$Var_K\left(c_i(K)\right) = ELP\left(\alpha_i, \beta_i\right) \simeq \sum_{\gamma \in \mathcal{S}} \left(Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 0) - Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 1)\right)^2 + 2^{-n} \quad \text{(B.17)}$$

where $\widehat{K} = (K_0, \ldots, K_r) \in \mathbb{F}_2^{n(r+1)}$ is uniformly distributed.

The next step towards an estimation of the success probability in the case of multiple linear cryptanalysis is finding the distribution of the capacity $C(K)$ for the set of approximations.

Theorem B.8 (Theorems 4,5 in [BN17])**.** *The expected value of the capacity of a set of M linear approximations $\nu_i : \alpha_i \cdot x \oplus \beta_i \cdot \hat{y}$ over the keyspace is*

$$Exp_K\left(C(K)\right) = \sum_{i=1}^{M} ELP(\alpha_i, \beta_i) \simeq \sum_{i=1}^{M} \sum_{\gamma \in \mathcal{S}} \left(Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 0) - Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 1)\right)^2 + M2^{-n} \quad \text{(B.18)}$$

Meanwhile, the variance of $C(K)$ is

$$\begin{aligned} Var_K(C(K)) &= \sum_{i=1}^{M} 2ELP(\alpha_i, \beta_i)^2 \\ &\simeq 2\sum_{i=1}^{M} \left(\sum_{\gamma \in \mathcal{S}} \left(Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 0) - Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 1)\right)^2 + 2^{-n} \right)^2 \\ &= 2\sum_{i=1}^{M} \left(\sum_{\gamma \in \mathcal{S}} \left(Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 0) - Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 1)\right)^2 \right)^2 \\ &\quad + 2^{2-n} \sum_{i=1}^{M} \sum_{\gamma \in \mathcal{S}} \left(Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 0) - Pr_{x,\widehat{K}}(\nu_i \oplus \gamma \cdot \widehat{K} = 1)\right)^2 + M2^{1-2n} \end{aligned} \quad \text{(B.19)}$$

Proof. These formulas are deduced from the assumption of the independence of the correlations and our previous estimation for the linear potential. In the case of the expected value, we have

$$Exp_K\left(C(K)\right) = \sum_{i=1}^{M} Exp_K\left(\left(c_i(K)\right)^2\right) = \sum_{i=1}^{M} Var_K\left(c_i(K)\right) = \sum_{i=1}^{M} ELP(\alpha_i, \beta_i)$$

For the variance, we have

$$Var_K\left(C(K)\right) = Var_K\left(\sum_{i=1}^{M} \left(c_i(K)\right)^2\right) = \sum_{i=1}^{M} Var_K\left(\left(c_i(K)\right)^2\right)$$

Since we assumed $\sqrt{1/ELP(\alpha_i, \beta_i)}c_i(K)$ follows a standard normal distribution, then its square $1/ELP(\alpha_i, \beta_i)\left(c_i(K)\right)^2$ should follow a χ^2 distribution with one degree of freedom. This distribution has a variance of 2, so we deduce that $Var_K\left(\left(c_i(K)\right)^2\right) = 2ELP(\alpha_i, \beta_i)^2$. □

Distribution of the right key and wrong key statistics. The final component of the model is the computation of the distribution of the right and wrong key statistics over the key K and the data $\mathcal{D}$. Two different scenarios can be considered: the classical known plaintext attack (KP), where the plaintexts x are randomly drawn from $\mathbb{F}_2^n$ with replacement (so there can be repeated plaintext-ciphertext pairs in the data), or the distinct known plaintext attack (DKP), where the plaintexts are drawn without replacement and there are no repetitions. The difference between these two scenarios will be accounted for in the success probability estimations by changing the value of the constant

$$B = \begin{cases} 1 & \text{for KP} \\ \frac{2^n - N}{2^n - 1} & \text{for DKP} \end{cases} \tag{B.20}$$

Theorem (2.14)**.** *In multiple and multidimensional linear cryptanalysis, the sttatistic Q_k approximately follows a normal distribution whose mean and variance are the following:*

$$\begin{array}{l} Q_k \sim \mathcal{N}(\mu_R, \sigma_R), \text{ where} \\ \left\{ \begin{array}{rcccl} \mu_R & = & Exp_{\mathcal{D},K}(Q_k) & = & BM + NExp_K\left(C(K)\right) \\ \sigma_R^2 & = & Var_{\mathcal{D},K}(Q_k) & = & 2B^2M + 4BNExp_K\left(C(K)\right) + N^2Var_K\left(C(K)\right) \end{array} \right. \end{array} \tag{B.21}$$

