Improving Key-Recovery in Linear Attacks: Application to 28-Round PRESENT

Antonio Flórez-Gutiérrez(✉) and María Naya-Plasencia(✉)

Inria, Paris, France
{antonio.florez_gutierrez,maria.naya_plasencia}@inria.fr

Abstract. Linear cryptanalysis is one of the most important tools in use for the security evaluation of symmetric primitives. Many improvements and refinements have been published since its introduction, and many applications on different ciphers have been found. Among these upgrades, Collard et al. proposed in 2007 an acceleration of the key-recovery part of Algorithm 2 for last-round attacks based on the FFT.

In this paper we present a generalized, matrix-based version of the previous algorithm which easily allows us to take into consideration an arbitrary number of key-recovery rounds. We also provide efficient variants that exploit the key-schedule relations and that can be combined with multiple linear attacks.

Using our algorithms we provide some new cryptanalysis on PRESENT, including, to the best of our knowledge, the first attack on 28 rounds.

Keywords: Linear cryptanalysis · FFT · Walsh Transform · Algorithm 2 · Key-recovery algorithm · PRESENT

1 Introduction

The foundation of the trust we have on symmetric primitives is based on the amount of cryptanalysis these primitives have received. The distance between the highest number of rounds that can be attacked and the full version is what determines the security margin of a cipher. For this quantity to have a meaning, the best reduced-round attacks within each known family of attacks should be accurately determined. In order to facilitate the application of known cryptanalysis families to new ciphers, generalizing the corresponding algorithms is an important task as it allows to: (1) accurately and semi-automatically determine the security margin, (2) find errors or suboptimal parts from previous attacks and (3) find new improvement ideas thanks to the clearer understanding of the attack. Several such examples exist, including impossible differential attacks [13,14], invariant attacks [5], and meet-in-the-middle attacks [15], to cite a few.

Linear cryptanalysis was introduced by Matsui in 1993 [31], and is one of the main symmetric cryptanalysis families. These statistical attacks, which in their most basic version exploit linear correlations between some bits of the plaintext, key and ciphertext, have benefited from many improvements and refinements over the years. For example, the introduction of linear hulls in [36] deepened the understanding of the underlying principles of linear attacks. There has also been a progressive development of techniques which exploit several linear approximations at the same time. In particular, multiple linear attacks were proposed in [7], and multidimensional attacks in [25,26]. Also important is the construction of statistical models which effectively predict the parameters of these attacks - in this respect, we highlight works such as [9,12,21,38].

In [31] Matsui proposed the partial key-recovery attack known as Algorithm 2 in the form of a last round-attack. The time complexity of this algorithm was greatly improved by the results from Collard et al. [18] using the FFT (Fast Fourier Transform). Despite the focus of many publications on improved ways of searching for linear distinguishers and estimating their capacity, little has been done regarding improvements of the key-recovery part, and the result from [18] and its application to some last-round multidimensional attacks in [35] are, to the best of our knowledge, the main known contributions in this direction.

Matsui introduced linear cryptanalysis for an attack on DES [34]. Linear cryptanalysis is a powerful tool that provides the best known attacks (like [12] or [24]) on several popular ciphers, such as PRESENT [11], NOEKEON [20], some variants of Simon [4] and most recently TRIFLE-BC [22].

In particular, the lightweight block cipher PRESENT [11], proposed in 2007 and made an ISO standard in 2012, is a popular cipher that has been the target of around 30 reduced-round cryptanalysis efforts, and some of the most successful are linear attacks. Out of its 31 total rounds, Ohkuma found a weak-key linear attack on 24 in [37]. Collard et al. found a statistical saturation attack on up to 26 rounds in 2009 [17]. Nakahara et al. proposed another 26-round linear attack in [33], and Cho described a multidimensional attack with a larger success probability in 2010 [16]. It wasn't until 2015 that 27 rounds were reached by Zheng et al. in [41]. A different 27-round attack was given by Bogdanov et al. in [12], but no attack on 28 rounds has been proposed.

Motivation of our work. The contrast between the amount of results devoted to the construction of effective linear distinguishers and the results regarding the key-recovery algorithms seemed quite surprising to us. In particular, the nice algorithm provided in [18] considers the simplified version in which only the final round is inverted and the only key to guess is directly xored to the ciphertext (though an application for a first and last round key-recovery attack is also sketched). In [35] a variant for multidimensional attacks with a fixed input mask is proposed. Many linear attacks and analysis don't consider this final round acceleration, for example in [8], or [33]. In [16], the author says *"The computational complexity may be further reduced by applying Fast Fourier Transform at the cost of the increased memory complexity"* without developing any further. In [27], the authors state *"It is not clear if the trick proposed by Collard, et al.* [18] *can be used in multiple dimensions"*. Of the ones that do, some only apply it as a black box in the simplified last-round case, like in [3], in [19], or in [2] where the authors state that *"...we note that when the key addition* *layer is composed of XOR, we can optimize the parity evaluations by applying the algorithm of* [18]". Others assume that the same formulas directly apply in the multiple-round case [6,23,29,30], and a few mention technical extended versions of the algorithm in dedicated cryptanalysis, for example [12,41], but a generalized algorithm for an arbitrary number of rounds in the key-recovery part has never been described in full.

It seems clear from the existing literature that the correct use of the key-recovery speed-up is not the norm, and its application is far from trivial. Furthermore, the treatment of the key-schedule relations has not been discussed either. A generalized FFT-based key-recovery algorithm would allow to build more efficient linear attacks easily. Taking into account the key-schedule relations and the scenario of multiple linear cryptanalysis in this algorithm also seem to be important tasks that should be considered.

Our main results. We have been able to provide an efficient generalized key-recovery algorithm with an associated time complexity formula. The algorithm is given in a matricial form (as opposed to the vectorial form of previous descriptions) as we believe it to be easier to understand and facilitate optimization in some cases, such as multiple linear attacks. When considering a linear attack with M approximations on a key-alternating cipher using N plaintext-ciphertext pairs with key-recovery on l_{ext} bits of the first and last subkey and l_{in} bits of the rest, the time complexity with our algorithm is

$$\mathcal{O}\left(N\right) + \mathcal{O}\left(Ml_{ext}2^{l_{ext}+l_{in}}\right).$$

In addition, we propose two methods which efficiently exploit the dependence relationships between the keybits that need to be guessed. The first reduces the second term to $\mathcal{O}\left(M2^{l_{ext}+l_{in}}\right)$, if some of the bits of the external keys can be deduced from the internal keys. The second allows to reduce the time complexity of this part to $\mathcal{O}\left(M2^{l_{tot}}\right)$ (where l_{tot} is the strict amount of information bits about the key which are necessary to deduce all the key-recovery bits) in some multiple linear attacks.

In our results on PRESENT we consider new multiple linear attacks which are only possible thanks to our algorithms, the best of which reach 28 rounds of the cipher for the first time. The expected time complexity was evaluated using the statistical model from [9]. In order to validate these predictions, we have implemented reduced-round versions of the attacks and found that the experimental results closely resemble the theoretical model.

Organization of the paper. Section 2 presents the preliminaries and notations that will be used throughout the paper, as well as the essential ideas of linear cryptanalysis, the 2007 FFT algorithm and PRESENT. In Sect. 3 we introduce our new generalized and efficient key-recovery algorithm and its variants. Section 4 describes the application to PRESENT and our new attacks, including discussions of the design of our linear distinguishers and key-recovery algorithms, as well as a comparison with previous attacks and the results of our experimental simulations. The conclusions of this paper are extracted in Sect. 5.

2 Preliminaries

We now cover some preliminaries and notations needed for the other sections of the paper. We briefly describe Matsui's Algorithm 2, which is the basis of linear key-recovery attacks. We also provide a short description of the ideas behind linear hulls and multiple linear cryptanalysis, as they are essential to our attacks on PRESENT. The statistical model that was used to compute the parameters of these attacks is also summarised. Next, we present the FFT-based algorithm which allows the speed-up of the key-recovery phase and was proposed in [18]. Finally we outline the specification of the PRESENT block cipher.

In the following, we will consider a block cipher E of length n and key length κ. Given a plaintext x and a key K, we denote the associated ciphertext by $y = E_K(x) = E(x, K)$, so that $E^{-1}(y, K) = E_K^{-1}(y) = x$. In particular we will consider key-alternating ciphers consisting of r rounds, each one being the composition of a round permutation F and the bitwise addition of a round subkey K_i which is derived from the master key K with a key schedule. We also consider that the first round is preceded by the addition of a whitening key K_0.

2.1 Matsui's Algorithm 2

Matsui's last round attack in [31] separates the last round of the cipher, $E_K(x) = (F \circ E'_K)(x) \oplus K_r$ as represented in Fig. 1, and supposes the attacker knows a correlated linear approximation $\alpha \cdot x \oplus \beta \cdot \hat{y} \oplus \gamma(K)$ of E'_K (where $\cdot$ denotes the dot product). The vectors α and β are the input and output masks, while γ determines the key mask. The correlation of the approximation is

$$\begin{aligned} c(\alpha, \beta, \gamma) = Pr_{x,K}(\alpha \cdot x \oplus \beta \cdot F^{-1}(E_K(x) \oplus K_r) \oplus \gamma(K) = 0) \\ - Pr_{x,K}(\alpha \cdot x \oplus \beta \cdot F^{-1}(E_K(x) \oplus K_r) \oplus \gamma(K) = 1). \end{aligned} \tag{1}$$

Matsui also proved that (under statistical independence assumptions) the correlation of the addition of several approximations is the product of their correlations (piling-up lemma). This allows to construct approximations of a cipher by chaining approximations of each individual round.

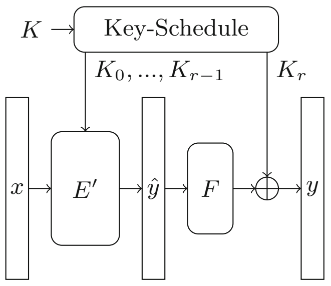

Fig. 1. Attack on last round of a cipher.

