Methods for estimating the linear hull of light-weight block ciphers

Mathias Hall-Andersen

July 3, 2017

Abstract

This report details aspects of linear cryptanalysis against light-weight block ciphers, ciphers designed for small implementation in hardware, in particular the MIBS Feistel cipher and PRESENT SPN. We start by exhibiting the central concepts and motivations for linear cryptanalysis, then discuss methods for estimating the correlation distribution between parities over block ciphers in practice. After describing the ciphers under consideration, we explore 3 distinct methods for 'hull estimation' of PRESENT, two of which construct individual trails followed by a key-enumeration phase and a new method which applies an inductive/iterative estimation for a fixed key. We observe that for PRESENT the new method significantly outperforms the prior methods. We estimate the linear hull of PRESENT consisting of all trails with selection masks having Hamming weight ≤ 4 over 100 000 keys using the iterative method and also apply a variant hereof to MIBS. We observe that the distributions over 22 rounds of PRESENT and 12 rounds of MIBS deviates significantly from those expected over an ideal cipher. Furthermore we note that the employed strategy can be used as a heuristic for finding good candidate approximations over PRESENT.

1 Introduction

The field of cryptography is notoriously hard to define, but perhaps the shortest description of cryptography is that it replaces trust with mathematics (algebra), whether it be trust in a central authority or the confidentially of information transmitted over a channel, we seek to make as few assumptions about the participating elements as possible[1]. Despite the elusiveness of a clear definition, cryptography has been among the central enabling factors of practically every modern digital technology, from e-cash, securing web traffic and e-mail, encrypting hard-drives, to enabling citizens to prove their identity and the construction of smart contracts without trusted entities.

All these features of modern life are provided by 'cryptographic constructions', which are formal descriptions of how to accomplish the task at hand, usually implemented in code. A cryptographic construction provides a high-level 'service', e.g. the ability to send electronic cash between accounts. Enabling these high-level constructions is a number of different 'cryptographic primitives', cryptographic building blocks, which provide a much narrower (and yet more general) service, e.g. a method for turning any bit-string into a fixed sized bit-string (hash function)[2]. The relation between constructions and primitives is somewhat like the difference between the functioning of a car vs the function of the crankshaft herein.

A central (and perhaps the oldest and most widely known) class of cryptographic primitives is that of symmetric encryption. In this 'setting' the primitive provides a means to encipher and decipher messages send between two participants holding a shared secret. Whether it is English text exchanged between an aspiring usurper and her accomplices in the 15th hundred or JSON structures send between two servers in a world-wide cluster in the 21st century, when the encryption/decryption process involves the same secret key, symmetric encryption is employed. Symmetric encryption (and authentication) are omnipresent in our digital lives, embedded into constructions allowing for everything from encrypted VPNs, secure web connections (HTTPS) and encrypting data at rest (e.g. LUKS). The most common method for applying symmetric encryption is the use of so-called 'block ciphers', often used to provide both secrecy and authenticity of the data (ensuring the data

[1] Preferably replacing it with an assumption of 'computational hardness'

[2] Usually while making assumptions about the hardness finding another bit-string mapping to the same output; 'collision resistance'

has remained unchanged since encipherment). As the name implies, a 'block cipher' operates on fixed sized block of bits, usually transforming a plaintext of n bits into a ciphertext of n bits by a applying a key (usually also encoded as a bit-string). In this report we study such block cipher constructions and an widely applied attack against these.

When constructing cryptographic ciphers we naturally wish to evaluate their 'security', however formally proving the 'security' of practical cryptographic ciphers is beyond our present ability. Instead we consider their resistance to known attacks, where the designers themselves often provide analysis of the resistance against the most common attacks and the community at large subsequently attempts to improve the art. Linear cryptanalysis is one such commonly applied cryptanalytic technique, first conceived by Matsui and applied to FEAL [15] and DES [14] in 1993. Linear cryptanalysis exploits the correlations between linear functions over the inputs and outputs of a block cipher: the correlation is between random variables described as a linear combination of input/output bits, from which the technique derives its name. The primary interest in linear cryptanalysis is then to find these linear functions and to study the distribution of their correlation as it varies over the keys. However calculating the exact distribution or even just the exact correlation for a single key is computationally infeasible, therefore we must find methods for estimating the correlations instead.

In this report we explore the general concepts from linear cryptanalysis, methods for constructing the linear functions described above, discuss methods for practically estimating the correlation and estimate the correlation distribution between a number of such functions for two real-world ciphers.

