Optimising Linear Key Recovery Attacks with Affine Walsh Transform Pruning

Antonio Florez Gutierrez

Optimising Linear Key Recovery Attacks with Affine Walsh Transform Pruning

Antonio Flórez-Gutiérrez[0000−0001−7749−8925]

Inria, France
antonio.florez_gutierrez@inria.fr

Abstract. Linear cryptanalysis [25] is one of the main families of key-recovery attacks on block ciphers. Several publications [16, 19] have drawn attention towards the possibility of reducing their time complexity using the fast Walsh transform. These previous contributions ignore the structure of the key recovery rounds, which are treated as arbitrary boolean functions. In this paper, we optimise the time and memory complexities of these algorithms by exploiting zeroes in the Walsh spectra of these functions using a novel affine pruning technique for the Walsh Transform. These new optimisation strategies are then showcased with two application examples: an improved attack on the DES [1] and the first known atttack on 29-round PRESENT-128 [9].

Keywords: Linear cryptanalysis, Key recovery attacks, FFT, Walsh Transform, Pruning, DES, PRESENT

1 Introduction

General Background

Linear Cryptanalysis. Matsui's linear cryptanalysis [25] is a widely studied family of statistical cryptanalysis against block ciphers and other symmetric constructions, and any new proposals are expected to justify their resilience against it. Linear attacks are commonly turned into key recovery attacks, in which a linear distinguisher is extended by one or more rounds by incorporating a key guess. If the attack requires a data complexity of N and l bits of the key are guessed, the time complexity of a standard linear key recovery attack is $\mathcal{O}(N) + \mathcal{O}\left(2^{2l}\right)$ [26].

Fast Key Recovery Algorithms. In the paper by Collard et al. [16], a new key recovery algorithm based on the fast Walsh transform[1] was presented which can sometimes reduce the time complexity of attacks on key-alternating ciphers to $\mathcal{O}(N)+\mathcal{O}\left(l2^{l}\right)$. However, this technique has several limitations, as it complicates common optimisations of previous attacks, most notably key schedule-induced relations. The technique was generalised to multiple rounds by Flórez-Gutiérrez et al. [19], however, many limitations to the algorithm remained.

[1] Called fast Fourier transform / FFT in the paper.

Our Contribution

New Pruned Walsh Transform Algorithm. We describe a new pruning technique for the fast Walsh transform which is effective when the nonzero inputs and the desired outputs lie in (unions of) affine subspaces of $\mathbb{F}_2^n$. The algorithm reduces the computation of the desired outputs to a Walsh transform of smaller size than that of the full transform, thus achieving a large reduction in time complexity.

Reduced Attack Complexity. We next show how this pruned algorithm can be used to optimise linear key recovery attacks. Previous techniques based on the Walsh transform treated the key recovery map as a black box, which meant that the size of the input to this map often became the bottleneck of the algorithm. In our new approach, we see that in some common cases the cipher construction leads to the presence of a lot of zeros in the Walsh spectrum which can be used to improve the key recovery. This, together with information about the key schedule, can greatly reduce the time complexity. We also show how additional zeros can be created by rejecting a small fraction of the data.

Applications

Cryptanalysis of the DES. The first application is a variant of Matsui's attack on the DES [26] in which the last round of the linear approximation has been removed, and is treated as a key recovery round. We improve the data complexity by a factor of $2^{0.5}$ with respect to the best previous result of Biham and Perle [4], but the memory complexity grows due to the larger key guess.

Cryptanalysis of Reduced-round PRESENT. We add a key recovery round to the 28-round attack on PRESENT by Flórez-Gutiérrez et al. [19] with the new pruning techniques, giving the first known attack on 29-round PRESENT-128.

Paper Structure Section 2 covers some techniques and notations which are used in the rest of the paper, as well as the specifications of the applications' target ciphers. Section 3 describes the affine pruning algorithm for the fast Walsh transform from a theoretical perspective. Section 4 provides tools which help the cryptanalyst identify zeroes in the Walsh spectra of the maps which appear in key recovery attacks. Chapter 5 combines the results of the previous two sections by optimising linear key recovery attacks to make use of the cipher structure. Sections 6 and 7 describe the applications to the DES and PRESENT.

2 Preliminaries

2.1 Linear Key Recovery Attacks

Linear approximation. Let $E : \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^n$ be a block cipher. A linear approximation of E is an expression of the form $\langle \alpha, x \rangle + \langle \beta, y \rangle$, where $\langle \cdot, \cdot \rangle$

denotes the dot product in $\mathbb{F}_2^n$. The correlation of the approximation is

$$\mathrm{cor}(\alpha,\beta) = \frac{1}{2^{n+\kappa}} \sum_{K\in\mathbb{F}_2^\kappa} \sum_{x\in\mathbb{F}_2^n} (-1)^{\langle\alpha,x\rangle+\langle\beta,E_K(x)\rangle}. \tag{1}$$

Linear attacks make use of biased linear approximations, that is, of approximations whose correlation is different from zero.

Key recovery attack. We consider a cipher of the form $E' = F \circ E$, and a biased linear approximation $\langle\alpha,x\rangle + \langle\beta,y\rangle$ of E. In a key recovery attack, we guess a part k of K so that the value of the linear approximation can be computed for each pair $(x, y = E'_K(x))$ in a collection $\mathcal{D}$ of N known plaintext-ciphertext pairs. We compute an experimental estimation[2] of the correlation for each guess:

$$\widehat{\mathrm{cor}}(k) = \frac{1}{N} \sum_{(x,y)\in\mathcal{D}} (-1)^{\langle\alpha,x\rangle+\langle\beta,F_k^{-1}(y)\rangle}. \tag{2}$$

If the value of k corresponding to the correct key K appears within the largest $2^{|k|-a}$ in the list[3], we say that the attack achieves an advantage of a [28]. As a rule of thumb, the attack requires $\mathcal{O}\left(\mathrm{cor}(\alpha,\beta)^{-2}\right)$ data pairs to succeed. In this paper we use the more precise model of Blondeau and Nyberg [8].

Multiple linear approximations. It is common for linear attacks to make use of more than one linear approximation [6, 21]. In the PRESENT attack we use the χ^2 multiple linear cryptanalysis statistic:

$$Q(k) = \sum_{i=1}^{M} \widehat{\mathrm{cor}}_i(k)^2, \tag{3}$$

where $\widehat{\mathrm{cor}}_i(k)$ denotes the experimental correlation for the i-th approximation. In a multiple linear attack, the data complexity is determined by the capacity $C = \sum_{i=1}^{M} \mathrm{cor}(\alpha_i,\beta_i)^2$. Detailed models were given by Blondeau and Nyberg [8].

2.2 The Walsh Transform

Definition 1 *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{C}$ be a complex-valued function on $\mathbb{F}_2^n$. We refer to the space of functions of this kind as $\mathbb{CF}_2^n$, which is isomorphic to $\mathbb{C}^{2^n}$. The Hadamard or Walsh transform of f is another map $\widehat{f} : \mathbb{F}_2^n \longrightarrow \mathbb{C}$ given by*[4]

$$\widehat{f}(u) = \sum_{x\in\mathbb{F}_2^n} (-1)^{\langle u,x\rangle} f(x). \tag{4}$$

[2] The notation $\widehat{\mathrm{cor}}$ should not be confused with the Walsh transform $\widehat{f}$.

[3] $|x|$ will denote the number of bits of a binary vector x.

[4] It is common to use the *normalised* Hadamard transform, which is divided by $\sqrt{2^n}$, but for the purposes of this paper we will not use this factor in the definition.

The transform of any vector in $\mathbb{C}\mathbb{F}_2^n$ can be computed efficiently using:

$$\begin{aligned}\widehat{f}(u) = & \sum_{x_{n-2}\in\mathbb{F}_2} \cdots \sum_{x_0\in\mathbb{F}_2} (-1)^{u_{n-2}x_{n-2}+\ldots+u_0x_0} f(0, x_{l-2}, \ldots, x_0) \\ & + (-1)^{u_{n-1}} \sum_{x_{n-2}\in\mathbb{F}_2} \cdots \sum_{x_0\in\mathbb{F}_2} (-1)^{u_{n-2}x_{n-2}+\ldots+u_0x_0} f(1, x_{l-2}, \ldots, x_0).\end{aligned} \tag{5}$$

This formula, in a divide-and-conquer approach, leads to the fast Walsh transform algorithm [18], which has a time complexity of $n2^n$ additions/subtractions.

An associated transformation can be defined for (vectorial) boolean functions:

Definition 2 *Let $g : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$ be any vectorial boolean function. We define its Walsh transform as the map $\widehat{g} : \mathbb{F}_2^n \times \mathbb{F}_2^m \longrightarrow \mathbb{C}$ given by the formula*

$$\widehat{g}(u, v) = \sum_{x\in\mathbb{F}_2^n} (-1)^{\langle u,x\rangle \oplus \langle v,g(x)\rangle}. \tag{6}$$

The coefficients of this map $\widehat{g}$ are often called the Walsh spectrum of g. It is a complex matrix whose columns are the Walsh transforms of $\mathrm{ind}_{g,v} : x \mapsto (-1)^{\langle v,g(x)\rangle}$, complex representations of its linear components $x \mapsto \langle v, g(x)\rangle$. When m=1 we can ignore the second input and assume $v = (1)$ to define $\widehat{g}(u)$. We will also use the Walsh spectrum restricted to a subset X:

Definition 3 *Let $g : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$ be a vectorial boolean function, and $X \subseteq \mathbb{F}_2^n$ a subset of its domain. The Walsh transform of g restricted to X is defined as*

$$\widehat{g_{x\in X}}(u, v) = \sum_{x\in X} (-1)^{\langle u,x\rangle \oplus \langle v,g(x)\rangle}. \tag{7}$$

We define the transform restricted to $Y \subseteq \mathbb{F}_2^m$ as $\widehat{g_{g(x)\in Y}} = \widehat{g_{x\in g^{-1}(Y)}}$.

2.3 Walsh Transform-accelerated Linear Cryptanalysis

FFT-accelerated linear cryptanalysis was introduced by Collard et al. [16]. Flórez-Gutiérrez et al. [19] provided a two-matrix description for instances in which the linear approximation can be separated into two independent parts, such as when key recovery is considered on both the plaintext and the ciphertext sides. We now show a small generalisation of this approach using d-dimensional arrays.

We consider a linear approximation whose value can be expressed as

$$f_0(x) \oplus \underbrace{f_1(x_1 \oplus k_1^O, k_1^I) \oplus \cdots \oplus f_d(x_d \oplus k_d^O, k_d^I)}_{F(X\oplus K^O, K^I)}, \tag{8}$$

where $(x_1, \ldots, x_d) = X$ are separate parts of the plaintext-ciphertext pair x (we denote this by $x \mapsto X$). $(k_1^O, \ldots, k_d^O) = K^O$ is *outer* key material which is xored

directly to x, and $(k_1^I, \ldots, k_d^I) = K^I$ is additional *inner* key material. Our aim is to compute all values of the experimental correlations $\widehat{\mathrm{cor}}(K^O, K^I)$:

$$\begin{aligned} N \cdot \widehat{\mathrm{cor}}(K^O, K^I) &= \sum_{x \in \mathcal{D}} (-1)^{f_0(x) \oplus f_1(x_1 \oplus k_1^O, k_1^I) \oplus \ldots \oplus f_d(x_d \oplus k_d^O, k_d^I)} \\ &= \sum_X (-1)^{F(X \oplus K^O, K^I)} \underbrace{\sum_{\substack{x \in \mathcal{D} \\ x \mapsto X}} (-1)^{f_0(x)}}_{A[X]} \\ &= \frac{1}{2^{|X|}} \sum_Y (-1)^{\langle K^O, Y \rangle} \left[\sum_Z (-1)^{\langle Y, Z \rangle} (-1)^{F(Z, K^I)} \right] \sum_X (-1)^{\langle Y, X \rangle} A[X] \\ &= \frac{1}{2^{|X|}} \sum_Y (-1)^{\langle K^O, Y \rangle} \left(\prod_{i=0}^{d} \widehat{f}_i(y_i, k_i^I) \right) \widehat{A}[Y], \end{aligned} \tag{9}$$

using the convolution theorem. The attack can be performed as follows:

1. For each f_i, precompute $2^{|k_i^I|}$ tables of size $2^{|k_i^O|}$ containing $\widehat{f}_i(\,\cdot\,, k_i^I)$.
2. *Distillation phase:* Construct the $2^{|x_1|} \times \cdots \times 2^{|x_d|}$-dimensional array A.
3. *Analysis phase:*
   (a) Apply the FWT on the array A to obtain $\widehat{A}$. We can consider A is a one-dimensional array of $2^{|X|}$ elements.
   (b) For each value of K^I:
      i. Multiply each entry $\widehat{A}[Y]$ of $\widehat{A}$ by $\prod_{i=0}^{d} \widehat{f}_i(y_i, k_i^I)$.
      ii. Apply another FWT to obtain an array containing $\widehat{\mathrm{cor}}[\,\cdot\,, K^I]$.
4. *Search phase:* Exhaustive search over the rest of the key for the guesses with the largest values of $2^{|X|} N \widehat{\mathrm{cor}}[K^O, K^I]$.