Algorithm 1: Naïve Matsui's Algorithm 2

```
Input: A set $\mathcal{D} = \{(x, y = E_K(x))\}$ of $N$ plaintext-ciphertext pairs.
Output: A probable guess for $k$.
$\mathbf{T} \leftarrow \underline{\mathbf{0}}$;
forall $(x, y) \in \mathcal{D}$ do          // Compute $T_k = \#\{(x,y) : \alpha \cdot x \oplus f(y|_\chi \oplus k) = 0\}$
    for $k \leftarrow 0$ to $2^{|k|} - 1$ do
        if $\alpha \cdot x \oplus f(y|_\chi \oplus k) = 0$ then $T_k \leftarrow T_k + 1$;
    end
end
return $argmax_k(|T_k - N/2|)$;          // Find the $T_k$ most different to $N/2$
```

We suppose that computing $\beta \cdot F^{-1}(y \oplus K_r)$ from y only requires guessing $|k| < |K_r| = n$ bits of K_r, which are selected by the mask χ (so that $k = K_r|_\chi$). Here $x|_\chi$ will denote the vector of length $HW(\chi)$ whose components are the coordinates of x corresponding to non-zero entries of χ, and $|x|$ just denotes the length of the vector x. We can substitute the term associated to the partial decryption of the last round for a map $f : \mathbb{F}_2^{|k|} \longrightarrow \mathbb{F}_2$:

$$f\left(y|_\chi \oplus K_r|_\chi\right) = \beta \cdot F^{-1}(y \oplus K_r) \text{ for all } y \in \mathbb{F}_2^n, K_r \in \mathbb{F}_2^n \tag{2}$$

Given a collection $\mathcal{D}$ of N plaintext-ciphertext pairs, the partial subkey k can be retrieved with Matsui's Algorithm 2, which relies on the assumption that for any wrong guess of the last round subkey, the linear approximation will have value 0 with probability 1/2. Matsui proved that the probability of success is reasonable when $N = \mathcal{O}\left(1/c(\alpha,\beta,\gamma)^2\right)$ pairs are available.

The complexity of the algorithm is $N2^{|k|}$ one-round decryptions and $2^{|k|}$ memory registers of up to $\log N$ bits to compute the counters T_k, with an additional $2^{\kappa-|k|}$ full encryptions if the rest of the key is searched for exhaustively.

In [32], Matsui noted that since the only information required about each (x, y) pair are the values of $\alpha \cdot x$ and $y|_\chi$, it is possible to first count the number of occurrences of each $(\alpha \cdot x,\ y|_\chi)$ in the data (*distillation phase*) and then compute the T_k using these counters (*analysis phase*). With this technique the attack takes N parity evaluations and $2^{2|k|}$ one-round decryptions, which reduces the complexity to $\mathcal{O}(N) + \mathcal{O}\left(2^{2|k|}\right)$ when $2^{|k|} < N$, which is often the case.

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 and the time complexity increase with the number of key-recovery rounds.

2.2 Linear Hulls

The original version of linear cryptanalysis by Matsui assumes that, given an input mask α and an output mask β, then there exists at most one key mask which leads to a biased approximation (in modern language, there is a dominant linear trail). This is often not the case, and there can exist many different sets of round subkey masks $(\gamma_0, \dots, \gamma_r)$ or linear trails which contribute to the linear approximation. Furthermore, when this happens, then the probability of success of Matsui's Algorithm 2 is dependant on the key K. Nyberg introduced the idea of the linear hull of an approximation in [36], as well as its linear potential:

$$ELP(\alpha, \beta) = \mathrm{Exp}_K(c(\alpha, \beta)^2) = \sum_{\gamma_0, \dots, \gamma_r} c(\alpha, \beta, (\gamma_0, \dots, \gamma_r))^2 \tag{3}$$

An Algorithm 2 attack using the approximation given by the masks α, β roughly requires $N = \mathcal{O}\left(1/ELP(\alpha, \beta)\right)$ plaintext-ciphertext pairs to succeed, although the specific success probability depends on the key K.

There are several algorithms which permit the estimation of the ELP of a linear approximation. In our attacks on PRESENT we used the sparse correlation matrix method in a similar manner to [1].

2.3 Multiple and Multidimensional Linear Attacks

Linear cryptanalysis can also be extended by using more than one linear approximation. The first approach to allow the use of any set of linear approximations was introduced by Biryukov et al. in [7], and is commonly referred to as *multiple linear cryptanalysis*.

We will now describe a multiple version of Matsui's Algorithm 2. Let ν_i be M linear approximations of E'_K with masks α_i, β_i. 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. For each guess of k, the attacker computes the empirical correlations

$$\begin{aligned} q_k^i &= \#\left\{(x, y) \in \mathcal{D} : \alpha_i \cdot x \oplus f_i(y|_\chi \oplus k) = 0\right\} \\ &\quad - \#\left\{(x, y) \in \mathcal{D} : \alpha_i \cdot x \oplus f_i(y|_\chi \oplus k) = 1\right\} \end{aligned} \tag{4}$$

which are then aggregated into the multiple linear cryptanalysis statistic

$$Q_k = \frac{1}{N} \sum_{i=1}^{M} \left(q_k^i\right)^2 \tag{5}$$

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 data complexity is inversely proportional to the capacity C of the set of 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

$$C(K) = \sum_{i=1}^{M} \left(c_i(K)\right)^2, \quad C = \mathrm{Exp}_K\left(C(K)\right) = \sum_{i=1}^{M} ELP(\alpha_i, \beta_i) \tag{6}$$

Hermelin et al. proposed multidimensional linear cryptanalysis in [25] and [26]. It uses linear approximations whose input and output masks constitute linear subspaces of $\mathbb{F}_2^n$, so 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.

2.4 Statistical Models for the Probability of Success

An issue that has also been studied is the probabilistic behaviour of linear approximations and how it can be used to better estimate the data complexity of a linear attack. In an attack based on Matsui's Algorithm 2, it is possible to keep more than one key candidate, which increases the probability of success. Selçuk introduced the notion of advantage [38] in order to measure the effectiveness of this type of attack. 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 a, the probability of success is the probability that the actual advantage surpasses a.

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

$$P_S = 1 - F_R\left(F_W^{-1}(1 - 2^{-a})\right) \tag{7}$$

For multiple and multidimensional linear cryptanalysis, Blondeau et al. have provided estimations of the distributions of the test statistics in [9]. These estimations can also be found in the Appendix C.

Another approach to estimating the probability of success was recently introduced by Bogdanov et al. in [12] with the name multivariate profiling. Its main advantage is the fact that it allows to use any set of linear approximations without supposing the statistical independence of the variables. In this case the estimate for the joint distribution of the correlation of the approximations is obtained by drawing a large enough sample of random keys, and computing the individual correlation contribution of each trail (in a large enough set of highly biased trails) for each of the random keys.

2.5 Last-Round Key-Recovery with FFT/FWT

We now describe the FFT-accelerated version of Algorithm 2 presented in [18], which applies to the construction from Fig. 1 and will be the starting point of our work in Sect. 3.

There are $2^{|k|}$ possibilities for the partial subkey guess, and we recall that χ is the mask which extracts these relevant bits, so $k = K_r|_\chi$. 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 $\mathbf{q} \in \mathbb{Z}^{2^m}$ of experimental correlations whose entries are

$$\begin{aligned} q_k = \#\{(x,y) \in \mathcal{D} : \alpha \cdot x \oplus f(y|_\chi \oplus k) = 0\} \\ - \#\{(x,y) \in \mathcal{D} : \alpha \cdot x \oplus f(y|_\chi \oplus k) = 1\} \end{aligned} \tag{8}$$

with the aim of extracting the key candidate(s) with the largest $|q_k|$ (as $q_k = 2T_k - N$). Each experimental correlation can be rewritten as a sum

$$q_k = \sum_{(x,y)\in\mathcal{D}} (-1)^{\alpha\cdot x \oplus f(y|_\chi \oplus k)} = \sum_{j=0}^{2^{|k|}-1} (-1)^{f(j\oplus k)} \sum_{\substack{(x,y)\in\mathcal{D} \\ y|_\chi = j}} (-1)^{\alpha\cdot x} \tag{9}$$

where 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

$$a_j = \sum_{\substack{(x,y)\in\mathcal{D} \\ y|_{\mathcal{X}}=j}} (-1)^{\alpha\cdot x} \tag{10}$$

This constitutes the distillation phase of the algorithm of [18]. We can also define the matrix $C \in \mathbb{Z}^{2^{|k|}\times 2^{|k|}}$ with entries

$$c_{jk} = (-1)^{f(j\oplus k)} \tag{11}$$

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

$$\mathbf{q}^T = \mathbf{a}^T C \tag{12}$$

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:

Proposition 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*

$$c_{ij} = (-1)^{f(i\oplus j)}, \quad 0 \le i, j \le 2^m - 1 \tag{13}$$

This matrix diagonalizes as

$$2^m C = H_{2^m} \Delta H_{2^m} \tag{14}$$

where H_{2^m} is the Hadamard-Sylvester matrix of size 2^m whose entries are $h_{ij} = (-1)^{i\cdot j}$, and $\Delta = \operatorname{diag}(\boldsymbol{\lambda})$, $\boldsymbol{\lambda} \in \mathbb{Z}^{2^m}$ 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.

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

$$2^{|k|}\mathbf{q}^T = \mathbf{a}^T H_{2^{|k|}} \operatorname{diag}(H_{2^{|k|}} C_{\cdot 1}) H_{2^{|k|}} \tag{15}$$

The decomposition of C justifies Algorithm 2, which reduces computing $\mathbf{q}$ to three products of the form $H_{2^{|k|}}\mathbf{v}$, which can in turn be evaluated efficiently with the Fast Walsh Transform (sometimes called Fast Walsh-Hadamard Transform or simply FWT or FWHT) with $|k|2^{|k|}$ additions/substractions (see the appendix for more details). We denote by ρ_D the cost of checking a plaintext-ciphertext pair in the distillation phase, by ρ_f the cost of evaluating $f(j)$, by ρ_A, ρ_M, ρ_C the cost of adding, multiplying and comparing two n-bit integers, by ρ_E the cost of one encryption and by a the advantage of the attack.