2 Fundamentals

We start by defining the most crucial concepts to form the basis for further discussion, before continuing with a general discussion on how practical block cipher are constructed and how this relates to linear cryptanalysis.

2.1 Mathematical preliminaries

We let $\mathbb{F}_2$ be the field of two elements and $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. Let $\langle \cdot, \cdot \rangle$ denote the dot product in a vector space over $\mathbb{F}_2$. A binary

Boolean function is a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, a Boolean function is a function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. We often refer to the elements of $\mathbb{F}_2^n$ as bit strings, since this is a natural representation, addition as 'xor' (exclusive or) and multiplication as 'and'. Throughout the report '||' denotes concatenation of such bit strings and '$|\alpha|$', the size / dimension of the bit string α, while '$|\mathcal{S}|$' for a set $\mathcal{S}$ is the cardinality of the set. The Hamming weight is the number of non-zero symbols/bundles in a string:

Definition 1. *Hamming weight. Let $\alpha \in \mathbb{F}_2^n$, the Hamming weight $w_b(\alpha)$ of α where $b|n$, is the number of non-zero b bundles in α, When the bundle size b is left unspecified, $b = 1$ is implicit.*

A common way to construct Boolean functions over a larger domain is to apply a number of smaller functions in parallel:

Definition 2. *Bricklayer function [7, 2.3.3]. A bricklayer function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a Boolean function, which can be represented as:*

$$F(a_1||a_2||\ldots||a_c) = \phi_1(a_1)||\phi_1(a_1)||\ldots||\phi_c(a_c)$$

With Boolean 'component functions' $\phi_1 : \mathbb{F}_2^m \to \mathbb{F}_2^m, \ldots, \phi_c : \mathbb{F}_2^m \to \mathbb{F}_2^m$, such that $n = m \cdot c$. Often $\phi_1 = \phi_2 = \ldots = \phi_c$

We shall often be more concrete and refer to the component functions as S-boxes, this terminology suggests that the functions are implemented as lookup tables: often the rational for describing the function using a number of components is to obtain a succinct description of a non-linear function, by employing a number of smaller 'random' permutations[3]. As hinted in the introduction, one of the most central concepts explored in this report will be the correlation between binary Boolean functions.

Definition 3. *Correlation. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and $g : \mathbb{F}_2^n \to \mathbb{F}_2$ be binary Boolean functions. The correlation $C(f, g)$ is a scaled version of the bias:*

$$C(f, g) = 2 \cdot Prob\big(f(x) = g(x)\big) - 1$$

Where the probability is taken over all $x \in \mathbb{F}_2^n$.

[3]In reality these are chosen based on their resistance to different attacks. Including linear crypt analysis

The correlation between random Boolean variables describes their statistical relationship. A positive correlation implies that the Boolean variables take the same value with a probability $> 50\%$ and negative correlation with a probability $< 50\%$, in particular the correlation between Boolean constants is always ± 1. The correlation between two independent Boolean random variables is 0.

2.2 Block ciphers

We now provide a general discussion of block ciphers. A block cipher is a function in two variables $E : \mathcal{K} \times \mathbb{F}_2^n \to \mathbb{F}_2^n$, which forms a Boolean permutation on $\mathbb{F}_2^n$ for any key $K \in \mathcal{K}$. We refer to $k = \lceil \log_2(|\mathcal{K}|) \rceil$ as the key size, n as the block size and $\mathcal{K}$ as the key space. Most often $\mathcal{K} = \mathbb{F}_2^k$ for some key size k. In practice only a small amount of such permutations are computationally feasible and have succinct descriptions.

2.2.1 Ideal cipher

The ideal cipher for a block size n, is a cipher for which every permutation on $\mathbb{F}_2^n$ can be produced by instantiating the cipher with some key; the ideal cipher is a bijection onto the set of permutations from the key space.

Definition 4. *Ideal cipher. The ideal cipher for the block size n is a bijection from the key space $\mathcal{K}$ to the set of Boolean permutations over $\mathbb{F}_2^n$:*

$$E : \mathcal{K} \to (P : \mathbb{F}_2^n \to \mathbb{F}_2^n)$$

i.e. Instantiating the ideal cipher with a uniformly chosen key, yields a uniformly chosen permutation.

This definition deviates from the one commonly used in provable security (i.e. The 'Ideal cipher model'), by not introducing a limit on the key space: in particular there is only a single ideal cipher for a given block size (up to reordering/renaming of the key space). The number of permutations from $\mathbb{F}_2^n \to \mathbb{F}_2^n$ is $|\mathbb{F}_2^n|! = 2^n!$ which is clearly too large for any reasonable key size. Therefore ideal ciphers are completely infeasible to construct in reality for any meaningful block size.