The memory complexity of this algorithm mainly consists $2^{|X|}$ memory registers to store $\widehat{A}$. The time complexity of the distillation phase is $\mathcal{O}(N)$, as each plaintext-ciphertext pair is checked once and discarded. The time complexity of the analysis phase is dominated by the loop on K^I, and consists of $d2^{|K^I|+|K^O|}$ multiplications and $|K^O|2^{|K^I|+|K^O|}$ additions/subtractions. The time complexity of the search phase is given by models such as [8].

Other improvements to this algorithm were proposed by Flórez-Gutiérrez et al. [19], most notably in the case of multiple linear cryptanalysis. By separating the key guesses into groups, it is possible to perform a "complete" key guess k_T (accounting for any dependencies which are induced by the key schedule) while still using the FFT algorithm on different parts of the key guess for each individual approximation. The authors also introduced some Walsh transform pruning techniques for cases in which some external keybits can be deduced from the internal keybits. This paper builds on that improvement.

2.4 DES Specification

The Data Encryption Standard [1] is one of the most widely analysed block ciphers due to its use in the industry. It has a block length of 64 bits and

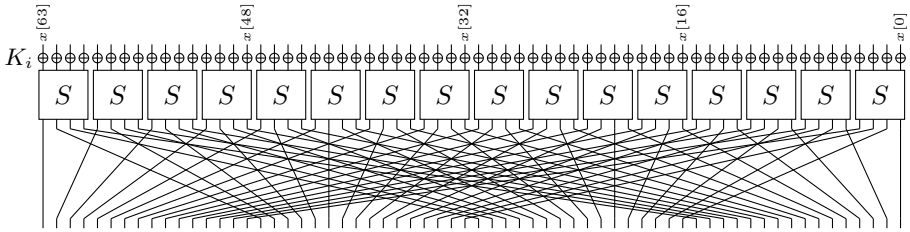

Fig. 1: A round of PRESENT.

a key size of 56 bits, and is a 16-round Feistel network. Each state (L, R) consists of two (left and right) 32-bit parts. The cipher operates as follows:

$(L_0, R_0) \leftarrow IP(P)$;
for $i \leftarrow 1$ **to** 16 **do**
  $L_i \leftarrow R_{i-1}$;
  $R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, K_i)$;
end
$C \leftarrow IP^{-1}(R_{16}, L_{16})$;

where IP is a fixed initial permutation and each K_i is a 48-bit round subkey.

The round function f. First, an expansion function E is applied on the 32-bit input to obtain a 48-bit string. This string is xored with the round subkey, and eight different 6-to-4-bit Sboxes $S_1, \ldots, S_8$ are applied to obtain a 32-bit string. Finally, an output permutation P is applied.

The key schedule. It extracts sixteen 48-bit subkeys $K_1, \ldots, K_{16}$ from the key:

$(C_0, D_0) \leftarrow PC_1(P)$;
for $i \leftarrow 1$ **to** 16 **do**
  $C_i \leftarrow LS_{p(i)}(C_{i-1})$;
  $D_i \leftarrow LS_{p(i)}(D_{i-1})$;
  $K_i \leftarrow PC_2(C_i, D_i)$;
end

where C_i and D_i are 28 bits long, PC_1 and PC_2 are two permutated choices, LS_j is a j bit rotation to the left, and $p(i)$ is either 1 or 2.

Notation. In this paper, $X[j]$ will denote the j-th rightmost (least significant) bit of X, starting from 0. We will also ignore IP, IP^{-1} and PC_1 and denote $P = (L_0, R_0),\ C = (R_{16}, L_{16}),\ K = (C_0, D_0)$ instead.

2.5 PRESENT Specification

PRESENT [9] is a lightweight block cipher which has received substantial attention from cryptanalysts since its inroduction, and is a popular target for linear

cryptanalysis. PRESENT has a block size of 64 and can operate with keys of either 80 or 128 bits. It is a substitution permutation network with 31 rounds:

$X \leftarrow P$;
for $i \leftarrow 1$ **to** 31 **do**
  $X \leftarrow \text{addRoundKey}(X, K_i)$;
  $X \leftarrow \text{sBoxLayer}(X)$;
  $X \leftarrow \text{pLayer}(X)$;
end
$C \leftarrow \text{addRoundKey}(X, K_{32})$;

Sbox Layer. The nonlinear operation consists of the parallel application of 16 identical 4-bit Sboxes on all the nibbles of the state.

Permutation Layer. The linear transformation is a bit permutation, which sends the bit in position i to the position $P(i) = 16i \bmod 63, i \neq 63,\ P(63) = 63$. For its inverse we do the same with $P^{-1}(j) = 4j \bmod 63, j \neq 63,\ P^{-1}(63) = 63$.

Key Schedule. A 64-bit round subkey K_i is xored to the state in each round. These are obtained from the master key K. For 128 bits:

for $i \leftarrow 1$ **to** 31 **do**
  $K_i \leftarrow K[127, \ldots, 64]$;
  $K \leftarrow LS_{61}(K)$;
  $K[127, 126, 125, 124] \leftarrow S(K[127, 126, 125, 124])$;
  $K[123, 122, 121, 120] \leftarrow S(K[123, 122, 121, 120])$;
  $K[66, \ldots, 62] \leftarrow K[66, \ldots, 62] \oplus RC_i$;
end
$K_{32} \leftarrow K[127, \ldots, 64]$;

Notation. We denote the i-th rightmost bit of X starting from 0 by $X[i]$.

3 Affine Pruned Walsh Transform Algorithm

In order to remove unnecesary computations from the algorithm of Section 2.3, we must efficiently compute the Walsh transform when the non-zero inputs or desired outputs are limited to previously-known fixed subsets of $\mathbb{F}_2^n$. An algorithm which obtains the desired outputs with less computations than the "full" fast transform will be called a *pruned* fast Walsh transform algorithm. The case of fixed values for some output position bits was already considered by Flórez-Gutiérrez et al. in [19]. Our algorithms generalise this result.

Definition 4 (Problem statement) *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{C}$ be any vector in $\mathbb{C}\mathbb{F}_2^n$. We assume that lists $L, M \subseteq \mathbb{F}_2^n$ are given, and that $f(x) = 0$ for all $x \in \mathbb{F}_2^n \setminus L$. The aim is to compute $\widehat{f}(y)$ for all $y \in M$ with as few operations as possible.*

3.1 Overview of Previous Results for the One-dimensional DFT

The pruning problem has already been studied for the one-dimensional discrete Fourier transform (DFT), as it arises naturally in some applications. Markel [24] prunes the decimation-in-frequency algorithm for the case in which L consists of the first 2^r, $r < s$ points of the input. Similarly, Skinner [30] prunes the decimation-in-time algorithm for the case in which L consists of the first 2^r points in bit-reversed order. An algorithm limiting both inputs and outputs at the same time was introduced by Sreenivas and Rao [32]. A pruned decimation-in-time algorithm which can compute the outputs in a consecutive (but possibly shifted) frequency window was presented by Nagai [27]. Sorensen and Burrus [31] proposed an alternative *transform decomposition* technique, which maps the nonzero inputs to a series of smaller DFTs, and then combines the results. All these algorithms exhibit similar complexities: evaluating 2^r points of a 2^n point transform costs $\mathcal{O}(r2^n)$. However, an interesting phenomenon was observed by Shousheng and Torkelson [20]: when the subset of outputs M is a *comb* of equidistant points, a smaller complexity of $\mathcal{O}(2^n + r2^r)$ can be achieved.

Alves et al. [2] introduced the first *traceback* pruning method for arbitrary input or output sets. Hu and Wan [22] showed a similar technique and found the average complexity as a function of $n, |L|$ and $|M|$. The overhead computations were reduced by Singh and Srinivasan [29]. Pruning has been recently generalised to mixed-radix and composite length DFTs in works such as [33, 14].

We consider the pruning problem for the Walsh transform or $(2, \ldots, 2)$-dimensional DFT, specifically the case when L and M lie in affine subspaces of $\mathbb{F}_2^n$. Our algorithm takes a different approach to the works mentioned above: we reduce the Walsh transform to one of significantly smaller dimension.

3.2 Walsh Transform Pruning for Affine Sets

We now describe a pruned algorithm which can be used when both the input and output sets of the Walsh transform lie in affine subspaces of $\mathbb{F}_2^n$.

Definition 5 (Affine pruning problem) *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{C}$ be a vector. We are given lists $L, M \subseteq \mathbb{F}_2^n$, vector subspaces $X, U \subseteq \mathbb{F}_2^n$ and vectors $x_0, u_0 \in \mathbb{F}_2^n$ so that $L \subseteq x_0 + X, M \subseteq u_0 + U$, and $f(x) = 0$ for all $x \notin L$. The aim is to compute $\widehat{f}(y)$ for all $y \in M$ with as few operations as possible.*

Example. Consider the Walsh transform of size $16 = 2^4$. The fast transform requires $4 \cdot 2^4 = 64$ additions. Let the lists $L = x_0 + X$ and $M = u_0 + U$ be

$$\begin{aligned} x_0 &= (0,0,1,0), \quad X = \operatorname{span}\{(0,0,0,1),\ (0,1,1,0),\ (1,0,1,0)\}, \\ u_0 &= (0,1,0,0), \quad U = \operatorname{span}\{(0,0,0,1),\ (0,0,1,0),\ (1,1,0,0)\}. \end{aligned}$$

A traceback-based pruning approach as done in [24, 30, 32, 27, 20, 2, 22] is shown in Figure 2. By removing unnecesary computations from the fast Walsh transform, we obtain the desired outputs with 32 additions and subtractions.

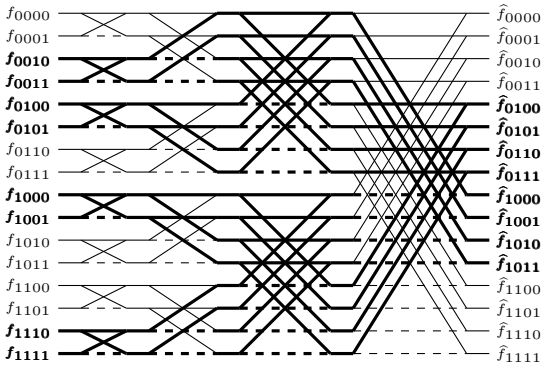

Fig. 2: Using traceback techniques, we can reduce the cost of this Walsh Transform of length 16 from 64 to 32 operations.