Proposition 2. *The previous algorithm has time complexity*

$$\underbrace{\rho_D N}_{\substack{distillation \\ phase}} + \underbrace{3\rho_A|k|2^{|k|} + (\rho_f + \rho_M + \rho_C)2^{|k|}}_{analysis\ phase} + \underbrace{\rho_E 2^{\kappa-a}}_{\substack{search \\ phase}} \tag{16}$$

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

Algorithm 2: The algorithm of [18] (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 $2^{|k|}$).
// 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 $\lambda_j \leftarrow f(j)$;      // First column of $C$
$\boldsymbol{\lambda} \leftarrow \mathrm{FWT}(\boldsymbol{\lambda})$;      // Eigenvalues of $C$
$\mathbf{a} \leftarrow \mathrm{FWT}(\mathbf{a})$;      // Apply the FWT to a
for $j \leftarrow 0$ to $2^{|k|} - 1$ do $a_j \leftarrow a_j \cdot \lambda_j$;      // Multiply a by $\boldsymbol{\lambda}$
$\mathbf{q} \leftarrow \mathrm{FWT}(\mathbf{a})$;      // Apply the FWT to a
return $\boldsymbol{q}$;
```

2.6 The Lightweight Block Cipher 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 whitening key at the output. Each round is the composition of 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 S-box $S : \mathbb{F}_2^4 \longrightarrow \mathbb{F}_2^4$ is applied to each nibble w_i of the state. The S-box S is given as a lookup table (in hexadecimal notation):

$\boldsymbol{x}$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
$\boldsymbol{S(x)}$	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.

$$\begin{aligned} P : \{0, \dots 63\} &\longrightarrow \{0, \dots, 63\} \\ j \neq 63 &\longmapsto 16j \bmod 63 \\ 63 &\longmapsto 63 \end{aligned} \tag{17}$$

Key-schedule. It is the only difference between the 80 and the 128-bit variants, both algorithms can be found in Appendix A.

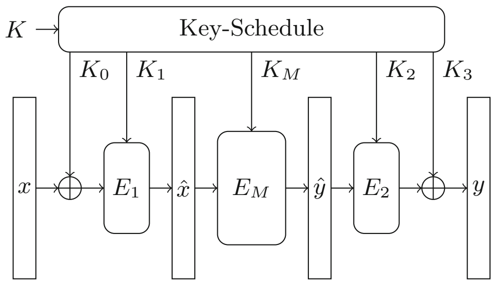

Fig. 2. The general description of the cipher.

3 Efficient Key-Recovery for Algorithm 2

In this section we will present our generalized efficient key-recovery algorithm inspired from the one in [18], described in Sect. 2.5.

We were surprised to see that after the publication of [18], many new linear attack publications did not use the algorithm to speed up the key-recovery part (see for instance [8,16,33]), or they just used it without getting into the details as a black box (see [2,3]). A few publications implicitly used extensions of the technique, such as [12,41], always in the context of a dedicated attack.

Here we propose a generalized version of the algorithm for an arbitrary number of rounds which encompasses these contributions and permits a finer analysis of the time complexity. We also propose two variants of the algorithm which efficiently exploit the key-schedule relations between keybits as well as the multiple approximation setting, which are interesting scenarios requiring consideration.

We believe that the new algorithm and its variants will simplify the evaluation of the time complexity of an attack given a suitable linear distinguisher, which would in turn help designers assess the security margin of a block cipher.

3.1 The Extended Algorithm

Consider a block cipher E of block size n and key size κ which can be decomposed as in Fig. 2. The ciphers E_1 and E_2 represent the first and last few rounds. They take some *inner keys* K_1 and K_2. The first and last round will be the *outer keys* K_0 and K_3. We suppose that the inner cipher E_M has a linear approximation

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

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

$$i = x|_{\chi_0},\ k_0 = K_0|_{\chi_0},\ k_1 = K_1|_{\chi_1},\ k_2 = K_2|_{\chi_2},\ j = y|_{\chi_3},\ k_3 = K_3|_{\chi_3} \qquad (18)$$

Let $f_1 : \mathbb{F}_2^{2^{|k_0|}} \times \mathbb{F}_2^{2^{|k_1|}} \longrightarrow \mathbb{F}_2$ and $f_2 : \mathbb{F}_2^{2^{|k_3|}} \times \mathbb{F}_2^{2^{|k_2|}} \longrightarrow \mathbb{F}_2$ be maps for which

$$f_1\left(x|_{\chi_0} \oplus K_0|_{\chi_0}, K_1|_{\chi_1}\right) = \alpha \cdot E_1\left(x \oplus K_0, K_1\right) \text{ for all } x, K_0, K_1 \tag{19}$$

$$f_2\left(y|_{\chi_3} \oplus K_3|_{\chi_3}, K_2|_{\chi_2}\right) = \beta \cdot E_2^{-1}\left(y \oplus K_3, K_2\right) \text{ for all } y, K_3, K_2 \tag{20}$$

The attacker has a set $\mathcal{D}$ of N pairs $(x, y = E(x, K))$ for some fixed key K. They need to compute, for each possible guess of the subkeys:

$$\begin{aligned} q(k_0, k_1, k_2, k_3) = \# \{(x, y) \in \mathcal{D} : f_1(i \oplus k_0, k_1) \oplus f_2(j \oplus k_3, k_2) = 0\} \\ - \# \{(x, y) \in \mathcal{D} : f_1(i \oplus k_0, k_1) \oplus f_2(j \oplus k_3, k_2) = 1\} \end{aligned} \tag{21}$$

The attack begins with the distillation phase, in which a matrix $A \in \mathbb{Z}^{2^{|k_0|} \times 2^{|k_3|}}$ is constructed from the data. Its entries are

$$a_{ij} = \# \{(x, y) \in \mathcal{D} :\ x|_{\chi_0} = i,\ y|_{\chi_3} = j\}. \tag{22}$$

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

$$q(k_0, k_1, k_2, k_3) = \sum_{i=0}^{2^{|k_0|}-1} \sum_{j=0}^{2^{|k_3|}-1} a_{ij} (-1)^{f_1(i \oplus k_0, k_1)} (-1)^{f_2(j \oplus k_3, k_2)} \tag{23}$$

Let us now consider that the values of k_1 and k_2 are fixed. The associated experimental correlations form a matrix $Q^{k_1,k_2} \in \mathbb{Z}^{2^{|k_0|} \times 2^{|k_3|}}$ with entries

$$q_{k_0,k_3}^{k_1,k_2} = q(k_0, k_1, k_2, k_3) \tag{24}$$

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

$$b_{k_0,i}^{k_1} = (-1)^{f_1(i \oplus k_0, k_1)}, \quad c_{j,k_3}^{k_2} = (-1)^{f_2(j \oplus k_3, k_2)} \tag{25}$$

The matrices B^{k_1}, C^{k_2} adhere to the structure described in Proposition 1, and the Fast Walsh Transform can be used to multiply vector by them.

$$2^{|k_0|} B^{k_1} = H_{2^{|k_0|}} \operatorname{diag}\left(\boldsymbol{\lambda}_1^{k_1}\right) H_{2^{|k_0|}}, \text{ where } \boldsymbol{\lambda}_1^{k_1} = H_{2^{|k_0|}} B_{\cdot 1}^{k_1} \tag{26}$$

$$2^{|k_3|} C^{k_2} = H_{2^{|k_3|}} \operatorname{diag}\left(\boldsymbol{\lambda}_2^{k_2}\right) H_{2^{|k_3|}}, \text{ where } \boldsymbol{\lambda}_2^{k_2} = H_{2^{|k_3|}} C_{\cdot 1}^{k_2} \tag{27}$$

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

$$2^{|k_0|+|k_3|} Q^{k_1 k_2} = H_{2^{|k_0|}} \operatorname{diag}\left(\boldsymbol{\lambda}_1^{k_1}\right) H_{2^{|k_0|}} A H_{2^{|k_3|}} \operatorname{diag}\left(\boldsymbol{\lambda}_2^{k_2}\right) H_{2^{|k_3|}} \tag{28}$$

As a result, the attack can be performed efficiently using Algorithm 3 as follows:

1. **Distillation phase:** Construct the matrix A by looking at each plaintext-ciphertext pair (x, y), finding the associated values of $i = x|_{\chi_0}$ and $j = y|_{\chi_3}$ and incrementing the corresponding a_{ij} by one.

Algorithm 3: General FFT algorithm (without the final phase)