Meanwhile, if the key guess $\tilde{k} \neq k$ is different from the right one, a multiple of the wrong key statistic follows a χ^2 distribution with M degrees of freedom:

$$\frac{1}{B + N2^{-n}} Q_{\tilde{k}} \sim \chi^2_M, \text{ so } \left\{ \begin{array}{rcccl} \mu_W & = & Exp_{\mathcal{D},K}(Q_{\tilde{k}}) & = & BM + NM2^{-n} \\ \sigma_W^2 & = & Var_{\mathcal{D},K}(Q_{\tilde{k}}) & = & 2M(B + N2^{-n})^2 \end{array} \right. \tag{B.22}$$

Proof. Here we will only consider the case of multiple linear cryptanalysis, although the result is also true for multidimensional attacks.

We will begin by deducing the distribution of the right-key statistic. We initially consider that the key K is fixed. This means that $\alpha_i \cdot x \oplus f_i(y|_\chi \oplus k) \oplus 1$ is drawn from the Bernoulli distribution with parameter $(1 + c_i(K))/2$. Therefore

$$\begin{cases} \#\left\{(x,y) \in \mathcal{D} : \alpha_i \cdot x \oplus f_i(y|_\chi \oplus k) = 0\right\} \sim \mathcal{B}\left(\frac{1+c_i(K)}{2}, N\right) & \text{for KP} \\ \#\left\{(x,y) \in \mathcal{D} : \alpha_i \cdot x \oplus f_i(y|_\chi \oplus k) = 0\right\} \sim \mathcal{HG}\left(\frac{1+c_i(K)}{2}2^n, 2^n, N\right) & \text{for DKP} \end{cases}$$

From this we deduce that $q_k^i \sim \mathcal{N}(Nc_i(K), NB)$. We then obtain, for a fixed K,

$$\frac{1}{B}Q_k \sim \chi^2_M\left(NB^{-1}C(K)\right), \quad \left\{ \begin{array}{rcl} Exp_{\mathcal{D}}\left(Q_k\right) & = & B(M + NB^{-1}C(K)) = BM + NC(K) \\ Var_{\mathcal{D}}\left(Q_k\right) & = & B^2(2M + 4NB^{-1}C(K)) = 2B^2M + 4BNC(K) \end{array} \right.$$

Therefore, assuming a normal approximation of the χ^2 distribution,

$$\begin{aligned} Exp_{\mathcal{D},K}\left(Q_k\right) &= Exp_K\left(Exp_{\mathcal{D}}\left(Q_k\right)\right) = BM + NExp_K\left(C(K)\right) \\ Var_{\mathcal{D},K}\left(Q_k\right) &= Exp_K\left(Var_{\mathcal{D}}\left(Q_k\right)\right) + Var_K\left(Exp_{\mathcal{D}}\left(Q_k\right)\right) \\ &= 2B^2M + 4BNExp_K\left(C(K)\right) + N^2Var_K\left(C(K)\right) \end{aligned}$$

We now proceed to the case of the wrong key statistic. For a fixed K, we now have

$$\begin{cases} \#\left\{(x,y) \in \mathcal{D} : \alpha_i \cdot x \oplus f_i(y|_\chi \oplus \tilde{k}) = 0\right\} \sim \mathcal{B}\left(\frac{1+c_i^W(K,\tilde{k})}{2}, N\right) & \text{for KP} \\ \#\left\{(x,y) \in \mathcal{D} : \alpha_i \cdot x \oplus f_i(y|_\chi \oplus \tilde{k}) = 0\right\} \sim \mathcal{HG}\left(\frac{1+c_i^W(K,\tilde{k})}{2}2^n, 2^n, N\right) & \text{for DKP} \end{cases}$$

Therefore $q_{\tilde{k}}^i \sim \mathcal{N}(Nc_i^W(K,\tilde{k}), NB)$ for a fixed key K and $q_{\tilde{k}}^i \sim \mathcal{N}(0, NB + N^2 2^{-n})$ when K and $\mathcal{D}$ are uniformly distributed. We thus deduce

$$\frac{1}{B + N2^{-n}} Q_{\tilde{k}} \sim \chi^2_M, \quad \left\{ \begin{array}{rcl} Exp_{\mathcal{D},K}\left(Q_k\right) & = & BM + NM2^{-n} \\ Var_{\mathcal{D},K}\left(Q_k\right) & = & 2M(B + N2^{-n})^2 \end{array} \right.$$

which concludes the proof. □