Let us examine the expressions for each of the outputs:

$$\begin{aligned}
\widehat{f}_{0100} &= +f_{0010} +f_{0011} -f_{0100} -f_{0101} +f_{1000} +f_{1001} -f_{1110} -f_{1111}\\
\widehat{f}_{0101} &= +f_{0010} -f_{0011} -f_{0100} +f_{0101} +f_{1000} -f_{1001} -f_{1110} +f_{1111}\\
\widehat{f}_{0110} &= -f_{0010} -f_{0011} -f_{0100} -f_{0101} +f_{1000} +f_{1001} +f_{1110} +f_{1111}\\
\widehat{f}_{0111} &= -f_{0010} +f_{0011} -f_{0100} +f_{0101} +f_{1000} -f_{1001} +f_{1110} -f_{1111}\\
\widehat{f}_{1000} &= +f_{0010} +f_{0011} +f_{0100} +f_{0101} -f_{1000} -f_{1001} -f_{1110} -f_{1111}\\
\widehat{f}_{1001} &= +f_{0010} -f_{0011} +f_{0100} -f_{0101} -f_{1000} +f_{1001} -f_{1110} +f_{1111}\\
\widehat{f}_{1010} &= -f_{0010} -f_{0011} +f_{0100} +f_{0101} -f_{1000} -f_{1001} +f_{1110} +f_{1111}\\
\widehat{f}_{1011} &= -f_{0010} +f_{0011} +f_{0100} -f_{0101} -f_{1000} +f_{1001} +f_{1110} -f_{1111}
\end{aligned}$$

We observe the following properties:

$$\widehat{f}_{0100} = -\widehat{f}_{1010},\ \widehat{f}_{0101} = -\widehat{f}_{1011},\ \widehat{f}_{0110} = -\widehat{f}_{1000},\ \widehat{f}_{0111} = -\widehat{f}_{1001}$$

The difference in the indices in each of these pairs is $(1,1,1,0)$, which is orthogonal to X. There are also pairs of inputs which always appear with opposite signs: (f_{0010}, f_{1110}), (f_{0011}, f_{1111}), (f_{0100}, f_{1000}), and (f_{0101}, f_{1001}). In this case, the difference between the indices is $(1,1,0,0)$, which is orthogonal to U.

This suggests an algorithm which subtracts the input pairs from each other at the beginning and duplicates the output pairs at the end, such as the one in Figure 3. With the appropriate intermediate values, the size 2^4 transform is reduced to a size 2^2 transform. The total cost is 24 additions and subtractions.

We now proceed to formalise the "trick", starting with the following lemma:

Lemma 6 *Let $X, U \subseteq \mathbb{F}_2^n$ be vector subspaces of $\mathbb{F}_2^n$. We can define t as*

$$t := \dim\left(\frac{X}{X \cap U^\perp}\right) = \dim\left(\frac{U}{U \cap X^\perp}\right). \tag{10}$$

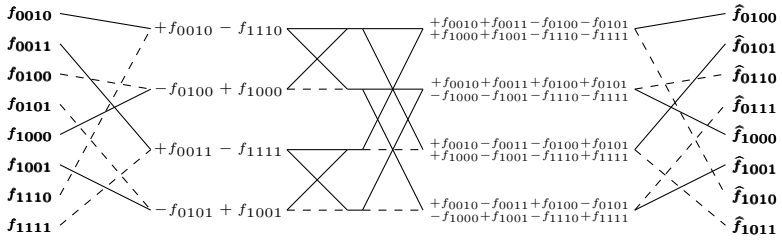

Fig. 3: Organising the inputs and outputs carefully allows us to reduce the cost of the transform to just 24 operations.

There exist isomorphisms $\phi : X/(X \cap U^\perp) \xrightarrow{\cong} \mathbb{F}_2^t$ *and* $\psi : U/(U \cap X^\perp) \xrightarrow{\cong} \mathbb{F}_2^t$ *which preserve the inner product:*

$$\langle y, v\rangle = \langle \phi(y), \psi(v)\rangle \text{ for all } y \in \frac{X}{X \cap U^\perp}, v \in \frac{U}{U \cap X^\perp}. \tag{11}$$

Proof. The equality of the dimensions is a consequence of the dimension formula and the properties of orthogonal spaces. It is also easy to show that the inner product $\langle y, v\rangle$ is well-defined for any $y \in X/(X \cap U^\perp)$ and $v \in U/(U \cap X^\perp)$.

We will construct a pair of "orthonormal" bases starting from two arbitrary bases $\{y_1, \ldots, y_t\}$ and $\{v_1, \ldots, v_t\}$. We will first ensure $\langle y_1, v_j\rangle = \delta_{1j}$ for all j and $\langle y_i, v_1\rangle = \delta_{i1}$ for all i, and then work recursively. There is at least one j so that $\langle y_1, v_j\rangle = 1$ (if $y_1 \perp v_j$ for all j, we'd have $y_1 \perp U$, $y_1 = 0$). We swap the v_j so that $\langle y_1, v_1\rangle = 1$. We then modify both bases as follows:

$$\begin{aligned} y_1^{new} &= y_1 & y_i^{new} &= y_i + \langle y_i, v_1\rangle y_1 \text{ for all } i \neq 1 \\ v_1^{new} &= v_1 & v_j^{new} &= v_j + \langle y_1, v_j\rangle v_1 \text{ for all } j \neq 1 \end{aligned}$$

These new bases have the following properties:

$$\begin{aligned} \langle y_1^{new}, v_1^{new}\rangle &= \langle y_1, v_1\rangle &&= 1 \\ \langle y_1^{new}, v_j^{new}\rangle &= \langle y_1, v_j\rangle + \langle y_1, v_j\rangle\langle y_1, v_1\rangle &&= 0 \text{ for all } j \neq 1 \\ \langle y_i^{new}, v_1^{new}\rangle &= \langle y_i, v_1\rangle + \langle y_i, v_1\rangle\langle y_1, v_1\rangle &&= 0 \text{ for all } i \neq 1 \end{aligned}$$

This process can be iterated on the rest of the elements until we obtain a pair of bases $\{y_1, \ldots, y_t\}$ and $\{v_1, \ldots, v_t\}$ which verify $\langle y_i, v_j\rangle = \delta_{ij}$. We obtain ϕ and ψ by mapping these bases to the standard basis of $\mathbb{F}_2^t$.

This lemma provides the basis for the following result and Algorithm 1:

Proposition 7 *Let* $\widehat{f}$ *be the Walsh transform of* $f \in \mathbb{CF}_2^n$*. We are given lists* $L \subseteq x_0 + X \subseteq \mathbb{F}_2^n$ *and* $M \subseteq u_0 + U \subseteq \mathbb{F}_2^n$*, where* $x_0 + X$ *and* $u_0 + U$ *are affine subspaces, and assume* $f(x) = 0$ *for all* $x \notin L$*. Let* $t = \dim\big(X/(X \cap U^\perp)\big) = \dim\big(U/(U \cap X^\perp)\big)$*. There is an algorithm which computes* $\widehat{f}(u)$ *for all* $u \in M$ *with* $|L| + t2^t + |M|$ *additions using* 2^t *memory registers.*

Algorithm 1: Fast Walsh transform pruned to affine subspaces

Parameters: $L \subseteq x_0 + X \subseteq \mathbb{F}_2^n, M \subseteq u_0 + U \subseteq \mathbb{F}_2^n$, ($X, U$ subspaces).
Input: $f : L \longrightarrow \mathbb{C}$
Output: $\widehat{f} : M \longrightarrow \mathbb{C}$
$\mathcal{B}_X = \{y_1, \ldots, y_t\} \leftarrow \text{GetBasis}(X/(X \cap U^\perp))$;
$\mathcal{B}_U = \{v_1, \ldots, v_t\} \leftarrow \text{GetBasis}(U/(U \cap X^\perp))$;
for $k \leftarrow 1$ **to** $t-1$ **do** `// Generate "good" bases`
  while $\langle y_k, v_k \rangle = 0$ **do** $(v_k, v_{k+1}, \ldots, v_{t-1}, v_t) \leftarrow (v_{k+1}, v_{k+2}, \ldots, v_t, v_k)$;
  for $i \leftarrow k+1$ **to** t **do** $y_i \leftarrow y_i + \langle y_i, v_k \rangle y_k$;
  for $j \leftarrow k+1$ **to** t **do** $v_j \leftarrow v_j + \langle y_k, v_j \rangle v_k$;
end
let $g : \mathbb{F}_2^t \longrightarrow \mathbb{C},\ g(y) = 0\ \forall y \in \mathbb{F}_2^t$;
foreach $x \in L$ **do** `// Absorb the nonzero inputs`
  $(i_1, \ldots, i_t) \leftarrow \text{GetCoordinates}(\overline{x - x_0}, \mathcal{B}_X)$;
  $g(i_1, \ldots, i_t) \leftarrow g(i_1, \ldots, i_t) + (-1)^{\langle x - x_0, u_0 \rangle} f(x)$;
end
$g \leftarrow \text{FWT}(g)$; `// Fast Walsh transform of size 2^t`
foreach $u \in M$ **do** `// Generate the desired outputs`
  $(j_1, \ldots, j_t) \leftarrow \text{GetCoordinates}(\overline{u - u_0}, \mathcal{B}_U)$;
  $\widehat{f}(u) \leftarrow (-1)^{\langle x_0, u \rangle} g(j_1, \ldots, j_t)$;
end
return $\widehat{f}$

Proof. Let $u = u_0 + u',\ u' \in U$ be one of the desired outputs.

$$\begin{aligned}\widehat{f}(u) &= \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle u, x \rangle} f(x) = \sum_{x' \in X} (-1)^{\langle u, x_0 \rangle + \langle u', x' \rangle + \langle u_0, x' \rangle} f(x_0 + x') \\ &= (-1)^{\langle u, x_0 \rangle} \sum_{y \in X/(X \cap U^\perp)} (-1)^{\langle u', y \rangle} \sum_{x' \in y} (-1)^{\langle u_0, x' \rangle} f(x_0 + x'),\end{aligned}$$

where $x \in y'$ means that x is a representative of the class y, in other words, $y = x' + (X \cap U^\perp)$. This suggests the following algorithm:

1. For each $y \in X/(X \cap U^\perp)$, compute $g(y) = \sum_{x' \in y} (-1)^{\langle u_0, x' \rangle} f(x_0 + x')$, forming an array g of length 2^t. We go over all $x \in L$, compute $x' = x - x_0$, and add $f(x)$ to the bin corresponding to the class of x'. This costs at most $|L|$ additions. We do not need to store any entries of f in memory.
2. We apply the fast Walsh transform on g with $t2^t$ additions. The result is a vector $\widehat{g}$ which contains, for each $v \in V \in U/(U \cap X^\perp)$:

$$\widehat{g}(v) = \sum_{y \in X/(X \cap U^\perp)} (-1)^{\langle v, y \rangle} \sum_{x' \in y} (-1)^{\langle u_0, x' \rangle} f(x_0 + x').$$

   Lemma 6 justifies the validity of this step.
3. For each output $u \in M$, separate $u = u_0 + u'$, and sign-swap the entry of $\widehat{g}$ indexed under the class of u' according to $\langle x_0, u \rangle$ to obtain $\widehat{f}(u)$. The total

cost is at most $|M|$. Each output can be queried individually, and we can even store the vector $\widehat{g}$ and query any output in $\mathcal{O}(1)$ afterwards.

Example. We return to the example to illustrate how the algorithm of Figure 3 is justified by proposition 7. Indeed, $U \cap X^{\perp} = X^{\perp} = \operatorname{span}\{(1,1,1,0)\}$ and $X \cap U^{\perp} = U^{\perp} = \operatorname{span}\{(1,1,0,0)\}$, so $t = 2$ and the transform reduces to one of size 2^2. The inputs and outputs of the reduced transform correspond to the bases $(\overline{(0,1,1,0)}, \overline{(0,0,0,1)})$ of $X/(X \cap U^{\perp})$ and $(\overline{(0,0,1,0)}, \overline{(0,0,0,1)})$ of $U/(U \cap X^{\perp})$.