```
Input: A collection $\mathcal{D} = \{(x, y = E_K(x))\}$ of $N$ plaintext-ciphertext pairs.
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 $A_{i\cdot} \leftarrow \mathrm{FWT}(A_{i\cdot})$;                         // FWT on rows
for $j \leftarrow 0$ to $2^{|k_3|} - 1$ do $A_{\cdot j} \leftarrow \mathrm{FWT}(A_{\cdot j})$;                     // FWT on columns
for $k_1 \leftarrow 0$ to $2^{|k_1|} - 1$; $i \leftarrow 0$ to $2^{|k_0|} - 1$ do $(\lambda_1^{k_1})_i \leftarrow f_1(i, k_1)$;      // $B^{k_1}_{\cdot 1}$
for $k_2 \leftarrow 0$ to $2^{|k_2|} - 1$; $j \leftarrow 0$ to $2^{|k_3|} - 1$ do $(\lambda_2^{k_2})_j \leftarrow f_2(j, k_2)$;      // $C^{k_2}_{\cdot 1}$
for $k_1 \leftarrow 0$ to $2^{|k_1|} - 1$ do $\boldsymbol{\lambda}_1^{k_1} \leftarrow \mathrm{FWT}(\boldsymbol{\lambda}_1^{k_1})$;                  // Compute $\boldsymbol{\lambda}_1^{k_1}$
for $k_2 \leftarrow 0$ to $2^{|k_2|} - 1$ do $\boldsymbol{\lambda}_2^{k_2} \leftarrow \mathrm{FWT}(\boldsymbol{\lambda}_2^{k_2})$;                  // 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 $Q^{k_1,k_2}_{k_0\cdot} \leftarrow \mathrm{FWT}(Q^{k_1,k_2}_{k_0\cdot})$;
    for $k_3 \leftarrow 0$ to $2^{|k_3|} - 1$ do $Q^{k_1,k_2}_{\cdot k_3} \leftarrow \mathrm{FWT}(Q^{k_1,k_2}_{\cdot k_3})$;
end
return $\{Q^{k_1,k_2}\}_{k_1,k_2}$;
```

2. **Analysis phase:** Compute all the experimental correlations $q(k_0, k_1, k_2, k_3)$:
   (a) Apply the FWT on all rows and columns of A to obtain a matrix $\widehat{A}$.
   (b) Construct the eigenvalue vectors $\boldsymbol{\lambda}_1^{k_1}$ and $\boldsymbol{\lambda}_2^{k_2}$ for all k_1, k_2 by calculating the first column of B^{k_1} or C^{k_2} and then applying the FWT.
   (c) Compute Q^{k_1,k_2} for all the values of k_1 and k_2:
      i. Copy $\widehat{A}$ and multiply each column by $\boldsymbol{\lambda}_1^{k_1}$ and each row by $\boldsymbol{\lambda}_2^{k_2}$ elementwise.
      ii. Apply the FWT on all the rows and columns to obtain Q^{k_1,k_2}.
   (d) Select the subkey guesses with the largest values of $|q(k_0, k_1, k_2, k_3)|$.
3. **Search phase.**

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 occupies $2^{|k_0|+|k_3|}$ memory registers of up to n bits. The cost of applying the initial FWTs of step 2(a) is $\rho_A \left(|k_0| + |k_3|\right) 2^{|k_0|+|k_3|}$ (ρ_A/ρ_M is the cost of addition/multiplication) with no additional memory. The eigenvalue vectors can be precomputed with cost

$$\rho_{f_1} 2^{|k_0|+|k_1|} + \rho_{f_2} 2^{|k_2|+|k_3|} + \rho_A \left(|k_0| 2^{|k_0|+|k_1|} + |k_3| 2^{|k_2|+|k_3|} \right) \tag{29}$$

where ρ_{f_1} and ρ_{f_2} are the costs of evaluating f_1 and f_2. These vectors are stored in $2^{|k_0|+|k_1|} + 2^{|k_2|+|k_3|}$ registers of $\max\{|k_0|, |k_3|\}$ bits. The cost of step 2(c) is

$$2\rho_M 2^{|k_0|+|k_1|+|k_2|+|k_3|} + \rho_A \left(|k_0| + |k_3|\right) 2^{|k_0|+|k_1|+|k_2|+|k_3|} \tag{30}$$

This computation requires $2^{|k_0|+|k_3|}$ working registers of up to $n+|k_0|+|k_3|$ bits. If the experimental correlations need to be stored in full (for example if the FFT algorithm is used as a part of a multiple linear attack), then $2^{|k_0|+|k_1|+|k_2|+|k_3|}$ memory registers of n bits are required (we can divide by $2^{|k_0|+|k_3|}$).

It's interesting to compare the computational costs $\rho_D, \rho_{f_1}, \rho_{f_2}, \rho_A$ and ρ_M with the cost of a block cipher encryption ρ_E. In general, ρ_D, ρ_{f_1} and ρ_{f_2} should be negligible, as they are much simpler operations. For most cases ρ_A and ρ_M should be comparable to or smaller to the cost of an encryption, though this depends on the implementations of the cipher and the operations.

The adaptability of this algorithm to multiple and multidimensional linear attacks should also be considered. The distillation phase only needs to be performed once, which means our approach generalises the results of [35]. If there is no structure to the set of approximations, then the time cost of the analysis phase is multiplied by the number of approximations M. Additionally, the cost of computing the statistic Q_k from the correlations of each approximation is

$$M(\rho_M + \rho_A)2^{|k_0|+|k_1|+|k_2|+|k_3|} \tag{31}$$

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 in time complexity.

3.2 Exploiting the Key Schedule of the Cipher

So far, we have assumed that the attacker must guess k_0, k_1, k_2 and k_3 independently. However, the key schedule of a cipher often induces dependence relationships between these four subkeys. These relationships can be easily exploited in the implementation of Matsui's Algorithm 2 without FFT, but it is not obvious how they can be used in accelerated attacks. We will now consider two strategies.

Walsh Transform Pruning

The first approach consists of applying the FWT algorithm but only computing the outputs which correspond to possible values of the subkeys, as suggested in [41]. To this end, we have studied pruned Walsh Transform algorithms, which efficiently compute a subset of outputs of the classical "full" transform. We have found a particularly useful pruned algorithm, which is detailed in Appendix B:

Proposition 3. *The components of the Walsh Transform of a vector of length 2^m which have n fixed bits in their position can be computed with complexity*

$$2^m + (m-n-1)2^{m-n} \tag{32}$$

We have designed a modified analysis phase which considers the bits of k_0 which can be deduced from (k_1, k_2) and the bits of k_3 which can be deduced from (k_0, k_1, k_2). The roles of k_0 and k_3 can be easily exchanged if necessary.

1. Compute $H_{2^{k_0}} A H_{2^{k_3}}$ as normal.
2. Only compute the products $\mathrm{diag}(\boldsymbol{\lambda}_1^{k_1}) H_{2^{k_0}} A H_{2^{k_3}} \mathrm{diag}(\boldsymbol{\lambda}_2^{k_2})$ for the values of (k_1, k_2) which are possible according to the key schedule.
3. For each pair (k_1, k_2), consider only the possible values of k_0 and prune the associated (column) Walsh Transforms accordingly.
4. For each of the rows of the resulting matrix, consider the possible values of k_3 and prune the associated Walsh Transform to these positions.

If (k_1, k_2) can only take $2^{|k_1|+|k_2|-l_{12}}$ different values, l_0 bits of k_0 can be deduced from (k_1, k_2) and l_3 bits of k_3 can be deduced from (k_0, k_1, k_2) then the complexity of the "pruned" analysis phase is

$$\begin{aligned} &\rho_A(|k_0|+|k_3|)2^{|k_0|+|k_3|} + 2\rho_M 2^{|k_0|+|k_1|+|k_2|+|k_3|-l_{12}} \\ &+ \rho_A 2^{|k_1|+|k_2|+|k_3|-l_{12}} \left(2^{|k_0|} + (|k_0|-l_0-1)2^{|k_0|-l_0}\right) \\ &+ \rho_A 2^{|k_0|-l_0+|k_1|+|k_2|-l_{12}} \left(2^{|k_3|} + (|k_3|-l_3-1)2^{|k_3|-l_3}\right) \end{aligned} \tag{33}$$

As l_0 and l_3 increase with respect to $|k_0|$ and $|k_3|$, the complexity approaches

$$\rho_A(|k_0|+|k_3|)2^{|k_0|+|k_3|} + 2\left(\rho_M+\rho_A\right)2^{|k_0|+|k_1|+|k_2|+|k_3|-l_{12}} \tag{34}$$

This variant of the attack algorithm requires $2^{|k_0|+|k_3|}$ memory registers to hold the counters from the distillation phase, $2^{|k_0|+|k_1|} + 2^{|k_2|+|k_3|}$ registers for the eigenvalue vectors and $2^{|k_0|+|k_1|+|k_2|+|k_3|-l_{12}-l_0-l_3}$ registers to hold the experimental correlations, if they need to be stored in full.

Since applying Walsh Transform on all the rows and columns of a matrix is equivalent to performing the Walsh Transform on a vectorization of the matrix, it should be possible to prune this unique transform to the possible values of (k_0, k_3) associated to the current (k_1, k_2). In particular, it would be interesting to consider more complex relationships between these bits.

Multiple Linear Cryptanalysis