2.2.2 Practical ciphers

Following the impossibility of constructing/using ideal ciphers, we now consider subsets of block ciphers which have succinct descriptions and a practical key space. In reality all modern block ciphers are constructed by repeating the same simple key-dependent permutation a number of times with different keys (in an attempt to strengthen resistance against attacks), this naturally leads to the definition of an iterated cipher:

Definition 5. *Iterated block ciphers [7, 2.4.1]. An iterated block cipher* $E : \mathbb{F}_2^k \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ *with* r *rounds, is a block cipher which can be represented as the repeated application of a invertible (round) function* $F : \mathbb{F}_2^q \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ *with (possibly) distinct round-keys* $k_1, \ldots, k_r$*:*

$$E(K, I) = F(K_r, F(K_{r-1}, \ldots F(K_1, I)))$$

Where $K_1, \ldots, K_r$ *is derived from* K *by applying a function* $KS : \mathbb{F}_2^k \to \mathbb{F}_2^q \times ... \times \mathbb{F}_2^q$ *which we refer to as the key schedule. We refer to* K *as the cipher key and* $K_1 || \ldots || K_r$ *as the expanded key.*

In practice many (but not all) ciphers alternate between adding a key (addition in the field) and applying a non-linear unkeyed operation, the composition of which forms a round in an iterated cipher:

Definition 6. *Key-Alternating cipher. A key-alternating cipher is an iterated block cipher, for which the round function* $R : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ *can be written as:*

$$R(K, I) = G(I + K)$$

In other words, as the addition of a round key K *and the application of an unkeyed permutation* G*. In particular, the size of the round-keys match the block size for any Key-Alternating cipher.*

Certain ciphers construct the unkeyed operation as a composition of a number of smaller permutations 'S-boxes' (substitution boxes) and a linear permutation (to provide diffusion or 'mixing' of the entire n-bit state). Such ciphers are called Substitution-permutation networks. Examples include AES and PRESENT (discussed later).

Definition 7. *Substitution-permutation network (SPN). An SPN is a Key-alternating cipher, where the unkeyed function* $G : \mathbb{F}_2^n \to \mathbb{F}_2^n$*. Can be described as the application of a substitution layer and a linear function [7,*

7.4.2]. The substitution layer is a Bricklayer function (Definition 2) with component functions commonly referred to as 'S-boxes' (substitution boxes).

Another common way to construct practical block ciphers are Feistel networks, among which are DES, Camellia and MIBS (discussed later).

Definition 8. *Feistel network. A (balanced) Feistel network is an iterated block cipher (Definition 5), where the round function* $G : \mathbb{F}_2^k \times \mathbb{F}_2^{2n} \to \mathbb{F}_2^{2n}$ *can be described as:*

$$G(K, L||R) = R||(F(K, R) + L)$$

With $|L| = |R|$*. Note that the function* $F : \mathbb{F}_2^k \times \mathbb{F}_2^n \to \mathbb{F}_2^n$*, need not be a permutation for* $G : \mathbb{F}_2^k \times \mathbb{F}_2^{2n} \to \mathbb{F}_2^{2n}$ *to be invertible. We often refer to F above simply as the 'F function' of the cipher.*

Some Feistel networks can be expressed as key-alternating ciphers, Camellia [11] and MIBS [10] are two such examples. In summary, we obtain the following inclusions:

$$\text{Feistel networks} \subseteq \text{Iterated ciphers}$$

$$\text{SPNs} \subseteq \text{Key-Alternating ciphers} \subseteq \text{Iterated ciphers} \subseteq \text{Block ciphers}$$

Since the set of permutation which can be obtained using the practical constructions above is minuscule compared to the set of all permutations, it seems reasonable to ask whether the constructions above are 'good enough' to obtain practical security.