3.3 A Small Generalisation

We have described a pruned fast Walsh transform algorithm which is effective when the inputs and/or outputs are restricted to affine subspaces of small dimension. We have also shown that the time complexity doesn't just depend on the dimensions of these subspaces, but also on their orthogonality. The next natural step is to look into algorithms for arbitrary subsets of $\mathbb{F}_2^n$.

We can find the smallest subspaces which cover all inputs and outputs by choosing random $x_0 \in L$ and $u_0 \in M$ and picking $X = \operatorname{span}(\{x - x_0\}_{x \in L})$ and $U = \operatorname{span}(\{u - u_0\}_{u \in M})$. However, if $|L|, |M| \gg n$ it is very likely that $X = U = \mathbb{F}_2^n$, and we just obtain the traditional fast Walsh transform algorithm. This is the case in the applications later in the paper.

In these applications, however, the nonzero coefficients can be covered by a small amount of low dimension subspaces. We assume that we separate the lists L and M as disjoint unions $L = L_1 \cup \cdots \cup L_p$ and $M = M_1 \cup \cdots \cup M_q$. Let's also assume that there exist $x_0^1, \ldots, x_0^p$ and $u_0^1, \ldots, u_0^q$, as well as $X_1, \ldots, X_p$ and $U_1, \ldots, U_q$ so that $L_i \subseteq x_0^i + X_i$ and $M_j \subseteq u_0^j + U_j$. Although the list families $\{L_i\}$ and $\{M_j\}$ are disjoint, the affine subspace families $\{x_0^i + X_i\}$ and $\{u_0^j + U_j\}$ need not be disjoint. Because of the linearity of the Walsh transform, we can compute the transform for each pair (L_i, M_j) separately, and combine the results at the end. Let $t_{ij} := \dim\left(X_i/(X_i \cap U_j^{\perp})\right)$. The time complexity is:

$$q|L| + \sum_{i=1}^{p} \sum_{j=1}^{q} t_{ij} \cdot 2^{t_{ij}} + p|M| \text{ additions/subtractions.} \tag{12}$$

4 Zeros in the Walsh Spectra of SPN Constructions

This section adapts some previously-known results on the Walsh transform (see [13]) to quickly identify zeroes in the Walsh spectra of block cipher constructions which alternate a bricklayer nonlinear map and a linear transformation, such as Substitution Permutation Networks. We also illustrate how in some cases slightly modifying to the key recovery map so that it rejects some plaintexts can drastically reduce the number of nonzero coefficients.

Lemma 8 *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$, $f(x) = Lx \oplus c$, where $L \in \mathrm{GL}(\mathbb{F}_2^n, \mathbb{F}_2^m)$ is a linear map and $c \in \mathbb{F}_2^m$ is a constant. Then*

$$\widehat{f}(u,v) = \begin{cases} 0 \text{ if } u \neq L^T v \\ (-1)^{\langle v,c\rangle} 2^n \text{ if } u = L^T v \end{cases} \text{ for all } u \in \mathbb{F}_2^n, v \in \mathbb{F}_2^m. \tag{13}$$

Lemma 9 *Let $f_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2^{m_i}$, $i = 1, \ldots, d$ be d vectorial boolean functions. We consider the* bricklayer map *$\mathbf{F} : \mathbb{F}_2^{\sum_i n_i} \longrightarrow \mathbb{F}_2^{\sum_i m_i}$, which is obtained by concatenation, $\mathbf{F}(x_1, \ldots, x_d) = (f_1(x_1), \ldots, f_d(x_d))$. Then, for any $(u_1, \ldots, u_d) \in \mathbb{F}_2^{n_1} \times \ldots \times \mathbb{F}_2^{n_d}$ and $(v_1, \ldots, v_d) \in \mathbb{F}_2^{m_1} \times \ldots \times \mathbb{F}_2^{m_d}$, we have*

$$\widehat{\mathbf{F}}((u_1, \ldots, u_d), (v_1, \ldots, v_d)) = \prod_{i=1}^{d} \widehat{f_i}(u_i, v_i). \tag{14}$$

Note that if $m_i = 1$ and f_i is balanced, then $\widehat{f_i}(u_i, v_i) = \begin{cases} \widehat{f_i}(u_i) & \text{if } v_i = 1 \\ 0 & \text{if } u_i \neq 0, v_i = 0 \\ 2^{n_i} & \text{if } u_i = 0, v_i = 0 \end{cases}$.

Lemma 10 *Let $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^l$ and $g : \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^m$ be vectorial boolean functions. Let $X \subseteq \mathbb{F}_2^n$, $Z \subseteq \mathbb{F}_2^l$ and $Y \subseteq \mathbb{F}_2^m$ be subsets. We have*

$$2^l \widehat{g \circ f}(u,v) = \sum_{w \in \mathbb{F}_2^l} \widehat{f}(u,w) \cdot \widehat{g}(w,v) \tag{15}$$

$$2^l \widehat{g \circ f}_{x \in X}(u,v) = \sum_{w \in \mathbb{F}_2^l} \widehat{f_{x \in X}}(u,w) \cdot \widehat{g}(w,v) \tag{16}$$

$$2^l \widehat{g \circ f}_{g \circ f(x) \in Y}(u,v) = \sum_{w \in \mathbb{F}_2^l} \widehat{f}(u,w) \cdot \widehat{g_{g(z) \in Y}}(w,v) \tag{17}$$

$$2^l \widehat{g \circ f}_{f(x) \in Z}(u,v) = \sum_{w \in \mathbb{F}_2^l} \widehat{f_{f(x) \in Z}}(u,w) \widehat{g}(w,v) = \sum_{w \in \mathbb{F}_2^l} \widehat{f}(u,w) \widehat{g_{z \in Z}}(w,v) \tag{18}$$

Using these results, we can often obtain compact formulas for the Walsh coefficients of some key recovery maps, such as the following:

Proposition 11 *Let $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^{l_i}$ be d balanced vectorial boolean functions, let $L : \mathbb{F}_2^{\sum_i l_i} \longrightarrow \mathbb{F}_2^l$ be a linear map, and let $g : \mathbb{F}_2^l \longrightarrow \mathbb{F}_2$ be a boolean function. In the applications, the f_i will be some Sboxes with possibly truncated outputs, L will be a truncation of the linear layer, and g will be a linear combination of outputs of an Sbox layer. We also consider a subset $Z \subseteq \mathbb{F}_2^l$. We consider the composition $h = g \circ L \circ \mathbf{F}$, where $\mathbf{F}$ is the bricklayer function $\mathbf{F}(x_1, \ldots, x_d) = (f_1(x_1), \ldots, f_d(x_d))$. The Walsh coefficients of h can be obtained through the*

following formula:

$$\widehat{h}_{L(\mathbf{F}(x))\in Z}(u_1,\ldots,u_d) = \frac{1}{2^l} \underbrace{\sum_{w_1\in\mathbb{F}_2^{l_1}} \cdots \sum_{w_d\in\mathbb{F}_2^{l_d}} \sum_{v\in\mathbb{F}_2^l}}_{\substack{w_i=0 \text{ if } u_i=0 \\ (w_1,\ldots,w_d)=L^t v}} \prod_{i=1}^{d} \widehat{f}_i(u_i,w_i)\widehat{g_{z\in Z}}(v). \quad (19)$$

Proof. We use Lemma 10 to write the Walsh coefficients of h as

$$\widehat{h}_{L(\mathbf{F}(x))\in Z}(u_1,\ldots,u_d) = \frac{1}{2^{\sum_i l_i + l}} \sum_{w\in\mathbb{F}_2^{\sum_i l_i}} \sum_{v\in\mathbb{F}_2^l} \widehat{\mathbf{F}}(u,w)\widehat{L}(w,v)\widehat{g_{z\in Z}}(v).$$

According to Lemma 8, $\widehat{L}(w,v) \neq 0$ if and only if $w = L^t v$, in which case $\widehat{L}(w,v) = 2^{\sum_i l_i}$. This means we only have to consider the sums over the w_i for which an appropriate v exists, and vice versa. Furthermore, we can write $\widehat{\mathbf{F}}(u,w) = \prod_{i=1}^{d} \widehat{f}_i(u_i,w_i)$ according to Lemma 9. Since $\widehat{f}_i(0,w_i) = 0$ if $w_i \neq 0$, we can assume $w_i = 0$ for the i for which $u_i = 0$.

In particular, for the case in which all $l_i = 1$:

Corollary 12 *Let $f_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2$ be l boolean functions and $g : \mathbb{F}_2^l \longrightarrow \mathbb{F}_2$. We consider $h(x_1,\ldots,x_l) = g(f_1(x_1),\ldots,f_d(x_l))$ and the subset $Z \subseteq \mathbb{F}_2^d$. Then*

$$\widehat{h_{\mathbf{F}(x)\in Z}}(u_1,\ldots,u_l) = \frac{2^{\sum_{i,u_i=0} n_i}}{2^l} \widehat{g_{z\in Z}}(w(u_1,\ldots,u_l)) \prod_{i,u_i\neq 0} \widehat{f}_i(u_i),$$

where $w(u_1,\ldots,u_l)_i = \begin{cases} 0 & \text{if } u_i = 0 \\ 1 & \text{if } u_i \neq 0 \end{cases}$.

We'll show how the previous result describes $\widehat{h_{f(x)\in Z}}$ and its zeroes in a compact manner. We first look at $\widehat{g_{z\in Z}}$. Given any $w \in \mathbb{F}_2^l$ so that $\widehat{g_{z\in Z}}(w) = 0$, we can deduce that $\widehat{h_{\mathbf{F}(x)\in Z}}(u_1,\ldots,u_l) = 0$ for all $(u_1,\ldots,u_l)$ so that $w = w(u_1,\ldots,u_l)$. Furthermore, for the $(u_1,\ldots,u_l)$ for which $\widehat{g_{z\in Z}}(w(u_1,\ldots,u_l)) \neq 0$, the Walsh coefficient $\widehat{h_{\mathbf{F}(x)\in Z}}(u_1,\ldots,u_l)$ can be written as the product of $\widehat{g_{z\in Z}}(w(u_1,\ldots,u_l))$ and the $\widehat{f}_i(u_i)$ corresponding to each $u_i \neq 0$.

An interesting situation appears when $\widehat{g_{z\in Z}}(1,\ldots,1) = 0$. Given $(u_1,\ldots,u_l)$ so that $u_i \neq 0$ for all i, we know that $\widehat{h_{\mathbf{F}(x)\in Z}}(u_1,\ldots,u_l) = 0$, and any nonzero Walsh coefficient must verify $u_i = 0$ for at least one i. As a result, the nonzero Walsh coefficients can be separated into l vector subspaces U_i of dimensions $\sum_{j\neq i} n_j - n_i$. Each U_i is determined by the n_i linear equations $u_i = 0$.

When $\widehat{g}(1\ldots1) = 0$, we obtain this decomposition without any modifications to the key recovery map. When $\widehat{g}(1\ldots1) \neq 0$, we would like to choose some large $Z \subseteq \mathbb{F}_2^d$ so that $\widehat{g_{z\in Z}}(1,\ldots,1) = 0$. We can use the following result:

Proposition 13 *Let $g : \mathbb{F}_2^l \longrightarrow \mathbb{F}_2$ be a map for which $\widehat{g}(1 \dots 1) = a \neq 0$. There exists $Z \subseteq \mathbb{F}_2^l$ with $|Z| = 2^l - |a|$ so that $\widehat{g_{z \in Z}}(1 \dots 1) = 0$.*

We have substituted the key recovery map, which normally takes values ± 1 depending on the linear approximation, for a modified map which is zero when $\mathbf{F}(x) \notin Z$. From the perspective of the attack, we are rejecting the plaintext-ciphertext pairs for which the input of g is not in Z. Assuming independence, the resulting attack has the same parameters except for the data complexity, which increases by a factor of $2^l/|Z|$ to compensate the rejected plaintexts.