The previous approach has limited results if $2^{|k_0|+|k_3|}$ is already too large. In the case of multiple linear cryptanalysis, it is possible to reduce the complexity further by performing the algorithm separately for each linear approximation and then combining the information to obtain Q_k, as done in [41] and [12].

Let $\nu_i : \alpha_i \cdot \hat{x} \oplus \beta_i \cdot \hat{y},\ i = 1, \ldots, M$ be a linear approximations of the inner cipher E_M. Multiple linear cryptanalysis requires the attacker to compute

$$Q(k_0, k_1, k_2, k_3) = \frac{1}{N} \sum_{i=1}^{M} \left(q^i(k_0, k_1, k_2, k_3)\right)^2 \tag{35}$$

In order to calculate one particular $\mathbf{q}^i$ 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 $\widehat{q}^i(k_0^i, k_1^i, k_2^i, k_3^i)$ (where $k_0^i = k_0|_{\chi_0^i}$ is a part of k_0, and so on), and that (k_0, k_1, k_2, k_3) can be deduced from a part k_T of the master key K. 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 M_1 and M_2 different values over the set of M approximations, respectively. In this situation, the attacker can perform the following modified attack:

1. In the distillation phase, construct M_1 tables: for each plaintext-ciphertext mask pair (X_0, X_3), the table $A_{(X_0,X_3)}$ of size $2^{HW(\chi_0^i)} \times 2^{HW(\chi_3^i)}$.
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 FFT algorithm on the appropriate table from the distillation phase, $A_{(\chi_0^i,\chi_3^i)}$.
3. Merge the M tables from the previous step into M_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 (X_0, X_1, X_2, X_3). The associated condensed table contains the coefficients:
$$\sum_{\substack{(\chi_0^i,\chi_1^i,\chi_2^i,\chi_3^i)\\=(X_0,X_1,X_2,X_3)}} \left(q^i(k_0^i, k_1^i, k_2^i, k_3^i)\right)^2 \quad \text{for all} \quad (k_0^i, k_1^i, k_2^i, k_3^i) \tag{36}$$
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 $Q(k_0, k_1, k_2, k_3)$.

Note that the individual calls to the FFT linear cryptanalysis algorithm can also be pruned in order to combine both key schedule exploitation approaches, and that steps 2 and 3 can be mixed in order to reduce the memory requirement.

The cost of the distillation phase is now $M_1\rho_D N$. If ρ_{KS} denotes the cost of computing (k_0, k_1, k_2, k_3) from k_T, then the cost of the analysis phase is

$$\begin{aligned}\sum_{i=1}^{M} &\left(2\rho_M 2^{|k_0^i|+|k_1^i|+|k_2^i|+|k_3^i|} + \rho_A(|k_0^i| + |k_3^i|)\left(2^{|k_0^i|+|k_1^i|+|k_2^i|+|k_3^i|} + 2^{|k_0^i|+|k_3^i|}\right)\right)\\ &+\sum_{i=1}^{M}\left(\rho_M 2^{|k_0^i|+|k_1^i|+|k_2^i|+|k_3^i|} + \rho_A 2^{|k_0^i|+|k_1^i|+|k_2^i|+|k_3^i|}\right) + (M_2\rho_A + \rho_{KS})2^{|k_T|}\end{aligned} \tag{37}$$

The algorithm requires $\sum_{(X_0,X_3)} 2^{HW(X_0)+HW(X_3)}$ memory positions for the distillation counters and $\sum_{(X_0,X_1,X_2,X_3)} 2^{HW(X_0)+HW(X_1)+HW(X_2)+HW(X_3)}$ positions for the M_2 condensed correlation tables.

This algorithm can produce large gains in the case of multiple linear cryptanalysis (especially when the $|k_\cdot^i|$ are significantly smaller than the $|k_\cdot|$), but its success is more limited in multidimensional attacks, as there is always a linear approximation for which the $|k_\cdot^i|$ are maximal.

Example implementation. We have implemented our key-recovery algorithm on a toy version of PRESENT with the aim of illustrating how all these different techniques can be used. The C code can be found at:

`https://project.inria.fr/quasymodo/results/codes/.`

4 Application to PRESENT

In order to showcase the potential of our key-recovery techniques, in this section we describe some new attacks on reduced-round variants of the block cipher PRESENT, which surpass the best previously known attacks (that were also linear), specifically [12,16,41]. Our results include new attacks on 26 and 27-round PRESENT which improve the parameters of the aforementioned attacks, as well as (to the best of our knowledge) the first attacks on 28-round PRESENT.

4.1 Linear Distinguishers for PRESENT

Previous linear attacks on PRESENT [12,16,33,37,41] have used the fact that the S-box has eight linear approximations with correlation 2^{-3} and whose input and output masks have Hamming weight 1. These approximations lead to many linear trails with one active S-box in each round, which form linear hulls with high potential and masks of weight 1. Our attacks make use of three different sets of approximations with masks of weight 1 or 2, which were found as follows.

We begin by computing the correlation of all approximations of one round of PRESENT which: (1) only have up to two active S-boxes and (2) only require up to two active S-boxes in the previous and the next rounds. There are 2800 input and output masks which verify these bounds on the number of active S-boxes, so a 2800×2800 correlation matrix was constructed. Then, the element-wise square of this matrix can be elevated to the number of rounds r to obtain an approximation of the ELP of all the linear approximations whose input and output masks are in this family. A similar approach is detailed in [1].

The analysis of the resulting matrices showed that the linear approximations of PRESENT with the largest ELP only have one active S-box in the first and the last rounds. Table 1 contains a classification (according to the ELP) of approximations with one active S-box in the first and the last round, and masks of Hamming weight 1 or 2. From these approximations, we have selected three different sets as linear distinguishers, considering both their linear capacity as well as the number of keybits involved in a two-round key-recovery.

Set I, with 128 approximations, has the lowest capacity, but only uses masks of Hamming weight 1 and has a cheaper key-recovery than the others. Set III has 448 approximations and the largest capacity but requires guessing a lot of bits in the key-recovery, as it has approximations with both masks of Hamming weight 2. Set II is an intermediate where masks of Hamming weight 2 are only used in the input. The capacity for these three sets can be found in Table 2. We have also estimated the advantage that is obtained by these sets of approximations using the statistical model that can be found in [9] and Appendix C.

Table 1. An empirical classification of linear approximations of PRESENT with input and output masks of Hamming weight 1 or 2 according to their ELP. Indicated are the active S-box of the first and last rounds, as well as the input and output masks of said S-boxes. Our three sets of approximations are indicated as I:*, II:○ and III:†.

Group	Input Mask	Input S-Box	Output Mask	Output S-Box	Qty.	ELP $r = 22$	ELP $r = 23$	ELP $r = 24$
A	$A^{\dagger}_{\circ}$	$5^{\dagger}_{\circ}, 6^{\dagger}_{\circ}, 9^{\dagger}_{\circ}, 10^{\dagger}_{\circ}$	$2^{\dagger}_{\circ}, 8^{\dagger}_{\circ}, 3^{\dagger}, 9^{\dagger}$	$5^{\dagger}_{\circ}, 7^{\dagger}_{\circ}, 13^{\dagger}_{\circ}, 15^{\dagger}_{\circ}$	64	$2^{-59.9}$	$2^{-62.5}$	$2^{-65.1}$
B	$C^{\dagger}_{\circ}$	$5^{\dagger}_{\circ}, 6^{\dagger}_{\circ}, 9^{\dagger}_{\circ}, 10^{\dagger}_{\circ}$	$2^{\dagger}_{\circ}, 8^{\dagger}_{\circ}, 3^{\dagger}, 9^{\dagger}$	$5^{\dagger}_{\circ}, 7^{\dagger}_{\circ}, 13^{\dagger}_{\circ}, 15^{\dagger}_{\circ}$	64	$2^{-60.4}$	$2^{-63.0}$	$2^{-65.6}$
C1	$A^{\dagger}_{\circ}$	$5^{\dagger}_{\circ}, 6^{\dagger}_{\circ}, 9^{\dagger}_{\circ}, 10^{\dagger}_{\circ}$	$2^{\dagger}_{\circ}, 8^{\dagger}_{\circ}, 3^{\dagger}, 9^{\dagger}$	$6^{\dagger}_{\circ}, 9, 11, 14^{\dagger}_{\circ}$	64	$2^{-60.6}$	$2^{-63.2}$	$2^{-65.8}$
C2	A	$5, 6, 9, 10$	$4, 5$	$5, 7, 13, 15$	32			