2.3 Motivation

2.3.1 Distinguisher

Given a cipher, henceforth referred to as the target cipher, we may wish to evaluate its resistance to different attacks. Ideally there should be no way to even 'distinguish' the cipher from the ideal cipher, for which no non-trivial attacks apply. Note that the existence of a non-trivial attack yields a way to distinguish the ideal and target cipher. This naturally leads to a (simplified) definition of a distinguisher:

Definition 9. *Distinguisher. Let* $F_0 : \mathcal{K}_0 \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ *and* $F_1 : \mathcal{K}_1 \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ *be two block ciphers. Given a uniformly chosen* $i \in \{0, 1\}$, $K \in \mathcal{K}_i$ *and a uniformly chosen set of inputs* $\mathcal{I}$. *Let* $\mathcal{W} = \{(I, F_i(K, I)) : I \in \mathcal{I}\}$, *A distinguisher is a computable function* $D : \mathcal{W} \to \{0, 1\}$, *if* $D(\mathcal{W}) = i$ *the distinguisher 'wins' the game. An effective distinguisher clearly has* $|Prob\big(D(\mathcal{W}) = i\big) - 1/2| \gg 0$, *this quantity we call the 'power' of the distinguisher.*

This definition of distinguishers reflect the known plaintext setting of our particular attack. In practice we only consider distinguishers with a 'low' computational complexity (e.g. enumerating the entire key-space is prohibited), since the alternative leads to trivial attacks. In general distinguishability between an ideal cipher and the target cipher is itself considered an attack, however we shall see (in the next section) that it often leads to key recovery which indicate that the notion of indistinguishability is not 'too strong'; excluding schemes which would otherwise be 'secure'.

In the following sections we explore distributions of random variables over $\mathcal{W}$ that distinguishers can use to distinguish two ciphers (not necessarily the ideal vs target cipher). Construction of more complicated/powerful distinguishers is considered out of scope for the project, but we will give intuitions for constructing simple ones.

2.3.2 Key recovery

Suppose we can construct a sufficiently powerful distinguisher D between input/outputs of $n + 1$ rounds of an iterative cipher and $n - 1$ rounds of the same cipher. Then we can recover key material from the full n round cipher: given the set of input/outputs from the cipher, we attempt partial decryption of the last round with all possible round keys[4], for each key we query D with the partially decrypted pairs, if we chose the correct round key material we have successfully decrypted the last round and expect the distinguisher to recognize this with high probability. If we chose the wrong key material, we decrypted a wrong key and in effect encrypted the output

[4]Usually we can reduce our computational complexity by only guessing the key material needed to query the distinguisher; for linear cryptanalysis the bits selected by the input/output selection patterns. Rather than the entire round key, which is usually computationally infeasible.

by decrypting under a random key, hence we expect the distinguisher to recognize this cipher-/plaintext set as belonging to the $n+1$ round cipher.

Usually we push our attack as far as possible (distinguishing as many rounds as possible) and expect the $n+1$ round cipher to act comparably to an ideal cipher[5] to our distinguisher, hence the scenario is similar to the one discussed in the previous section, with $n-1$ rounds being the target and $n+1$ the 'ideal' cipher.

Variations of this methods exists, partial encryption of the plaintext and a combination of partial decryption/encryption of the ciphertext/plaintext is also commonly used in practice.

Generally the definition of a successful attack is any attack with a lower computational complexity than brute force and a data complexity less than full codebook[6], with a significant success probability over the key space.

2.4 Introduction to linear approximations

We now explore one method for constructing distinguishers in practice. First we define the central concept of linear approximations, then note useful relations applicable in the later analysis.

Definition 10. *Linear approximation. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a Boolean function[7]. A linear approximation of F is a pair of selection patterns (α, β), $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^m$, defining binary Boolean functions $f : x \mapsto \langle \alpha, x \rangle$, $g : x \mapsto \langle \beta, F(x) \rangle$. We refer to f as the input parity and g as the output parity.*

When expressing the correlation $C(f, g)$ between f and g we shall often write $C_F(\alpha, \beta)$ or omit F completely if it is clear from context. In this report we use will often simply use the word 'approximation' instead of the full 'linear approximation'.

[5] Of course the practical limitations still apply. Later we shall describe in more detail what 'acting comparably' implies in the case of linear cryptanalysis.

[6] All plaintext/ciphertext pairs over the cipher for the given key.

[7] Often F will be a bijection (e.g instantiation of a block cipher under a fixed key or a bijective S-box) in which case $n = m$ necessarily.

2.4.1 Linear distinguishers

In linear cryptanalysis, we use the correlation of linear approximations (Definition 10) to construct distinguishers. However the correlation of a linear approximation does not remain constant over all keys[8], since each key selects a (possibly) different permutation. We will be distinguishing between plaintext/ciphertext pairs from an ideal cipher and the target cipher under an unknown key, if the correlation distribution for the approximation over the ideal and the cipher differ significantly we can build a distinguisher which uses the likelihood of sampling the observed correlation from either distribution.