These results describe *static* key recovery maps $F(X \oplus K^O)$ without inner key guesses. We must also consider maps of the form $F(X \oplus K^O, K^I)$. When all $l_i = 1$, the xoring of a round subkey between rounds only changes the Walsh coefficient signs, and the positions of the zero coefficients remain unaltered:

Corollary 14 *Let $f_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2$ be l boolean functions, $g : \mathbb{F}_2^l \longrightarrow \mathbb{F}_2$, and let $k \in \mathbb{F}_2^l$ be a fixed parameter. We consider the parametric function $h(x_1, \dots, x_l; k) = g((f_1(x_1), \dots, f_l(x_l)) \oplus k)$ and the subset $Z \subseteq \mathbb{F}_2^l$. Then*

$$\widehat{h(\cdot, k)}_{\mathbf{F}(x) \oplus k \in Z}(u_1, \dots, u_l) = (-1)^{\langle k, w(u_1, \dots, u_l) \rangle} \widehat{h(\cdot, 0)}_{\mathbf{F}(x) \in Z}(u_1, \dots, u_l).$$

5 Optimised Attack Algorithm

We now provide a linear key recovery algorithm which makes use of the affine pruned Walsh transform. We assume that the target linear approximation is of the form $f_0(x) + f(X \oplus K^O, K^I)$, but it also applies to key recovery maps with several parts. We will also make some redundancy assumptions:

- The parts of the plaintext-ciphertext pair X which are xored with the outer key guess K^O lie in an affine subspace of the form $x_0 + Y \subseteq \mathbb{F}_2^{|K^O|}$.
- The nonzero Walsh coefficients of $F(\cdot, 0)$ lie in the union of l affine subspaces $u_0^i + U_i \subseteq \mathbb{F}_2^{|K^O|}$. We denote the number of nonzero coefficients in $u_0^i + U_i$ by $|L_i|$. We also assume that the nonzero Walsh coefficients of $F(\cdot, K^I)$ occupy the same subspaces. If the latter is not true, each value of K^I must be treated separately, and the cost of the analysis phase is multiplied by $2^{|K^I|}$.
- Given the key schedule of the cipher, for a given guess of K^I, the possible values of K^O lie within an affine subspace of the form $v_0^{K^I} + V_{K^I} \subseteq \mathbb{F}_2^{|K^O|}$.

We denote the dimensions of the relevant quotient spaces for the first Walsh transform as $t_i = \dim\left(Y/(Y \cap U_i^\perp)\right)$. For the last set of Walsh transforms, we assume that these dimensions are constant for all the K^I, that is $r_i = \dim\left(U_i/(U_i \cap V_{K^I}^\perp)\right)$ for all $K^I \in \mathbb{F}_2^{|K^I|}$. This assumption is not necessary but it simplifies the complexity calculation.

The broad idea of the attack procedure is to compute $\widehat{\mathrm{cor}}(\cdot, K^I)$ as the sum of l linear transformations of A. Each linear operation corresponds to the part of the Walsh spectrum of F which lies in the affine subspace $u_0^i + U_i \subseteq \mathbb{F}_2^{|K^O|}$. The full attack algorithm is the following:

1. *Distillation phase:* We can merge the first step of the first pruned Walsh transform into the distillation phase to save time and memory. Depending on the relative sizes of N and $2^{\dim(Y)}$, we have two options:
   - Perform the distillation phase as usual (compute A in full) and compute the first step of the pruned Walsh transform algorithm for each of the l pruned Walsh transforms separately to obtain l tables g_i of lengths 2^{t_i}. We note that we only need $2^{\dim(Y)}$ counters to store A. The cost of this operation is $N + l \cdot 2^{\dim(Y)}$ additions.
   - We can instead construct l distilled tables g_i directly, without building the intermediate array A. The cost of this operation is $l \cdot N$ additions.

   Both options require $\sum_{i=1}^{l} 2^{t_i}$ registers to store the resulting distilled data.
2. *Analysis phase:* We also save time and memory by mixing the last step of the first Walsh transform, the eigenvalue multiplication step, and the first step of the second set of Walsh transforms.
   (a) *First Walsh transform:* Perform the (standard) fast Walsh transform on each of the arrays g_i to obtain l arrays $\widehat{g_i}$. The time complexity of this operation is $\sum_{i=1}^{l} t_i 2^{t_i}$ additions.
   (b) *Walsh spectrum multiplication:* This step and the next are repeated for each guess of K^I. Inside each subspace $u_0^i + U_i$, we go over all the nonzero Walsh coefficients. For the nonzero coefficients which belong to more than one subspace, we must only consider them in one of these subspaces. For each coefficient, we fetch the appropriate entry of $\widehat{g_i}$ and multiply it by the coefficient $\widehat{F}(u_0^i + u', K^I)$. The result is then added to the appropriate coordinate of an array h of length 2^{r_i}. This step uses $2^{|K^I|} \sum_{i=1}^{l} |L_i|$ products and additions and requires $\sum_{i=1}^{l} 2^{r_i}$ additional memory registers (assuming we can reuse the same memory from one K^I to the next). If Corollary 14 applies, it is possible to achieve further savings by performing the multiplication step a single time.
   (c) *Second set of Walsh transforms:* We perform the fast Walsh transform on each of the h_i to obtain $\widehat{h_i}$, at a cost of $\sum_{i=1}^{l} r_i 2^{r_i}$ additions.
   (d) *Unfolding step:* For each guess of K^O, we compute $\widehat{\mathrm{cor}}(K^O, K^I)$ by adding l values (with appropriate signs), one from each of the $\widehat{h_i}$. This costs $l 2^{|K^I|} 2^{\dim(V)}$ additions.

By adding up the cost of each step we find that the total time and memory complexity of the algorithm is, after removing terms of lower order:

$$\underbrace{2^{|K^I|}}_{*} Nl + \underbrace{2^{|K^I|}}_{*} \sum_{i=1}^{l} t_i 2^{t_i} + \underbrace{2^{|K^I|}}_{**} \sum_{i=1}^{l} L_i + 2^{|K^I|} \sum_{i=1}^{l} r_i 2^{r_i} \text{ additions,} \tag{20}$$

$$\underbrace{2^{|K^I|}}_{**} \sum_{i=1}^{l} L_i \text{ products, and} \tag{21}$$

$$\sum_{i=1}^{l} 2^{t_i} + \sum_{i=1}^{l} 2^{r_i} \text{ registers,} \tag{22}$$

where the factors indicated by $*$ can be removed when the nonzero Walsh coefficients of $F(\cdot, K^I)$ occupy the same subspaces $u_0^i + U_i$ independently from K^I, and the factors with $**$ can be removed when the Walsh coefficients of the different $\widehat{F}(\cdot, K^I)$ only differ by sign as in Corollary 14.

6 Application to the DES

As an application example, we present a variant of Matsui's linear attack [25, 26] on the DES [1]. This variant has lower data complexity ($2^{41.5}$ vs. 2^{43}), but has a larger memory complexity ($2^{38.75}$ vs. $2^{26.00}$) due to the larger key guess.

We use a 13-round linear approximation identical to the 14-round linear approximation used in [26], but with the last round removed. This increases the correlation from $-2^{-19.75}$ to $2^{-19.07}$. The input mask is $(\mathtt{00000000}, \mathtt{01040080})$ at (L_1, R_1) and the output mask is $(\mathtt{21040080}, \mathtt{00000000})$ at (L_{14}, R_{14}).

Figure 4 indicates the active keybits in the key recovery in rounds 1, 15, and 16. There are 40 active keybits in total: 3 are active in round 1, one is active in round 15, 28 are active in round 16, 3 are active in both rounds 1 and 16, and 5 are active in both rounds 15 and 16. All active keybits are represented as part of K, after applying the appropriate bit rotation. There are 43 active plaintext/ciphertext bits (four of which are duplicated before the key addition because of the expansion map) and 40 active keybits (eight of which are used twice). An attack using the same version of Algorithm 2 as [26] would have a time complexity of $O(N)+2^{43+40} \simeq 2^{83}$ operations. An attack based on the FFT without any kind of optimisation [17] would require $O(N) + 48 \cdot 2^{48}$ operations.

6.1 The Walsh Spectrum of the Key Recovery Map

Figure 5 shows the full key recovery map for the attack, including all the key material. In other words, it shows how the linear approximation is computed from the plaintext, ciphertext, and key. Our aim is to identify the zeroes in this function's Walsh spectrum. We note that all key material is xored to the plaintext/ciphertext, and that there are seven plaintext/ciphertext bits which are xored at the end and can be considered separately as the term f_0. The rest of the map consists of two independent parts if we ignore the key schedule: one corresponds to the first round and the other corresponds to the last two rounds.

In the case of the map for the first round, which we will denote by f_1, we can see that it consists of the application of S_5 and the xoring of three of its output bits. If we look at the Walsh spectrum of S_5, we can see that for the output $y_1 \oplus y_2 \oplus y_3$ we have 50 nonzero coefficients out of the total 64.

The map for the last two rounds f_2 is a little more complex. It is the composition of three maps: the first is an $\mathbb{F}_2^{42} \to \mathbb{F}_2^{12}$ map consisting of the application of the six active Sboxes in round 16 (selecting a single output bit for each), as well as the identity on the six active bits on the left part of the ciphertext. We then apply a linear $\mathbb{F}_2^{12} \to \mathbb{F}_2^6$ map which xors the outputs of the six sboxes into the ciphertext material. Finally, we apply S_5 and xor the four outputs. If we

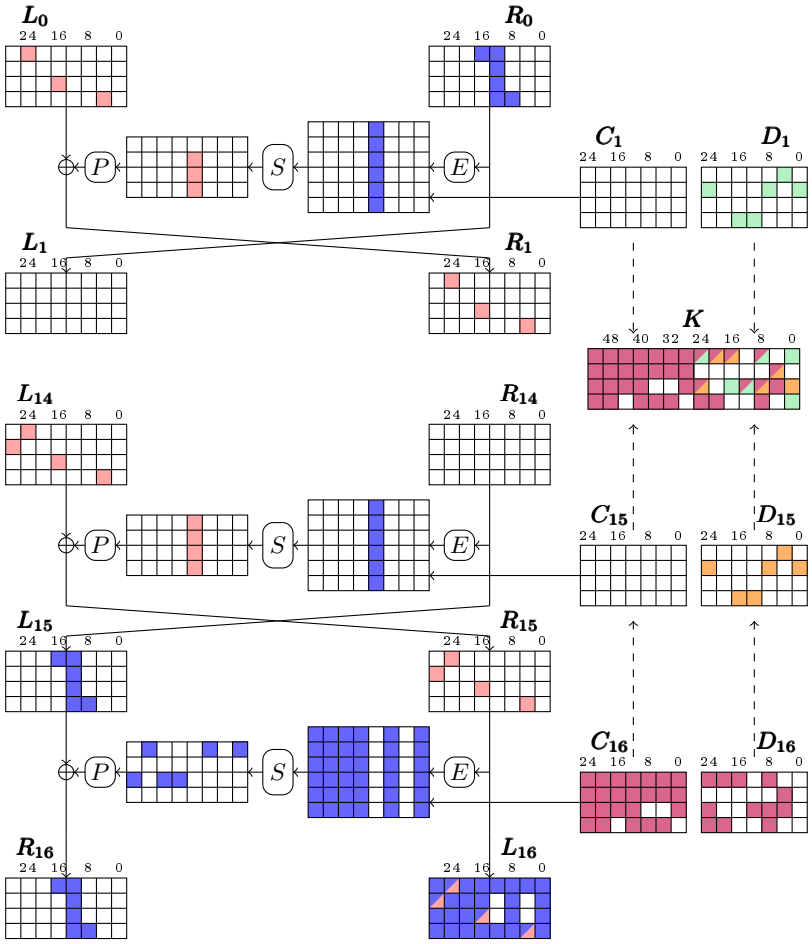

Fig. 4: Key recovery in rounds 1, 15 and 16 of the DES. States are represented as divided into nibbles, except for those before the S-box layer which are divided in groups of 6 bits. The least significant bit is the one on the upper right. ■ represents a bit which appears linearly in the linear approximation, while ■ represents any other active (nonlinear) bit. ■, ■, and ■ represent keybits which are active in rounds 1, 15 and 16, respectively.