C3	$A^{\dagger}_{\circ}$	$7^{\dagger}, 11^{\dagger}, 13^{\dagger}_{\circ}, 14^{\dagger}_{\circ}$	$2^{\dagger}_{\circ}, 8^{\dagger}_{\circ}, 3^{\dagger}, 9^{\dagger}$	$5^{\dagger}_{\circ}, 7^{\dagger}_{\circ}, 13^{\dagger}_{\circ}, 15^{\dagger}_{\circ}$	64			
D	${}^{*}2^{\dagger}_{\circ}, {}^{*}4^{\dagger}_{\circ}, 3, 5$	${}^{*}5^{\dagger}_{\circ}, {}^{*}6^{\dagger}_{\circ}, {}^{*}9^{\dagger}_{\circ}, {}^{*}10^{\dagger}_{\circ}$	${}^{*}2^{\dagger}_{\circ}, {}^{*}8^{\dagger}_{\circ}, 3^{\dagger}, 9^{\dagger}$	${}^{*}5^{\dagger}_{\circ}, {}^{*}7^{\dagger}_{\circ}, {}^{*}13^{\dagger}_{\circ}, {}^{*}15^{\dagger}_{\circ}$	256	$2^{-60.8}$	$2^{-63.4}$	$2^{-66.0}$
E1	$C^{\dagger}_{\circ}$	$5^{\dagger}_{\circ}, 6^{\dagger}_{\circ}, 9^{\dagger}_{\circ}, 10^{\dagger}_{\circ}$	$2^{\dagger}_{\circ}, 8^{\dagger}_{\circ}, 3^{\dagger}, 9^{\dagger}$	$6^{\dagger}_{\circ}, 9, 11, 14^{\dagger}_{\circ}$	64	$2^{-61.1}$	$2^{-63.7}$	$2^{-66.3}$
E2	C	5,6,9,10	4,5	5,7,13,15	32			
E3	$C^{\dagger}_{\circ}$	$7^{\dagger}, 11^{\dagger}, 13^{\dagger}_{\circ}, 14^{\dagger}_{\circ}$	$2^{\dagger}_{\circ}, 8^{\dagger}_{\circ}, 3^{\dagger}, 9^{\dagger}$	$5^{\dagger}_{\circ}, 7^{\dagger}_{\circ}, 13^{\dagger}_{\circ}, 15^{\dagger}_{\circ}$	64			
F1	A	$5, 6, 9, 10$	$2, 8, 3, 9$	10	16	$2^{-61.3}$	$2^{-63.9}$	$2^{-66.5}$
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	$A_{\circ}$	$7, 11, 13_{\circ}, 14_{\circ}$	$2_{\circ}, 8_{\circ}, 3, 9$	$6_{\circ}, 9, 11, 14_{\circ}$	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	${}^{*}2_{\circ}, {}^{*}4_{\circ}, 3, 5$	${}^{*}5_{\circ}, {}^{*}6_{\circ}, {}^{*}9_{\circ}, {}^{*}10_{\circ}$	${}^{*}2_{\circ}, {}^{*}8_{\circ}, 3, 9$	${}^{*}6_{\circ}, 9, 11, {}^{*}14_{\circ}$	256	$2^{-61.5}$	$2^{-64.1}$	$2^{-66.7}$
G2	$2, 4, 3, 5$	$5, 6, 9, 10$	$4, 5$	$5, 7, 13, 15$	128			
G3	$8_{\circ}, 9$	$5_{\circ}, 6_{\circ}, 9_{\circ}, 10_{\circ}$	$2_{\circ}, 8_{\circ}, 3, 9$	$5_{\circ}, 7_{\circ}, 13_{\circ}, 15_{\circ}$	64			
G4	${}^{*}2_{\circ}, {}^{*}4_{\circ}, 3, 5$	$7, 11, {}^{*}13_{\circ}, {}^{*}14_{\circ}$	${}^{*}2_{\circ}, {}^{*}8_{\circ}, 3, 9$	${}^{*}5_{\circ}, {}^{*}7_{\circ}, {}^{*}13_{\circ}, {}^{*}15_{\circ}$	256			

Table 2. The capacities of our three sets of approximations.

	# **Approx.**	**Capacity** ($r = 22$)	**Capacity** ($r = 23$)	**Capacity** ($r = 24$)
I (*)	128	$2^{-54.11}$	$2^{-56.71}$	$2^{-59.31}$
II (○)	296	$2^{-52.60}$	$2^{-55.20}$	$2^{-57.80}$
III (†)	448	$2^{-51.78}$	$2^{-54.38}$	$2^{-56.98}$

These approximations are not statistically independent (as they are not even linearly independent). One possible solution would be the application of multidimensional linear cryptanalysis. However, this would consider all the linear combinations of the approximations, and the benefits of the masks of low Hamming weight would be lost. Instead, we use the multiple linear cryptanalysis statistic, and we have estimated the probability of success under the assumption that the approximations are statistically independent. In order to justify the validity of the resulting estimations, we provide experimental results which conform to the theoretical predictions for a reduced number of rounds. Another possible approach would be the multivariate profiling technique of [12].

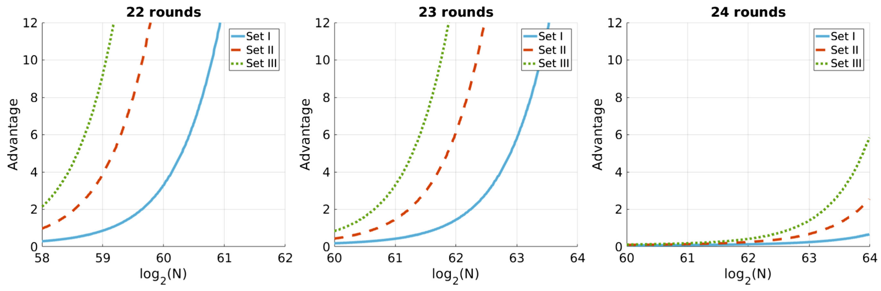

Fig. 3. Advantage obtained by each of our sets of approximations for 22, 23 and 24 rounds of PRESENT with 0.95 probability in a distinct known plaintext scenario.

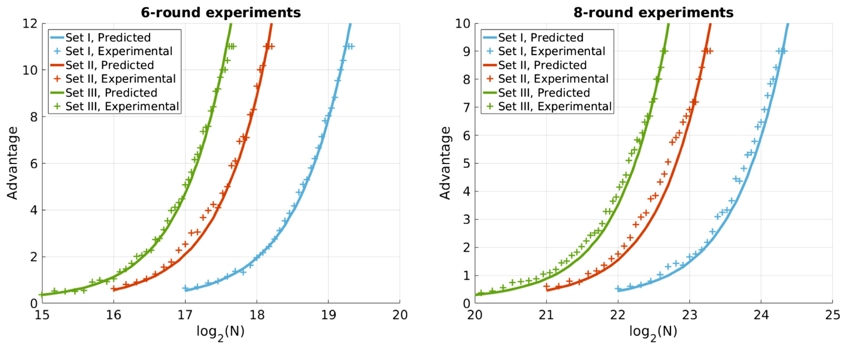

Fig. 4. Experimental advantage for attacks on 10 (resp. 12) rounds of PRESENT (using the linear distinguishers over 6 (resp. 8) rounds, with key-recovery on the first two and last two rounds). The statistic Q_k of the right key was compared against a random sample of 2^{12} (resp. 2^{10}) keys. The position of the right-key statistic among these provides an estimation of the advantage of up to 12 (resp. 10) bits. This was repeated for 20 different random right keys and 20 different random data samples for each value of N, providing a sample of 400 values of the advantage. The 5th percentile was used as an estimation of the advantage that's achieved with probability 0.95.

Figures 3 and 4 contain our advantage predictions for the 22, 23 and 24 round distinguishers as well as experiments for 6 and 8 rounds.

4.2 Improved Key-Recovery Attacks on 26 and 27-Round PRESENT

The first attack on PRESENT that we propose is based on set I of linear approximations. Since this set is only effective on up to 23 internal rounds and the attack will perform a key-recovery on the first two and last two rounds, the attack is effective on up to 27 rounds. In order to describe of these attacks more easily, we make use of the following properties of the bit permutation:

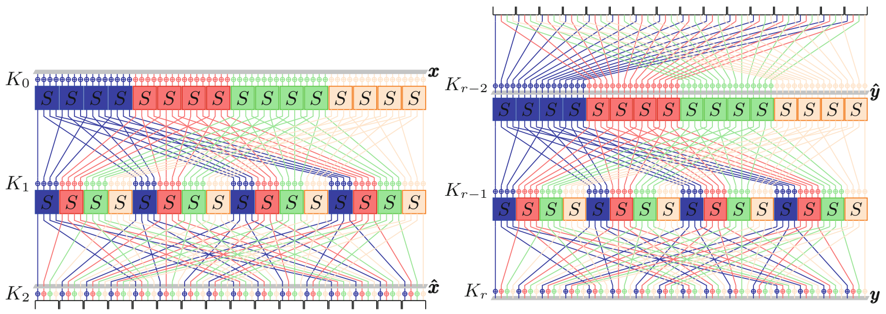

Fig. 5. The four groups of bits for the key-recovery on the first and last rounds.

Proposition 4 (Key-recovery on PRESENT). *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 four bits $\hat{x}_{48+4i+j}, \hat{x}_{32+4i+j}, \hat{x}_{16+4i+j}$ and $\hat{x}_{4i+j}$ 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 $\kappa^1_{16i+4j+3}\kappa^1_{16i+4j+2}\kappa^1_{16i+4j+1}\kappa^1_{16i+4j}$.*

Let $\tilde{y}$ be the state before the application of sBoxLayer in the $(r-1)$-th round of PRESENT. Given two fixed values of i, j between 0 and 3, the four bits $\tilde{y}_{P(16j+12+i)}, \tilde{y}_{P(16j+8+i)}, \tilde{y}_{P(16j+4+i)}$ and $\tilde{y}_{P(16j+i)}$ can be obtained from the 16 bits of the ciphertext $y_{60+i}, y_{56+i}, \dots, y_{4+i}, y_i$, as well as the 16 bits of the last round subkey $\kappa^r_{60+i}, \kappa^r_{56+i}, \dots, \kappa^r_{4+i}, \kappa^r_i$ and the 4 bits of the previous round subkey $\kappa^{r-1}_{48+4i+j}\kappa^{r-1}_{32+4i+j}\kappa^{r-1}_{16+4i+j}\kappa^{r-1}_{4i+j}$.

With the help of the previous proposition, we can mount key-recovery attacks on up to 27-round PRESENT-80 by extending approximation set I with two rounds of key-recovery at both the top and bottom of the cipher using our multiple linear cryptanalysis key-recovery algorithm. The parameters of the time complexity formula can be computed using Proposition 4, and the details on the key schedule for 26 and 27 rounds can be found in Figs. 6 and 7. In particular

$$\begin{aligned} &M_1 = 4,\ M_2 = 16,\ |k_0| = |k_3| = 32,\ |k_1| = |k_2| = 12 \\ &|k_0^i| = |k_3^i| = 16,\ |k_1^i| = |k_2^i| = 4 \text{ for all } i \\ &|k_T| = 61 \text{ for 26 rounds},\ |k_T| = 68 \text{ for 27 rounds} \end{aligned} \tag{38}$$

A simple lower bound on the cost of a PRESENT encryption ρ_E is $2 \cdot 64 \cdot r + 64$ binary operations (since each round requires at the very least adding the round subkey and writing each output bit for sBoxLayer). For 26 rounds, this is 3392 binary operations. On the other hand, $\rho_A \simeq 128$, $\rho_M \simeq 3 \cdot 64^{\log_2(3)} \simeq 2143$. This means that the time complexity of the analysis phase should be lower than 2^{65} full encryptions for 26 rounds and 2^{72} full encryptions for 27 rounds. The search phase time complexity depends on the available data and can

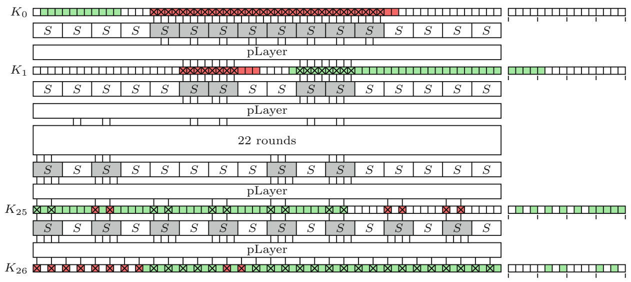

Fig. 6. Key-recovery on 26-round PRESENT-80 using approximation set I. The key-schedule effect is also represented in the figure. In total there are 96 bits of the subkeys which need to be guessed, which have been indicated by a cross. However, they can all be deduced from the $|k_T| = 61$ bits of key which have been highlighted in (dark) red. From these bits of key, all the bits in (light) green can be extracted, which includes all the necessary bits for the attack. (Color figure online)

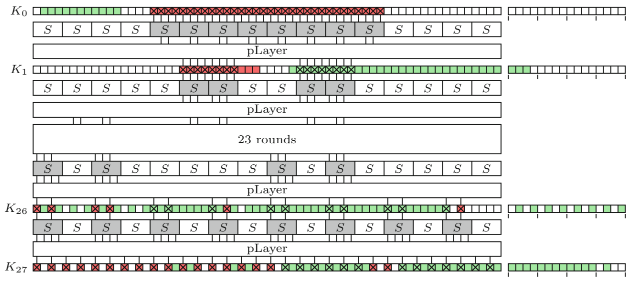

Fig. 7. Key-recovery on 27-round PRESENT-80 using approximation set I.