For an ideal cipher we expect the correlation to follow a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 2^{-n}$ [8, Theorem 4.7], if the distribution of the correlation for the approximation over the target cipher deviates from this by a significant margin we can construct an effective distinguisher, between the two.

2.4.2 Key enumeration

We are therefore interested in finding the distribution of the correlation between parities over the key space, both to determine how our distinguisher should function for the particular cipher and to estimate the power of our distinguisher. Practical methods for doing so is the primary topic for the remainder of this report.

The exact distribution can be found by instantiating the cipher with every possible key and enumerating every possible input, however this is clearly infeasible. Rather than calculating the exact distribution we can obtain an estimate by picking a large amount of random keys and calculating the correlation. However calculating the exact correlation using Definition 3 is still infeasible for all but the smallest block sizes (anything greater that $\approx$ 32-bit). Hence we need to use the structure of the target cipher to lower the computational cost of obtaining such an estimate, this will naturally lead us to the definition of trails and hulls, but first we exhibit classes of functions for which calculating the correlation of an approximation is feasible.

[8] Contrary to the common 'standard key-equivalence assumption' [5].

2.5 Properties of linear approximations

We now explore properties of the correlation for approximations over different classes of Boolean functions, commonly found in modern block ciphers.

2.5.1 Linear permutations

First we observe that approximations over bijective linear Boolean functions are trivial and the output parity is unique determined by the input parity:

Lemma 11. *If $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a bijective linear Boolean function, described as $F(x) = Mx$ (where M is an invertible matrix). Let $\beta \in \mathbb{F}_2^m$ be an output selection pattern, then there exists a unique input selection pattern $\alpha \in \mathbb{F}_2^n$, such that $C_F(\alpha, \beta) \neq 0$, given by $\alpha = F(\beta)$.*

Proof. Since:

$$\langle \beta, F(x) \rangle = \beta^T F(x) = \beta^T M x = (M^T \beta)^T x = \alpha^T x = \langle \alpha, x \rangle$$

□

In which case it follows from the proof that the correlation $C_F(\alpha, \beta)$ is 1.

Often we shall use the observation by simply noting the linear bijections of the cipher under consideration (e.g. The bit permutation of PRESENT in Section 4.1 and P function of MIBS in Section 4.2), then omit further discussion.

2.5.2 Addition of constants

The correlations of functions described as the addition of constants (xor of bit strings), is equally trivial, but highly relevant when considering key addition.

Lemma 12. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be described as:*

$$F(x) \mapsto x + c$$

Then $C_F(\alpha, \beta) = (-1)^{\langle \alpha, c \rangle}$ iff. $\beta = \alpha$ [7, 7.4.1].

Again this observation will be used without explicitly stating so, for example when 'keying the S-boxes' of MIBS and PRESENT; which amounts to nothing more than flipping the sign of the correlations for the approximations, based on the parity between the key and the input/output pattern.

2.5.3 Composition of functions

When approximating the composition of functions, we can describe the correlation by considering the correlation with all parities over the domain of the outer function, this will later lead to the definition of the 'linear hull'.

Lemma 13. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $G : \mathbb{F}_2^m \to \mathbb{F}_2^k$ be Boolean functions, let $\alpha \in \mathbb{F}_2^n, \beta \in \mathbb{F}_2^k$ be selection patterns. Then:*

$$C_{G \circ F}(\alpha, \beta) = \sum_{\gamma \in \mathbb{F}_2^m} C_F(\alpha, \gamma) \cdot C_G(\gamma, \beta)$$

We take note of the special case where one of the functions is key addition, in which case the sign of the correlation for the approximation over the function is filliped according to the parity between the key and the input parity. Another observation is that we can estimate the correlation $C_{G \circ F}(\alpha, \beta)$, by only considering the most significant term in the sum above. This will eventually lead to the concept of trails.

2.5.4 Bricklayer functions

Since the S-boxes are often stored as lookup tables, their domain is necessarily small (for practical reasons), small enough for us to enumerate all linear approximations over these functions and calculate their exact correlation, using the naive method implicit in Definition 3. We can the use the correlations of approximations over the component functions to construct approximations over the bricklayer function F for which the exact correlation is known. Since the bundles are disjoint, the parities over the components are independent and the correlation for the approximation over F is simply the product of the correlations for the approximations over the components:

Lemma 14. *Correlation over Bricklayer function (Definition 2). Given a bricklayer function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ with component functions: $\phi_1 : \mathbb{F}_2^m \to \mathbb{F}_2^m, \phi_2 : \mathbb{F}_2^m \to \mathbb{F}_2^m, \ldots, \phi_c : \mathbb{F}_2^m \to \mathbb{F}_2^m$. Let:*

$$\alpha = \alpha_1 || \alpha_2 || \ldots || \alpha_c$$

$$\beta = \beta_1 || \beta_2 || \ldots || \beta_c$$

With $|\alpha_i| = |\beta_i| = m : \forall c \geq i \geq 1$*, be input and output selection patterns over* F *respectively. Then [7, 7.4.3]:*

$$C_F(\alpha, \beta) = \prod_{i=1}^{c} C_{\phi_i}(\alpha_i, \beta_i)$$

The correlation whenever $\phi_i = \beta_i = 0$ is always 1 over any function. Hence approximation over bricklayer functions with few non-zero approximations over components tend to have a greater correlation magnitude. We shall later consider this observation in relation to cipher design, in particular the wide trail strategy and the rational for applying linear transformations to strengthen resistance against linear cryptanalysis; even though the approximations of these are trivial.

2.6 Correlation matrices

In later analysis, the notion of correlation matrices will provide a useful tool for understanding the rational for the 'iterative estimation'. Furthermore this concept is central to the work in [2], which has inspired this method.

Definition 15. *Correlation matrix [7, 7.3]. Let* $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *be a Boolean function. A correlation matrix* C^F *for* F *is a* $2^n \times 2^m$ *matrix where the entry* $C^F_{\beta,\alpha}$ *in row* β *and column* α *is defined as follows:*

$$C^F_{\beta,\alpha} = C_F(\alpha, \beta)$$

Note that constructing the full correlation matrix is infeasible for most Boolean functions e.g. the round function of an iterated cipher with a meaningful block size. Given correlation matrices for two functions, we can compute the correlation matrix for the composition simply as the matrix product:

Lemma 16. *Given two functions* $F : \mathbb{F}_2^n \to \mathbb{F}_2^k$ *and* $G : \mathbb{F}_2^k \to \mathbb{F}_2^m$ *the correlation matrix* $C^{F^{(2)} \circ F^{(1)}}$ *is the matrix product of the correlation matrices (Definiton 15) for the functions in the composition:*

$$C^{F^{(2)} \circ F^{(1)}} = C^{F^{(2)}} \times C^{F^{(1)}}$$

This follows from Lemma 13 and the definition of matrix multiplication.

This is a useful property when analysing iterative ciphers; even though we can not construct the full correlation matrix.

3 Linear trails

Let (α, β) be selections patterns over a block cipher. Calculating the exact correlation of (α, β) is computationally infeasible for all but the smallest block size. Furthermore, we are interested in the distribution of the correlation over the key-space, since the cipher is instantiated using a uniform key in the attack setting.

Most modern block ciphers are iterated ciphers (Definition 5) as described earlier. Because of this we can estimate the correlation by consider *linear trails*, which are list of selection patterns over these (almost) identical rounds:

Definition 17. *Linear trail. Let $F = R_r \circ \ldots \circ R_2 \circ R_1$ be an iterative Boolean function, a linear trail is a tuple of selection patterns $U = (U_1, \ldots, U_{r+1})$, defining linear approximations of $R_1, \ldots, R_r$ with selection patterns $\alpha_i, \beta_i = U_i, U_{i+1}$ for R_i for $1 \leq i \leq r$*

We call the set of all trails in which the first selection pattern is α and the last β, the 'linear hull' between α and β in F.

Definition 18. *Correlation contribution of linear trail [7, 7.59]. The correlation contribution $C_F(U)$ of a trail U is defined as:*

$$C_F(U) = \prod_{i=1}^{r} C_{R_i}(U_i, U_{i+1})$$

As the following lemma states, the correlation over the iterative function F between α and β can be described by the trails in the hull which start in α and terminate in β.

Lemma 19. *Let $F = R_r \circ \ldots \circ R_2 \circ R_1$ be an iterative Boolean function, let α, β be selection patterns over F, then the correlation $C_F(\alpha, \beta)$ becomes:*

$$C_F(\alpha, \beta) = \sum_{U, U_1 = \alpha, U_r = \beta} C_F(U)$$

This follows directly by induction from Lemma 13 and Definition 17

Note that the sign of the correlations for trails will vary, leading to the notion of 'constructive inference' where the correlation contribution of two trails have the same sign and similarly 'destructive inference' where the signs differ. We can estimate the linear hull / correlation by considering only the terms with a large absolute value (the 'dominant' terms in the summation). Informally we call the trails which have such large absolute correlations 'dominant trails' in the hull of the cipher (when the parities are clear from context). Analogously we can estimate the entries in the correlation matrices discussed earlier, by only inserting/calculating the entries corresponding to approximations with a large magnitude, operating on 'sparse matrices'.