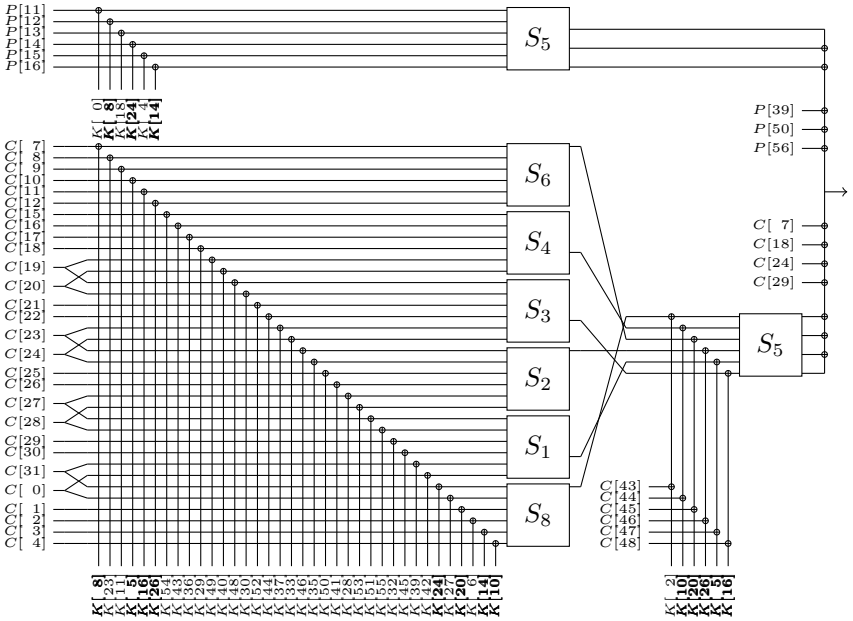

Fig. 5: Schematic of the key recovery map for the DES attack.

Table 1: Part of the Walsh spectrum of S_5: $\widehat{S_5}(\cdot, \mathtt{F})$.

00	0	08	8	10	−40	18	−8	20	0	28	0	30	8	38	0
01	0	09	−8	11	8	19	−8	21	0	29	0	31	8	39	0
02	−8	0A	0	12	0	1A	0	22	−24	2A	8	32	0	3A	−8
03	−8	0B	0	13	0	1B	0	23	−8	2B	8	33	0	3B	8
04	0	0C	−8	14	0	1C	0	24	0	2C	0	34	0	3C	8
05	8	0D	0	15	8	1D	8	25	−8	2D	−8	35	8	3D	0
06	0	0E	−8	16	0	1E	0	26	0	2E	0	36	0	3E	8
07	−8	0F	0	17	8	1F	−8	27	−8	2F	−8	37	−8	3F	0

look at the Walsh spectrum of S_5 with output mask F (Table 1), we note that there are 32 zeros, one of them corresponding to the input mask 3F.

We consider the coefficients $\widehat{f_2}(u_0, \ldots, u_5, u_6)$, where u_6 corresponds to the six active bits in the left half of the ciphertext. The mask u_6 will be determined by the rest of parts of the mask, as from Corollary 12 we can deduce that

$$\widehat{f_2}(u_0, \ldots, u_5, u_6) \neq 0 \Longrightarrow (u_6[i] = 1 \longleftrightarrow u_i \neq 0 \text{ for all } i). \tag{23}$$

Furthermore, the following expression for the Walsh coefficient can be deduced:

$$\begin{aligned}&\widehat{f_2}(u_0,\ldots,u_5,\mathrm{ind}(u_0,\ldots,u_5)) = \\ &\quad \widehat{S_8}(u_0,\mathtt{1})\widehat{S_4}(u_1,\mathtt{4})\widehat{S_6}(u_2,\mathtt{1})\widehat{S_2}(u_3,\mathtt{1})\widehat{S_1}(u_4,\mathtt{4})\widehat{S_3}(u_5,\mathtt{4})\widehat{S_5}(\mathrm{ind}(u_0,\ldots,u_5),\mathtt{F}),\end{aligned} \tag{24}$$

where $\mathrm{ind}(u_0,\ldots,u_5)[i] = 1 \iff u_i \neq 0$. We also collect the following information about the Walsh spectra of the active Sboxes in the last round:

- $\widehat{S_8}(\,\cdot\,,\mathtt{1})$ has 15 zeros.
- $\widehat{S_6}(\,\cdot\,,\mathtt{1})$ has 17 zeros.
- $\widehat{S_1}(\,\cdot\,,\mathtt{4})$ has 14 zeros.
- $\widehat{S_4}(\,\cdot\,,\mathtt{4})$ has 12 zeros.
- $\widehat{S_2}(\,\cdot\,,\mathtt{1})$ has 20 zeros.
- $\widehat{S_3}(\,\cdot\,,\mathtt{4})$ has 18 zeros.

By adding the number of nonzero coefficients associated to each value in Table 1, we conclude that despite being a map defined in $\mathbb{F}_2^{42}$, the Walsh spectrum of f_2 only has around $2^{30.31}$ nonzero coefficients. Furthermore, since $\widehat{S_5}(\mathtt{3F},\mathtt{F}) = 0$, $u_0,\ldots,u_5$ cannot all be nonzero at the same time. All nonzero coefficients belong to at least one of six vector subspaces of dimension 35. Each subspace $\widetilde{U_i}$ is determined by fixing one $u_i = 0$, which is a six bit condition, as well as the bit condition $u_6[i] = 0$. Since $\widehat{S_5}(\mathtt{3D},\mathtt{F}) = 0$, we can ignore the subspace $\widetilde{U_1}$.

6.2 Attack Algorithm and Complexity

Based on the observations we have made on the key recovery map for the attack, we propose the following attack algorithm. We have provided a more thorough description of the subspaces Y, V and U_i as supplementary material.

Distillation phase and first set of Walsh transforms. The nonzero Walsh coefficients of the key recovery map form five affine subspaces which are handled separately. The first step in the analysis phase consists of five pruned transforms whose inputs are restricted to a subspace Y of dimension 40 (due to the duplicate input bits in the key recovery map) and whose outputs are restricted to subspaces $U_i = \mathbb{F}_2^6 \times \widetilde{U_i}$ of dimension 41. We can show that $\dim(Y/(Y \cap U_2^{\perp})) = 33$, $\dim(Y/(Y \cap U_0^{\perp})) = 35$, and $\dim(Y/(Y \cap U_i^{\perp})) = 37$ for $i = 3, 4, 5$.

1. Initialise three arrays g_3, g_4, g_5 of length 2^{37}, one array g_0 of length 2^{35} and one array g_2 of length 2^{33}.
2. For each pair (x, y), increment or decrement one position in each of the g_i according to the values of the appropriate parts of the plaintext and ciphertext and to $P[39] + P[50] + P[56] + C[7] + C[18] + C[24] + C[29]$.
3. Apply the fast Walsh transform on each of the g_i.

The time complexity of these steps is around $6N$ memory accesses and $3 \cdot 37 \cdot 2^{37} + 35 \cdot 2^{35} + 33 \cdot 2^{33} \simeq 2^{43.93}$ additions and subtractions.

Multiplying by the Walsh coefficients. The key recovery map has $50 \cdot 2^{30.31} \simeq 2^{35.95}$ nonzero Walsh coefficients. They can be enumerated by separating them into 32 sets, one for each $\widehat{S5}(\,\cdot\,, \mathtt{F}) \neq 0$, and looking at the nonzero positions in the LATs of (up to) 7 other active Sboxes.

1. Initialise one array h_2 of length 2^{38}, one array h_0 of length 2^{37}, and three arrays h_3, h_4, h_5 of length 2^{34}.
2. For each of the nonzero Walsh coefficients, retrieve the associated output of the first Walsh transform from one of the g_i (if the coefficient lies in more than one of the U_i, we can choose any), multiply it by the coefficient and add or subtract the result to the appropriate position of the array h_i.

The time complexity of this step is $7 \cdot 2^{35.95} \simeq 2^{38.76}$ products.

Second set of Walsh transforms and exhaustive search. The Walsh transforms in the second set are pruned at the inputs according to the subspaces U_i, and at the outputs according to a subspace V of dimension 40 given by the key schedule. We can show that $\dim(V/(V \cap U_2^{\perp})) = 38$, $\dim(V/(V \cap U_0^{\perp})) = 37$, and $\dim(V/(V \cap U_i^{\perp})) = 34$ for $i = 3, 4, 5$.

1. Perform the standard Walsh transform on the five arrays h_0, h_2, h_3, h_4, h_5.
2. For each of the 2^{40} possible key guesses, we add one coordinate from of each of the five arrays to obtain the experimental correlations. We keep the 2^{24} guesses with the highest correlation, as we aim for an advantage of 16 bits.
3. For each one of the 40-bit partial key guesses, we try all possibilities of the 16 other keybits exhaustively until either the key is found or the attack fails.

The time complexity of these steps is $38 \cdot 2^{38} + 37 \cdot 2^{37} + 3 \cdot 34 \cdot 2^{34} + 5 \cdot 2^{40} \simeq 2^{44.41}$ additions/subtractions and 2^{40} trial encryptions.

Attack complexity. The data complexity of the attack was determined using the model of Blondeau and Nyberg [8]. We obtain a 16 bit advantage with 70% probability with $N = 2^{41.5}$ data. The memory complexity is dominated by the ten arrays, which require $2^{39.74}$ memory registers of 64 bits. This can be reduced to around $2^{38.75}$ by performing the multiplication step in a way in which the g_i and the h_i do not have to be allocated at the same time.

For the time complexity, we consider that on a modern processor a DES encryption takes 16 clock cycles, a product takes 6 clock cycles, and a memory access or an addition take 1 clock cycle. We obtain

$$\frac{1}{16} \cdot 6 \cdot 2^{41.5} + \frac{1}{16}\left(2^{43.93} + 2^{44.41}\right) + \frac{6}{16} \cdot 2^{38.76} + 2^{40} \simeq 2^{42.13} \text{ DES encryptions.}$$

This attack is, to the best of our knowledge, the best in terms of data complexity. However, it has rather high time (if we exclude data generation) and especially memory complexities when compared to previous attacks.

Table 2: Comparison of selected attacks on the Data Encryption Standard.

	Complexity				
Type	**Data**	**Time**	**Memory**	P_S	**Source**
Differential Cryptanalysis	$2^{47.00}$ CP	$2^{37.00}$	$\mathcal{O}(1)$	58%	[5]
Linear Cryptanalysis	$2^{43.00}$ KP	$2^{39.00}$	$2^{26.00}$	50%	[26]
Multiple linear Cryptanalysis	$2^{42.78}$ KP	$2^{38.86}$	$2^{30.00}$	85%	[11]
Conditional Linear Cryptanalysis	$2^{42.00}$ KP	$2^{42.00}$	$2^{28.00}$	90%	[4]
Linear Cryptanalysis	$2^{41.50}$ KP	$2^{42.13}$	$2^{38.75}$	70%	Sect. 6

7 Application to PRESENT-128

In this section we introduce the first, to the best of the authors' knowledge, attack on 29-round PRESENT-128. It is based on a previous attack on PRESENT-80 [19] and adds an additional key recovery round. The attack uses the full codebook and has a time complexity of $2^{124.06}$ 29-round PRESENT encryptions.