be estimated thanks to the graphs in Fig. 3. The complexities of both attacks are given in Table 3. These attacks can be easily extended to the 128-bit key variant.

4.3 Key-Recovery Attacks on 28-Round PRESENT

Sets II and III can be extended by two rounds of key-recovery at both sides to construct attacks on up to 28-round PRESENT. As set III has a larger capacity but requires an expensive key-recovery, we found that set II is best suited to attack PRESENT-80 and set III gives better results on PRESENT-128.

The parameters for an attack using approximation set II on PRESENT-80, with the key-schedule analysis represented in Fig. 8 are:

$$\begin{aligned} &M_1 = 8,\ M_2 = 32,\ |k_0| = 48,\ |k_1| = 24,\ |k_2| = 16,\ |k_3| = 32 \\ &|k_0^i| \leq 32,\ |k_1^i| \leq 8,\ |k_2^i| = 4,\ |k_3^i| = 16 \text{ for all } i,\ |k_T| = 73 \end{aligned} \tag{39}$$

This attack requires use of the Pruned Walsh Transform. There are 160 approximations for which the input and output masks have weight 1. For each of

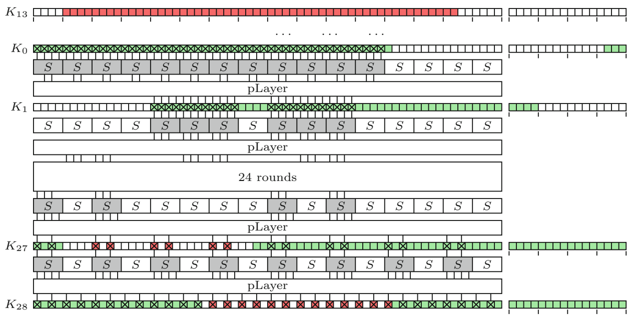

Fig. 8. Key-recovery on 28-round PRESENT-80 using approximation set II.

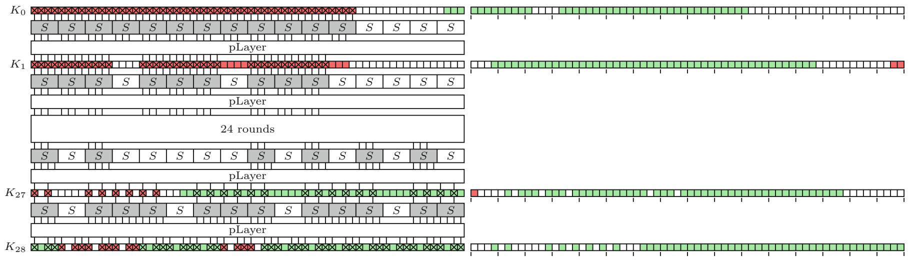

Fig. 9. Key-recovery on 28-round PRESENT-128 using approximation set III.

these approximations, computing the 2^{40} experimental correlations has cost $(16+16)\cdot 2^{16+4+4+16} = 2^{45}$ operations. For the remaining approximations, the cost should be $(32+16)2^{32+8+4+16} = 2^{65.58}$. However, all these approximations have an input S-box mask A or C. A look at the key-recovery diagrams shows that at least 5 bits of k_0^i can be deduced from k_1^i. By pruning the Walsh Transforms corresponding to the matrices B^{k_1}, the cost is reduced from 2^{37} to $2^{32.9}$ each. It also means that the memory requirement for each approximation is reduced by a factor of 2^5. After this first pruning, the transforms associated with the last two rounds (or the matrices C^{k_2}) can be pruned by fixing the bits of k_3 which can be deduced from k_0 and k_1, reducing the complexity of each transform to 2^{16}. This allows to keep the time complexity of the analysis phase below 2^{77} full PRESENT encryptions, and reduces the memory cost to 2^{51} registers.

For an attack using approximation set III on PRESENT-128, with the key-recovery part represented in Fig. 9 we have:

$$\begin{aligned} &M_1 = 16,\ M_2 = 96,\ |k_0| = 48,\ |k_1| = 36,\ |k_2| = 36,\ |k_3| = 48 \\ &|k_0^i| \le 32,\ |k_1^i| \le 8,\ |k_2^i| \le 8,\ |k_3^i| \le 32 \text{ for all } i,\ |k_T| = 114 \end{aligned} \qquad (40)$$

This means that the time complexity of the analysis phase of the attack should be smaller than 2^{121}. The memory is mainly devoted to the condensed correlation tables corresponding to the largest value of $|k_0^i| + |k_1^i| + |k_2^i| + |k_3^i|$, that is, for approximations which require 80 bits of subkey to be guessed. Since the correlation of these approximations can be condensed into 18 tables, we conclude that the memory cost is $18 \cdot 2^{80} \simeq 2^{84.6}$ memory registers of 80 bits. The complexities of these attacks can be found in Table 3.

For the table we have considered that the full codebook is available, but it is possible to consider different trade-offs between the available data and the time complexity of the exhaustive search. For instance, in the case of PRESENT-128, if $N = 2^{63.5}$ distinct plaintext-ciphertext pairs are available, the advantage is 2.8 bits. This translates into an attack with $2^{125.2}$ time complexity.

5 Conclusion

New general and efficient key-recovery algorithm. First and foremost, we have provided an efficient generalized key-recovery algorithm which applies to any number of rounds of a key-alternating cipher. We have also proposed two variants of this algorithm which allow to take key-schedule dependencies into account.

The new algorithm is not only capable of accelerating existing attacks, it is also sometimes possible to use more effective linear distinguishers than with previous algorithms. In the case of PRESENT, we chose approximations fitted to exploit the position of the key-recovery bits.

We expect that, in the future, this algorithm will not only represent a new cryptanalysis tool, but will also allow to easily and accurately evaluate the security margin of new primitives with respect to linear attacks.

Table 3. Comparison of linear attacks on PRESENT. DKP: Distinct known plaintexts.

#Rounds	Key size	#KR rounds	#Approx.	Capacity	Data	Time	Memory	Success prob.	Source
26	80	2	2295 (MD)	$2^{-55.38}$	$2^{64.0}$ KP	$2^{72.0}$	$2^{32.0}$	0.95	[16]
	80	2	2295 (MD)	$2^{-55.38}$	$2^{63.8}$ KP	$2^{72.0}$	$2^{32.0}$	0.51	[9,16] †
	80	4	135	$2^{-55.47}$	$2^{63.0}$ KP	$2^{68.6}$	$2^{48.0}$	0.95	[12] *
	80	4	128	$2^{-54.11}$	$2^{61.1}$ KP	$2^{68.2}$	$2^{44.0}$	0.95	Set I
	80	4	128	$2^{-54.11}$	$2^{60.8}$ KP	$2^{71.8}$	$2^{44.0}$	0.95	Set I
27	80	4	405 (MD)	$2^{-55.33}$	$2^{64.0}$ KP	$2^{74.0}$	$2^{67.0}$	0.95	[41] ‡
	80	4	135	$2^{-58.06}$	$2^{63.8}$ DKP	$2^{77.3}$	$2^{48.0}$	0.95	[12] *
	80	4	128	$2^{-56.71}$	$2^{63.4}$ DKP	$2^{72.0}$	$2^{44.0}$	0.95	Set I
28	80	4	296	$2^{-57.80}$	$2^{64.0}$ DKP	$2^{77.4}$	$2^{51.0}$	0.95	Set II
	128	4	448	$2^{-56.98}$	$2^{64.0}$ DKP	2^{122}	$2^{84.6}$	0.95	Set III

†: [9] reevaluated the success probability of [16] with a more recent statistical model.
‡: [41] effectively uses one fourth of the data, as well as an older statistical model.
*: The capacities differ from those of [12] ($2^{-55.01}$ and $2^{-56.38}$ for 26 and 27 rounds) due to the different methods for its estimation. Furthermore, here we consider just the signal component, while [12] also includes noise (second term in Eq. 46).

Best attacks on PRESENT. Thanks to our algorithms, we have been able to provide the best known attacks on reduced round PRESENT, which in particular reach 28 rounds, while the best previous ones only reached up to 27. We believe it would be very hard to extend this attack further without any new ideas, and PRESENT still seems secure with 3 (instead of 4) rounds of security margin.

Open problems

- It would be interesting to implement semi-automatic tools to find the key-recovery complexity for a given set of approximations. Or, even further, one which finds an optimal set of approximations in terms of linear capacity *and* cost of key-recovery. The first seems feasible, but the second seems harder. It would be very interesting to find some results in this direction.
- Better linear attacks on other primitives, like NOEKEON, TRIFLE-BC, Simon,...
- Future applications to other cryptanalysis families: in [10] an equivalent to the algorithm from [18] was applied to zero-correlation attacks, and in [39] the same was done regarding integral attacks. It might be possible to extend and generalize these algorithms as we did with linear key-recovery.

Acknowledgements. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 714294 - acronym QUASYModo).

A Key-Schedule of PRESENT

Algorithm 4: 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.
$\kappa^0_{63} \dots \kappa^0_0 \longleftarrow \kappa_{79} \dots \kappa_{16}$;                // Extract first round subkey
for $i \leftarrow 1$ to $r$ do
    $\kappa_{79} \dots \kappa_0 \longleftarrow \kappa_{18} \dots \kappa_{19}$;             // Rotate 19 bits to the right
    $\kappa_{79}\kappa_{78}\kappa_{77}\kappa_{76} \longleftarrow S(\kappa_{79}\kappa_{78}\kappa_{77}\kappa_{76})$;       // $S$ 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 round counter
    $\kappa^i_{63} \dots \kappa^i_0 \longleftarrow \kappa_{79} \dots \kappa_{16}$;                // Extract round subkey
end
return $\{K_i\}_{i=0}^r$;
```