3.1 Sign of correlations

For key-alternating ciphers the sign of the correlation contribution of a linear trail, depends on the parity between the trail and the expanded key.

Lemma 20. *Let $F = R \circ (+k_r) \circ ... \circ R \circ (+k_1)$ be a key-alternating cipher with round function R and expanded key $k_1, \ldots, k_r$. Then the correlation contribution (depending on the cipher-key) can be calculated as:*

$$C_F(U) = \prod_i (-1)^{\langle U_i, k_i \rangle} C_R(U_i, U_{i+1})$$

Sometimes a final key addition will occur, in which case the correlation is over the identity and the sign becomes ± 1 depending on the parity of the output mask with the final round-key.

Hence we can collect the set of trails first and estimate the correlation over the key space by computing the parity between each trail and each key enumerated, rather than repeat the search multiple times.

3.2 Linear potential

Simply considering the correlation contribution of the found trails cannot be a measure of success for the search, since we may find a subset with mostly destructive/constructive interference and the interference varies over the key space. Therefore we introduce the correlation potential:

Definition 21. *Correlation potential [7, 7.1.5] is simply the square of the correlation $C_F(\alpha, \beta)^2$. When referring to the potential of a trail, this is the square of the correlation contribution.*

Informally when we refer to 'better' trails, these are trails with a larger squared correlation contribution.

Lemma 22. *Assuming independent round-keys. The expected correlation potential (squared correlation) between input/output selection patterns (α, β), is the sum of the correlation potentials of all trails U_i between α and β [7, 7.67]*

$$E(C^2) = \sum_i C_F(U_i)^2$$

For key-alternating ciphers the expected correlation potential is key-independent and gives a quantitative measurement for 'how much' of the linear hull we take into account using the different methods examined later.

3.3 Wide trails

We now return to the discussion of approximations over bricklayer functions from earlier. As noted earlier the approximations with a lower number of non-trivial approximations over the non-linear component functions tends to have a greater squared correlation. Therefore we intuitively expect the dominant trails to be those 'activating' a small number of S-boxes (non-linear components) of the cipher. In the later chapters we will often use this observation as a heuristic when finding trails, since the Hamming (with bundle size matching the domain of the S-box) of the input selection pattern corresponds directly to the number of 'active' S-boxes:

Definition 23. *Linear branch number [7, 9.3.2]. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a Boolean function, the linear branch number $\mathcal{B}_b(F)$ of F is the minimum sum of Hamming weights of the input/output selection patterns for any non-trivial (i.e. $\alpha \neq 0 \wedge \beta \neq 0$), non-zero linear approximation:*

$$\mathcal{B}_b(F) = \min_{\alpha,\beta,C_F(\alpha,\beta)\neq 0} \{w_b(\alpha) + w_b(\beta)\}$$

The linear branch number over the round function is often used to argue about the maximum linear potential of any trail over the cipher, by giving a lower bound for the number of active S-boxes. To increase the linear branch number many cipher designers introduce linear transformations into

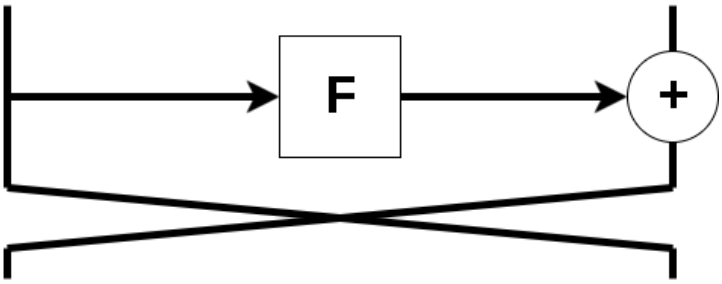

Figure 1: One round of a generic balanced Feistel

the round function; even if we only activate a single S-box in the current round a high branch number ensures that a large number of S-boxes are activated in the subsequent round. This is commonly referred to as the 'Wide trail strategy' [7, p. 126]). PRESENT does not employ the wide trail strategy, but opts for a bit-wise permutation (with no hardware cost) due to the emphasize on maintaining a low GE (gate equivalence), which results in PRESENT having a large number of trails with a low number of active S-boxes and consequently relatively high linear potential. MIBS incorporates a small matrix multiplication into the F-function to in an attempt to increase the linear branch number.