The attack uses one of the three sets of linear approximations which were defined in [19] and provide a trade-off between capacity and key recovery complexity. We will use set II, which has 296 approximations and a total capacity of $2^{-57.8}$. Table 3 shows all the approximations which conform this distinguisher in a compact form. All of them have a single active Sbox in the first round and a single active Sbox in the last round, and the input mask always has Hamming weight 1 or 2 while the output mask always has Hamming weight 1.

7.1 Key Recovery Example for a Single Approximation

As an example, we consider the 24-round linear approximation with input mask `0000000000A00000` and output mask `0000000000200000`, between the 3rd and the 26th rounds. We add two rounds of key recovery at the input side and three rounds at the output side. We will apply the pruned Walsh tranform-based attack algorithm to compute its experimental correlation for all key guesses.

For comparison purposes, we consider the cost when using the Walsh transform without pruning. There are 32 active bits in K_1, 8 active bits in K_2, 4 active bits in K_{28}, 16 active bits in K_{29}, and 64 active bits in K_{30} (crossed-out in Figure 6). They add up to 28 bits of inner key guess K^I and 96 bits of outer key guess K^O. We thus require 2^{96} memory registers, and the time complexity is in the order of $96 \cdot 2^{96+28} \simeq 2^{130.6}$ additions, which leaves little margin to repeat it for several approximations using less than 2^{128} equivalent encryptions.

In order to reduce this cost as much as possible, we consider both the structure of the key recovery map and the key schedule in order to prune both stages of Walsh transforms. The key recovery map consists of three independent parts corresponding to each of the three active bits in the input and output masks.

Both parts corresponding to the input mask are essentially identical. If we denote by S_1 the second component of S (that is, the second output bit), then

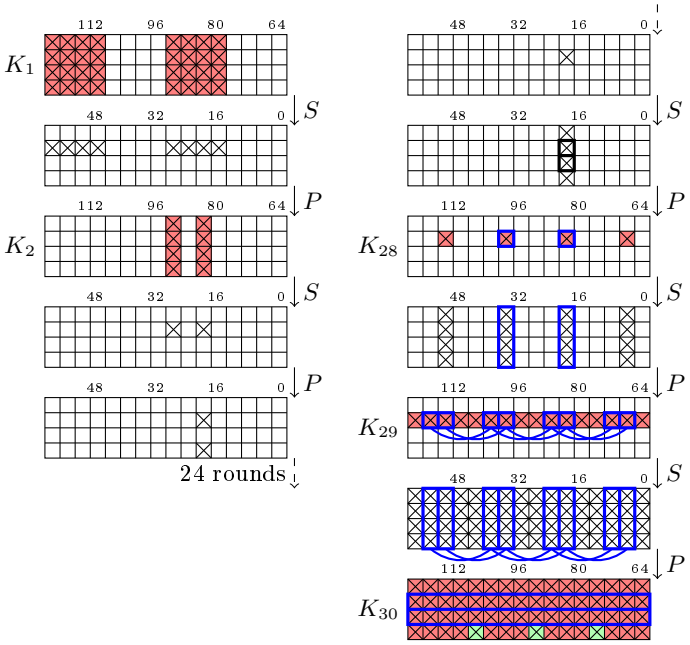

Fig. 6: Key recovery for one approximation.

Table 4: Restricted Walsh spectra used in the PRESENT-128 attack.

	v	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\widehat{S}_{S(x)\in\mathbb{F}_2^4\setminus\mathcal{X}}(2,\cdot)$	$\mathcal{X}=\varnothing$	0	0	4	4	−4	−4	0	0	4	−4	0	8	0	8	−4	4
	$\mathcal{X}=\{3,5,B,D\}$	0	0	4	4	−4	−4	0	0	0	0	0	8	0	8	0	0
	v	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\widehat{S}_{S(x)\in\mathbb{F}_2^4\setminus\mathcal{X}}(4,\cdot)$	$\mathcal{X}=\varnothing$	0	0	−4	4	−4	−4	0	8	−4	−4	0	−8	0	0	−4	4
	$\mathcal{X}=\{1,3,D,F\}$	0	0	0	0	−4	−4	0	8	−4	−4	0	−8	0	0	0	0
	v	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\widehat{S}_{S(x)\in\mathbb{F}_2^4\setminus\mathcal{X}}(8,\cdot)$	$\mathcal{X}=\varnothing$	0	0	4	−4	0	0	−4	4	−4	4	0	0	−4	4	8	8
	$\mathcal{X}=\{0,1,2,4,5,7,9,C\}$	0	0	0	−4	0	0	−4	0	0	4	0	0	−4	0	8	0

Table 3: The linear approximations of 24-round PRESENT which conform set II of [19], and which are also used in our attack.

Group ([19])	Input Mask	Input S-Box	Output Mask	Output S-Box	Qty.	24R ELP
A	A	5, 6, 9, 10	2,8	5, 7, 13, 15	32	$2^{-65.1}$
B	C	5, 6, 9, 10	2,8	5, 7, 13, 15	32	$2^{-65.6}$
C	A	5, 6, 9, 10	2,8	6, 14	16	$2^{-65.8}$
	A	13, 14	2,8	5, 7, 13, 15	16	
D	2,4	5, 6, 9, 10	2,8	5, 7, 13, 15	64	$2^{-66.0}$
E	C	5, 6, 9, 10	2,8	6, 14	16	$2^{-66.3}$
	C	13, 14	2,8	5, 7, 13, 15	16	
F	A	13, 14	2,8	6, 14	8	$2^{-66.5}$
G	2,4	5, 6, 9, 10	2,8	6, 14	32	
	8	5, 6, 9, 10	2,8	5, 7, 13, 15	32	$2^{-66.7}$
	2,4	13, 14	2,8	5, 7, 13, 15	32	
				Total	296	$2^{-57.8}$

these $\mathbb{F}_2^{16} \longrightarrow \mathbb{F}_2$ maps are of the form $S_1(S_1(x_3), S_1(x_2), S_1(x_1), S_1(x_0) \oplus k^I)$. Using Corollary 14, each of their Walsh coefficient is the product of up to five coefficients of the Walsh spectrum of S_1, which we note has six zeros.

We next look at the remaining part, which has a similar structure but over three rounds. In round 27, we consider $\widehat{S^{-1}}(\mathtt{F}, 2) = \widehat{S}(2, \mathtt{F}) = 4$. By rejecting the ciphertexts which lead to an input 3, 5, B or D to this Sbox, this coefficient becomes zero. We can split all nonzero Walsh coefficients into two affine subspaces of dimension 48 corresponding to the nonzero Walsh coefficients $\widehat{S^{-1}}_{x \in \mathbb{F}_2^4 \setminus \mathcal{X}}(\mathtt{B}, 2)$ and $\widehat{S^{-1}}_{x \in \mathbb{F}_2^4 \setminus \mathcal{X}}(\mathtt{D}, 2)$. The "inactive" bits in each of these subspaces have been surrounded by a thicker outline in Figure 6. The cost of this modification is a reduction of the available data by a factor of $3/4 = 2^{-0.42}$.

We now consider the key schedule. When pruning the Walsh transforms, we prefer relationships which are linear or which describe outer active keybits in terms of inner active keybits. We first guess the 28 innner keybits, painted (dark) red in the figure. The outer bits which can be deduced from these are colored (light) green. The other outer bits are guessed individually. There are three bits of K_{30} which can be deduced from the guess for K_{28}.

Let us compute the time complexity of obtaining all the $\widehat{\mathrm{cor}}(K^O, K^I)$. We start with two Walsh transforms whose outputs are restricted to subspaces of dimension $48 + 32 = 80$. The distillation phase costs $2N$ operations and requires $2 \cdot 2^{80}$ memory registers. The cost of the Walsh transforms themselves is $2 \cdot 80 \cdot 2^{80} \simeq 2^{87.32}$ additions. The cost of the Walsh coefficients multiplication is at most $\left(\frac{10}{16}\right)^{20} \cdot 2 \cdot 2^{80} \simeq 2^{67.44}$ products.

The second pair of Walsh transforms is repeated once for each of the 2^{28} guesses of K^I. In every case, the Walsh transforms have inputs restricted to subspaces of dimension 80 and outputs restricted to subspaces of dimension 93. The dimension of $X/(X \cap U^{\perp})$ is minimal and equal to 77. The total cost of these transforms is thus $2^{28} \cdot 2 \cdot 77 \cdot 2^{77} \simeq 2^{112.27}$ additions. The cost of combining the resulting arrays would be 2^{120} additions. However, 25 bits of the key guess at K_1 and three bits of the guess at K_{28} can be deduced from the guess at K_{30}, and the calculation can be performed with 2^{92} additions.

We note that we have decreased the time complexity by a factor of almost 2^{16} by increasing the data complexity by a factor of $4/3 \simeq 2^{0.42}$, illustrating that a carefully picked filtering of the data can lead to significant time gains.

7.2 Overview of the complete attack

We divide the 296 linear approximations into two groups depending on the Hamming weight of the input mask and their time complexity contribution:

Type I (groups D,G). These are the 160 approximations with input masks of Hamming weight 1. For these approximations, there are 16 active bits in K_1, 4 in K_2, 4 in K_{28}, 16 in K_{29} and 64 in K_{30}. We can compute $\widehat{\mathrm{cor}}$ for these approximations with around $160 \cdot 2^{24} \cdot 80 \cdot 2^{80} \simeq 2^{117.64}$ additions without pruning.

Type II (groups A,B,C,E,F). These are the 136 approximations with input masks of Hamming weight 2. There are 32 active bits in K_1, 8 in K_2, 4 in K_{28}, 16 in K_{29}, and 64 bits in K_{30}. We treat these approximations as in the example: we study the Walsh spectrum of the active Sbox in round 27 (see Table 4). For 48 approximations, we are interested in $\widehat{S}(2, \cdot)$, and we can split the Walsh spectrum of the key recovery map into two affine subspaces of dimension 80 by discarding $1/4$ of the data. For 40 approximations, the coefficient is $\widehat{S}(4, \cdot)$, and we can split the spectrum into 2 spaces of dimension 80 by discarding $1/4$ of the data. For the other 48 approximations, the coefficient is $\widehat{S}(4, \cdot)$, in which case the spectrum lies on a subspace of dimension 80 after discarding $1/2$ of the data. Given these restrictions, we can compute $\widehat{\mathrm{cor}}$ with either $2 \cdot 2^{28} \cdot 80 \cdot 2^{80} \simeq 2^{115.32}$ or $2^{28} \cdot 80 \cdot 2^{80} \simeq 2^{114.32}$ additions. Further reductions are possible if we used the key schedule, but they are different for each approximation. Ignoring these, $2^{123.08}$ additions are required in total. We also have to combine both arrays corresponding to each approximation, which requires at most 2^{92} additions per approximation if we consider the key schedule.

Computing the multiple linear cryptanalysis statistic: first step. We must also consider the cost of computing the multiple linear cryptanalysis statistic. Using the notation of [19], the approximations form $M_2 = 32$ groups (8 groups for type I and 24 groups for type II) which share the same key guesses K^O and K^I. We can compute the sum of squares within each group and combine them in the next step. Considering the key schedule, for the Type II approximations we need to guess at most 92 bits, and for Type I at most 88 bits. This step can thus be performed with $136 \cdot 2^{92} + 160 \cdot 2^{88} \simeq 2^{99.2}$ products and additions.

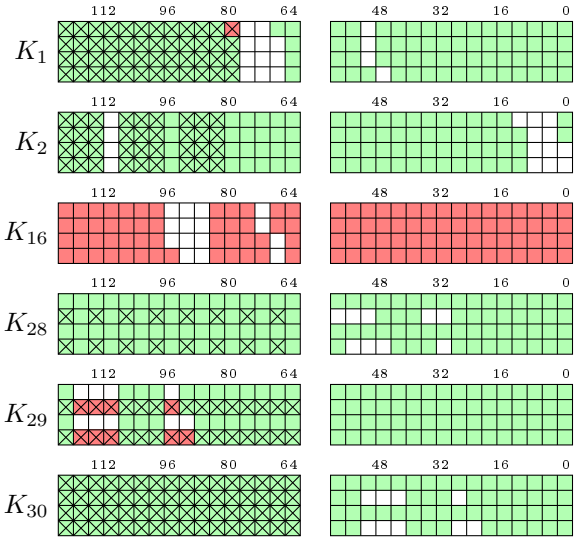

Fig. 7: Determining all the active keybits for all approximations (crossed out in the figure) with as few guesses as possible. We can deduce the (light) green bits if we know the 123 keybits highlighted in (dark) red.