Algorithm 5: 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.
$\kappa^0_{63} \dots \kappa^0_0 \longleftarrow \kappa_{127} \dots \kappa_{64}$;          // Extract the first round subkey
for $i \leftarrow 1$ to $r$ do
    $\kappa_{127} \dots \kappa_0 \longleftarrow \kappa_{66} \dots \kappa_{67}$;          // Rotate 61 bits to the left
    $\kappa_{127}\kappa_{126}\kappa_{125}\kappa_{124} \longleftarrow S(\kappa_{127}\kappa_{126}\kappa_{125}\kappa_{124})$;
    $\kappa_{123}\kappa_{122}\kappa_{121}\kappa_{120} \longleftarrow S(\kappa_{123}\kappa_{122}\kappa_{121}\kappa_{120})$;          // $S$ on 2 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 round counter
    $\kappa^i_{63} \dots \kappa^i_0 \longleftarrow \kappa_{127} \dots \kappa_{64}$;          // Extract $i$-th round subkey
end
return $\{K_i\}_{i=0}^r$;
```

B The (Pruned) Fast Walsh Transform

This appendix discusses the Fast Walsh Transform and how its pruned version can be computed efficiently in some cases. Other results on the Walsh-Hadamard matrices are covered by [40], while our pruning approach to the Walsh Transform is inspired by the treatment of the Fast Fourier Transform that was done in [28].

Definition 1. *The recursively-defined matrices*

$$H_1 = (1), \quad H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_{2^m} = \left((-1)^{i \cdot j}\right)_{0 \le i,j < 2^m} = H_2 \otimes H_{2^{m-1}} \in \mathbb{Z}^{2^m \times 2^m} \tag{41}$$

(where $\cdot$ denotes the inner product of binary vectors) are called Hadamard-Sylvester matrices*. Given a vector $\boldsymbol{x} \in \mathbb{Z}^{2^m}$, we define its Walsh or Walsh-Hadamard Transform as the product $\mathcal{W}(\boldsymbol{x}) = H_{2^m}\boldsymbol{x}$:*

$$\mathcal{W}(\boldsymbol{x})_i = \sum_{j=0}^{2^m-1} (-1)^{i \cdot j} x_j \tag{42}$$

If the absolute values of the coordinates of $\boldsymbol{x}$ are bound by the constant M, then the coordinates of its Walsh Transform are bound by $2^m M$.

The Walsh Transform of a vector can be computed efficiently using the result:

Proposition 5. *Given any $\pi \in S_m$, the matrix H_{2^m} can be decomposed as:*

$$H_{2^m} = \prod_{k=1}^{m} \left(I_{2^{m-\pi(k)+1}} \otimes H_2 \otimes I_{2^{\pi(k)}}\right) \tag{43}$$

Proof. This matrix equality is derived from the mixed product property:

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

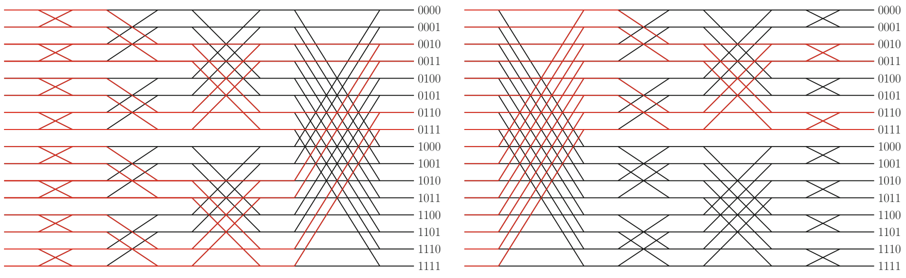

Fig. 10. Two different ways of computing the Pruned Walsh Transform of size $2^4 = 16$ when the leftmost bit of the output coordinates is set to zero and the second rightmost bit is set to one. The algorithm on the left requires 36 integer operations while the one on the right only requires 20.

The product of a vector by the matrix $H^k_{2^m} = I_{2^{m-k+1}} \otimes H_2 \otimes I_{2^k}$ can be computed with 2^m operations. This is represented graphically by 2^{m-1} "butterflies" (a denomination is borrowed from the literature on the FFT) which apply the matrix H_2 on pairs of coordinates.

The Walsh Transform of any vector is thus computable with $m2^m$ additions/substractions. Since we can choose any permutation of the indices k, there are $m!$ different ways of doing this. There are two examples in Fig. 10.

A *pruned* Fast Walsh Transform is any algorithm which aims to efficiently compute a subset of coordinates of $\mathcal{W}(\mathbf{x})$. Here we will consider the strict case in which n binary digits of the output indices are fixed. An approach to pruning the Fast Walsh Transform is working back from the desired outputs and only performing the operations which are strictly necessary. Since the number of required operations depends on the ordering of the matrices of the transform, we want to know which is the optimal ordering.

Proposition 6. *The Pruned Walsh Transform of a vector of length 2^m with n fixed output index bits can be computed with the following number of operations:*

$$2^m + (m-n-1)2^{m-n} \tag{44}$$

Proof. The number of operations in each stage increases from the last stage of the transform to the first. This suggests that the last stages should be those which require the same number of inputs as they do outputs. There are $m-n$ such stages: those corresponding to the matrices $H^k_{2^m}$ where k is one of the bits whose values are *not* fixed. This is true because the output y_{i2^k+j} must be computed iff so must $y_{i2^k+2^{k-1}+j}$. These stages have a cost $(m-n)2^{m-n}$.
The other n stages, which should be performed at the beginning, successively double the number of operations from 2^{m-n} to 2^m. This means that the total cost of the optimized pruned FWT is

$$(m-n)2^{m-n} + \sum_{i=m-n+1}^{m} 2^{i-1} = (m-n)2^{m-n} + 2^m - 2^{m-n} = 2^m + (m-n-1)2^{m-n}$$

An analysis of the computational cost formula shows that, as n increases, the second term decreases and the cost quickly approaches 2^m (instead of $m2^m$).

C Estimates of the Distribution of the Multiple Linear Cryptanalysis Statistic

In a multiple linear attack using M linear approximations and N available plaintexts, the right-key statistic Q_k approximately follows a normal distribution:

$$\begin{aligned}
&Q_k \sim \mathcal{N}(\mu_R, \sigma_R), \text{ where} \\
&\quad \begin{cases} \mu_R = Exp_{\mathcal{D},K}(Q_k) = BM + NExp_K\left(C(K)\right) \\ \sigma_R^2 = Var_{\mathcal{D},K}(Q_k) = 2B^2M + 4BNExp_K\left(C(K)\right) + N^2Var_K\left(C(K)\right) \end{cases} \\
&B = \begin{cases} 1 & \text{if repeated plaintexts are allowed} \\ \frac{2^n-N}{2^n-1} & \text{for distinct known plaintexts} \end{cases}
\end{aligned} \tag{45}$$

The moments of $C(K)$ can be estimated using a set $\mathcal{S}$ of significant linear trails:

$$Exp_K\left(C(K)\right) \simeq \sum_{i=1}^{M}\sum_{\gamma\in\mathcal{S}} \left(c(\alpha_i,\beta_i,\gamma)\right)^2 + M2^{-n} \tag{46}$$

$$Var_K\left(C(K)\right) \simeq 2\sum_{i=1}^{M}\left(\sum_{\gamma\in\mathcal{S}} \left(c(\alpha_i,\beta_i,\gamma)\right)^2 + 2^{-n}\right)^2 \tag{47}$$

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:

$$\frac{1}{B+N2^{-n}}Q_{\tilde{k}} \sim \chi_M^2, \text{ so } \begin{cases} \mu_W = Exp_{\mathcal{D},K}(Q_{\tilde{k}}) = BM + NM2^{-n} \\ \sigma_W^2 = Var_{\mathcal{D},K}(Q_{\tilde{k}}) = 2M(B+N2^{-n})^2 \end{cases} \tag{48}$$

References

1. Abdelraheem, M.A.: Estimating the probabilities of low-weight differential and linear approximations on PRESENT-like ciphers. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 368–382. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37682-5_26
2. Bar-On, A., Dinur, I., Dunkelman, O., Lallemand, V., Keller, N., Tsaban, B.: Cryptanalysis of SP networks with partial non-linear layers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 315–342. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_13
3. Bay, A., Huang, J., Vaudenay, S.: Improved linear cryptanalysis of reduced-round MIBS. In: Yoshida, M., Mouri, K. (eds.) IWSEC 2014. LNCS, vol. 8639, pp. 204–220. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09843-2_16
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 7–11 June 2015, pp. 175:1–175:6. ACM (2015)

5. Beyne, T.: Block cipher invariants as eigenvectors of correlation matrices. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part I. LNCS, vol. 11272, pp. 3–31. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03326-2_1
6. Biham, E., Perle, S.: Conditional linear cryptanalysis - cryptanalysis of DES with less than 242 complexity. IACR Trans. Symmetric Cryptol. **2018**(3), 215–264 (2018). https://doi.org/10.13154/tosc.v2018.i3.215-264
7. Biryukov, A., De Cannière, C., Quisquater, M.: On multiple linear approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_1
8. Blondeau, C., Nyberg, K.: New links between differential and linear cryptanalysis. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 388–404. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_24
9. Blondeau, C., Nyberg, K.: Improved parameter estimates for correlation and capacity deviates in linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2016**(2), 162–191 (2016). https://doi.org/10.13154/tosc.v2016.i2.162-191
10. Bogdanov, A., Geng, H., Wang, M., Wen, L., Collard, B.: Zero-correlation linear cryptanalysis with FFT and improved attacks on ISO standards Camellia and CLEFIA. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 306–323. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_16
11. Bogdanov, A., et al.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
12. Bogdanov, A., Tischhauser, E., Vejre, P.S.: Multivariate profiling of hulls for linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2018**(1), 101–125 (2018). https://doi.org/10.13154/tosc.v2018.i1.101-125
13. Boura, C., Lallemand, V., Naya-Plasencia, M., Suder, V.: Making the impossible possible. J. Cryptol. **31**(1), 101–133 (2018). https://doi.org/10.1007/s00145-016-9251-7
14. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: applications to CLEFIA, Camellia, LBlock and Simon. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 179–199. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_10
15. Canteaut, A., Naya-Plasencia, M., Vayssière, B.: Sieve-in-the-middle: improved MITM attacks. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 222–240. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_13
16. Cho, J.Y.: Linear cryptanalysis of reduced-round PRESENT. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11925-5_21
17. Collard, B., Standaert, F.: A statistical saturation attack against the block cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00862-7_13
18. Collard, B., Standaert, F., Quisquater, J.: Improving the time complexity of Matsui's linear cryptanalysis. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 77–88. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76788-6_7

19. Collard, B., Standaert, F., Quisquater, J.: Experiments on the multiple linear cryptanalysis of reduced round serpent. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 382–397. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71039-4_24
20. Daemen, J., Peeters, M., Assche, G.V., Rijmen, V.: Nessie proposal: the block cipher Noekeon. Nessie submission (2000). http://gro.noekeon.org/
21. Daemen, J., Rijmen, V.: Probability distributions of correlation and differentials in block ciphers. J. Math. Cryptol. **1**(3), 221–242 (2007). https://doi.org/10.1515/JMC.2007.011
22. Datta, N., Ghoshal, A., Mukhopadhyay, D., Patranabis, S., Picek, S., Sadhukhan, R.: TRIFLE. Candidates to the NIST Lightweight competition (2019). https://csrc.nist.gov/projects/lightweight-cryptography/round-1-candidates
23. Etrog, J., Robshaw, M.J.B.: The cryptanalysis of reduced-round SMS4. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 51–65. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_4
24. Florez-Gutierrez, A.: Cryptanalysis of TRIFLE-BC. Official comment to the NIST-LWC forum (2019). https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/official-comments/TRIFLE-official-comment.pdf
25. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional linear cryptanalysis of reduced round serpent. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 203–215. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70500-0_15
26. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional extension of Matsui's algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 209–227. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03317-9_13
27. Hermelin, M., Nyberg, K.: Linear cryptanalysis using multiple linear approximations. Cryptology ePrint Archive, Report 2011/093 (2011). https://eprint.iacr.org/2011/093
28. Hu, Z., Wan, H.: A novel generic fast fourier transform pruning technique and complexity analysis. IEEE Trans. Signal Process. **53**(1), 274–282 (2005). https://doi.org/10.1109/TSP.2004.838925
29. Kim, T.H., Kim, J., Hong, S., Sung, J.: Linear and differential cryptanalysis of reduced SMS4 block cipher. IACR Cryptology ePrint Archive 2008, 281 (2008). http://eprint.iacr.org/2008/281
30. Liu, M., Chen, J.: Improved linear attacks on the Chinese block cipher standard. J. Comput. Sci. Technol. **29**(6), 1123–1133 (2014). https://doi.org/10.1007/s11390-014-1495-9
31. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_33
32. Matsui, M.: The first experimental cryptanalysis of the data encryption standard. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 1–11. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_1
33. Nakahara, J.J., Sepehrdad, P., Zhang, B., Wang, M.: Linear (Hull) and algebraic cryptanalysis of the block cipher PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 58–75. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_5
34. National Institute of Standards and Technology (ed.): "FIPS-46: DataEncryption Standard (DES)" revised as FIPS 46-1:1988, FIPS 46-2:1993, FIPS46-3:1999 (1979). http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

35. Nguyen, P.H., Wu, H., Wang, H.: Improving the Algorithm 2 in multidimensional linear cryptanalysis. In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 61–74. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22497-3_5
36. Nyberg, K.: Linear approximation of block ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0053460
37. Ohkuma, K.: Weak keys of reduced-round PRESENT for linear cryptanalysis. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 249–265. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05445-7_16
38. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. J. Cryptol. **21**(1), 131–147 (2008). https://doi.org/10.1007/s00145-007-9013-7
39. Todo, Y., Aoki, K.: FFT key recovery for integral attack. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 64–81. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12280-9_5
40. Yarlagadda, R.K., Hershey, J.E.: Hadamard Matrix Analysis and Synthesis - With Applications to Communications and Signal/Image Processing. The Springer International Series in Engineering and Computer Science, vol. 383. Springer, Heidelberg (1997). https://doi.org/10.1007/978-1-4615-6313-6
41. Zheng, L., Zhang, S.: FFT-based multidimensional linear attack on PRESENT using the 2-bit-fixed characteristic. Secur. Commun. Netw. **8**(18), 3535–3545 (2015). https://doi.org/10.1002/sec.1278