Computing the multiple linear cryptanalysis statistic: second step. The combined tables of the previous step are used to compute Q_{k_T} for each value of a global key guess k_T. Figure 7 shows a guess of 123 keybits from which all the key guesses can be deduced. For each guess of k_T, we must add 32 values, one from each of the tables constructed in the previous step, at a cost of $32 \cdot 2^{123} = 2^{128}$ additions. After this we can find the the five remaining keybits with an exhaustive search costing 2^{123} encryptions if we aim for a five-bit advantage.

Data complexity. We once again use the model from [8], with careful consideration that the number of available plaintexts depends on the approximation. We find that if the whole codebook is used (2^{64} distinct known plaintexts), a 5 bit advantage is achieved with 67% probability.

Memory complexity. There are two main steps which contribute to the memory complexity. The distillation phase requires $160 \cdot 2^{16+64} + 136 \cdot 2 \cdot 2^{32+48} \simeq 2^{88.75}$ registers. The 32 intermediate multiple linear cryptanalysis statistic tables use $2^{99.2}$ memory registers, which dominate the memory complexity of the attack.

Time complexity. The dominant parts of the time complexity are the computation of the multiple linear cryptanalysis statistic and the final exhaustive key search. The latter takes 2^{123} PRESENT encryptions, while the former requires 2^{128} additions. If we assume that a sum requires at most 128 bit operations and

Table 5: Comparison of linear attacks on reduced-round PRESENT. Attacks on PRESENT-80 are also included for the sake of completeness. KP = Known Plaintext. DKP = Distinct Known Plaintext.

		Complexity				
Key	**Rds.**	**Data**	**Time**	**Memory**	P_S	**Source**
80	26	$2^{63.8}$ KP	$2^{72.0}$	$2^{32.0}$	51%	[15, 7]
		$2^{63.0}$ KP	$2^{68.6}$	$2^{48.0}$	95%	[10]
		$2^{61.1}$ KP	$2^{68.2}$	$2^{44.0}$	95%	[19]
		$2^{60.8}$ KP	$2^{71.8}$	$2^{44.0}$	95%	[19]
	27	$2^{64.0}$ KP	$2^{74.0}$	$2^{67.0}$	95%	[34]
		$2^{63.8}$ DKP	$2^{77.3}$	$2^{48.0}$	95%	[10]
		$2^{63.4}$ DKP	$2^{72.0}$	$2^{44.0}$	95%	[19]
	28	$2^{64.0}$ DKP	$2^{77.4}$	$2^{51.0}$	95%	[19]
128	28	$2^{64.0}$ DKP	2^{122}	$2^{84.6}$	95%	[19]
	29	$2^{64.0}$ DKP	$2^{124.06}$	$2^{99.2}$	67%	Section 7

a 29-round PRESENT encryption requires at least 3776 (64 for each subkey addition and Sbox layer), these will be equivalent to at most $2^{123.12}$ encryptions. The total time complexity is thus $2^{124.06}$ encryptions.

8 Conclusion

Summary of results. We have introduced a new framework for pruning of the fast Walsh transform to affine subspaces and used it as part of an optimised version of the attack algorithms of [16, 19], whose time complexity can be significantly lowered with respect to previous iterations.

In general terms, the time complexity of a key recovery attack using the fast Walsh transform largely depends on three factors: the number of active bits in the plaintext/ciphertext, the number of active keybits, and the number of input bits to the key recovery map which combines the two to evaluate the linear approximation. Previous versions of the Walsh-based attack algorithm often ran into a bottleneck imposed by the latter, in the sense that any additional redundancy in the key or the data would not reduce the time complexity or only reduce it by a logarithmic factor. Our improved algorithm can effectively exploit the construction of the map. In the application examples, the number of independent active keybits becomes the bottleneck of the attack.

We have showcased the usability of this improved version of the algorithm by describing two attacks which are only possible (in the sense of having a smaller time complexity than exhaustive search) thanks to it. We have provided the best known attack on the DES with regards to data complexity as well as the first attack on 29-round PRESENT-128 in the literature.

Further research. The first continuation to this work would be application to other ciphers. This technique might prove particularly useful in differential-linear cryptanalysis [23, 3], as using the same key guess for both ciphertexts in a pair introduces a lot of redundancy. We also think that, since applying the framework to an attack is a fairly technically involved task, an automatic tool which computes an optimal key recovery algorithm given a linear distinguisher of a block cipher could be of great use to the community.

The results shown in this paper are most effective in the case of specific cipher constructions, such as ciphers which use a bit permutation as the linear layer, a case in which the Walsh spectrum can be described in a succint way. It would be of interest to try to generalise these results to other common constructions. In the applications, we also find that the attacks become limited by the number of active keybits. For this reason, another open question would be whether it is possible to adapt conditional guessing techniques such as [12] to Walsh transform-based linear key recovery attacks. A broader open problem would be to find interesting applications of pruned fast Walsh transform algorithms to other problems in symmetric cryptology.

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).

References

1. Data Encryption Standard (DES). Federal Information Processing Standards Publication 46-3, U.S. Department of Commerce, National Institute of Standards and Technology (1977, reaffirmed 1988,1993,1999, withdrawn 2005)
2. Alves, R., Osorio, P., Swamy, M.: General FFT pruning algorithm. In: Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems (Cat.No.CH37144). vol. 3, pp. 1192–1195 vol.3 (2000)
3. Biham, E., Dunkelman, O., Keller, N.: Differential-linear cryptanalysis of serpent. In: Fast Software Encryption, 10th International Workshop, Revised Papers. Lecture Notes in Computer Science, vol. 2887, pp. 9–21. Springer (2003)
4. Biham, E., Perle, S.: Conditional linear cryptanalysis - cryptanalysis of DES with less than 2^{42} complexity. IACR Trans. Symmetric Cryptol. 2018(3), 215–264 (2018)
5. Biham, E., Shamir, A.: Differential cryptanalysis of the full 16-round DES. In: Advances in Cryptology - CRYPTO '92, Proceedings. Lecture Notes in Computer Science, vol. 740, pp. 487–496. Springer (1992)
6. Biryukov, A., Cannière, C.D., Quisquater, M.: On multiple linear approximations. In: Advances in Cryptology - CRYPTO 2004, Proceedings. Lecture Notes in Computer Science, vol. 3152, pp. 1–22. Springer (2004)
7. Blondeau, C., Nyberg, K.: Improved parameter estimates for correlation and capacity deviates in linear cryptanalysis. IACR Trans. Symmetric Cryptol. 2016(2), 162–191 (2016)
8. Blondeau, C., Nyberg, K.: Joint data and key distribution of simple, multiple, and multidimensional linear cryptanalysis test statistic and its impact to data complexity. Des. Codes Cryptogr. 82(1-2), 319–349 (2017)

9. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Cryptographic Hardware and Embedded Systems - CHES 2007, Proceedings. Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer (2007)
10. Bogdanov, A., Tischhauser, E., Vejre, P.S.: Multivariate profiling of hulls for linear cryptanalysis. IACR Trans. Symmetric Cryptol. 2018(1), 101–125 (2018)
11. Bogdanov, A., Vejre, P.S.: Linear cryptanalysis of DES with asymmetries. In: Advances in Cryptology - ASIACRYPT 2017, Proceedings. Lecture Notes in Computer Science, vol. 10624, pp. 187–216. Springer (2017)
12. Broll, M., Canale, F., Flórez-Gutiérrez, A., Leander, G., Naya-Plasencia, M.: Generic framework for key-guessing improvements. In: Advances in Cryptology - ASIACRYPT 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13090, pp. 453–483. Springer (2021)
13. Carlet, C.: Boolean Functions for Cryptography and Coding Theory. Cambridge University Press (2021)
14. Castro-Palazuelos, D., Medina-Melendrez, M., Torres-Roman, D., Yuriy, S.: Unified commutation-pruning technique for efficient computation of composite DFTs. EURASIP Journal on Advances in Signal Processing 11-2015 (2015)
15. Cho, J.Y.: Linear cryptanalysis of reduced-round PRESENT. In: Topics in Cryptology - CT-RSA 2010, Proceedings. Lecture Notes in Computer Science, vol. 5985, pp. 302–317. Springer (2010)
16. Collard, B., Standaert, F., Quisquater, J.: Improving the time complexity of Matsui's linear cryptanalysis. In: Information Security and Cryptology - ICISC 2007, Proceedings. Lecture Notes in Computer Science, vol. 4817, pp. 77–88. Springer (2007)
17. Collard, B., Standaert, F., Quisquater, J.: Experiments on the multiple linear cryptanalysis of reduced round serpent. In: Fast Software Encryption, FSE 2008, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5086, pp. 382–397. Springer (2008)
18. Cooley, J., Tukey, J.: An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation 19, 297–301 (01 1965)
19. Flórez-Gutiérrez, A., Naya-Plasencia, M.: Improving key-recovery in linear attacks: Application to 28-round PRESENT. In: Advances in Cryptology - EUROCRYPT 2020, Proceedings. Lecture Notes in Computer Science, vol. 12105, pp. 221–249. Springer (2020)
20. He, S., Torkelson, M.: Computing partial DFT for comb spectrum evaluation. IEEE Signal Processing Letters 3(6), 173–175 (1996)
21. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional linear cryptanalysis. Journal of Cryptology 32(1), 1–34 (2019)
22. Hu, Z., Wan, H.: A novel generic fast Fourier transform pruning technique and complexity analysis. IEEE Transactions on Signal Processing 53(1), 274–282 (2005)
23. Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Advances in Cryptology - CRYPTO '94, Proceedings. Lecture Notes in Computer Science, vol. 839, pp. 17–25. Springer (1994)
24. Markel, J.: FFT pruning. IEEE Transactions on Audio and Electroacoustics (1971)
25. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Advances in Cryptology - EUROCRYPT '93, Proceedings. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993)
26. Matsui, M.: The first experimental cryptanalysis of the Data Encryption Standard. In: Advances in Cryptology - CRYPTO '94, Proceedings. Lecture Notes in Computer Science, vol. 839, pp. 1–11. Springer (1994)

27. Nagai, K.: Pruning the decimation-in-time FFT algorithm with frequency shift. IEEE Transactions on Acoustics, Speech, and Signal Processing 34(4), 1008–1010 (1986)
28. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. Journal of Cryptology 21(1), 131–147 (2008)
29. Singh, S., Srinivasan, S.: Architecturally efficient FFT pruning algorithm. Electronics Letters 41(23), 1–2 (Nov 10 2005)
30. Skinner, D.: Pruning the decimation in-time FFT algorithm. IEEE Transactions on Acoustics, Speech, and Signal Processing 24(2), 193–194 (1976)
31. Sorensen, H., Burrus, C.: Efficient computation of the DFT with only a subset of input or output points. IEEE Transactions on Signal Processing 41(3), 1184–1200 (1993)
32. Sreenivas, T., Rao, P.: FFT algorithm for both input and output pruning. IEEE Transactions on Acoustics, Speech, and Signal Processing 27(3), 291–292 (1979)
33. Wang, L., Zhou, X., Sobelman, G.E., Liu, R.: Generic mixed-radix FFT pruning. IEEE Signal Processing Letters 19(3), 167–170 (2012)
34. Zheng, L., Zhang, S.: FFT-based multidimensional linear attack on PRESENT using the 2-bit-fixed characteristic. Security and Communication Networks 8(18), 3535–3545 (2015)