3.4 Branching relations

Lastly we note useful relations when constructing linear trails over Feistel networks. In particular these will be used in the analysis of MIBS (see 4.2).

3.4.1 Fork branch

We can describe the "three-forked branch", seen on the left in Figure 1, as $x \to (x, x)$. Let α be an input mask and (γ, β) corresponding output masks. Then it is easily seen:

$$\langle x, \alpha \rangle = \langle x, \gamma \rangle + \langle x, \beta \rangle \tag{1}$$
$$\langle x, \beta \rangle = \langle x, \alpha \rangle + \langle x, \gamma \rangle \tag{2}$$
$$\langle x, \beta \rangle = \langle x, \alpha + \gamma \rangle \tag{3}$$

Usually β will be the selection pattern propagated as the input mask to the following round.

3.4.2 XOR branch

Consider the "XOR branch", as seen on the right of Figure 1. This is simply addition in the field: $x, y \to x + y$. Let α be fixed a shared input mask, and let β be an output mask. Then:

$$\langle x + y, \beta \rangle = \langle x, \alpha \rangle + \langle y, \alpha \rangle = \langle x + y, \alpha \rangle$$

Hence $\beta = \alpha$, are the only approximations with non-zero correlation.

4 Ciphers

In this project the ciphers under consideration are so-called light-weight ciphers. Such ciphers are designed for highly efficient[9] implementation in hardware, while still offering an acceptable level of security. As a result these constructions are typically considerably simpler than those aimed for typical server/desktop applications.

4.1 PRESENT

4.1.1 Overview

PRESENT is a light-weight SPN, standardized by ISO in 2015 [9]. PRESENT has a block size of 64-bit and a key size of 80/128-bits (smaller versions has been defined to aid analysis [13]). The round function comprises of 3 stages:

1. Key addition; A 64-bit round key is xored into the state.
2. Substitution; 16 identical 4×4 S-boxes are applied to the state.
3. Permutation; A bit permutation, yielding full diffusion after 2 rounds[10].

The ordinary round function is repeated 31 times, followed by a final key addition (to avoid reversing the last round trivially). See [1] and [9] for a

[9] In terms of size (gate equivalents, proportional to die space) and power consumption.
[10] The input of any S-box can affect the input of any other after 2 rounds

full description of PRESENT. One crucial observation is that all operations in PRESENT are bitwise, this means that PRESENT has a linear branch number of just 2.

The only non-linear operation is the S-box applications, the full linear approximation table (LAT) over the PRESENT S-box can be found in appendix, showing the number of inputs for which the approximation holds minus half the size of the domain[11]. We (implicitly) use this table in the methods explored in Section 5 when constructing approximations over the substitution layer of PRESENT, following our observations in the discussion of bricklayer functions (Section 2.5.4).

4.1.2 Prior work

Ohkuma [16] has argued that the best linear trails for PRESENT are 'Single-Bit Paths', with only single bits active on either side of each S-Box. Since such trails remain 'narrow', they activate at most the same number of S-boxes in the following round and hence are good candidates for trails with large squared correlation; following our discussion on brick layer functions. Such linear trails are used in the work by Cho [6] claiming to break 26 rounds using multidimensional linear cryptanalysis, with a data complexity of $2^{62.4}$. Work by Abdelraheem [2] estimates the linear potential for all section patterns of low Hamming weight using sparse correlation matrices (Definition 15). This work has inspired methods for estimating the linear hull of PRESENT and MIBS explored later in this report.

4.2 MIBS

4.2.1 Overview

MIBS is a simple balanced Feistel network with a block size of 2×32 bit. The F-function has an SPN structure (in particular the F-function is a permutation), comprised of the following layers:

1. Key addition: A 32-bit round key is xored into the input.
2. Substitution: 8 identical bijective 4×4 S-boxes are applied.

[11] The correlation can be calculated from the entries by dividing with the size of the domain (yielding the 'bias') and multiplying by 2 to get the correlation.

3. Permutation: A linear operation on nibbles, see the section below.

The F-function is somewhat similar to that of Camellia. Observe that the construction allows MIBS to be expressed as a key-alternating cipher with 33 key additions, by letting the rightmost 32-bits of any round key be the leftmost 32-bit of the previous round key, see (Figure 6 in appendix). The only non-linear element of the F-function is the S-box applications, the linear approximation table over the S-box can be found in appendix. The key schedule is heavily inspired by PRESENT adopting the same pattern of rotation, S-box application and addition of a round constant, [10] describes versions of MIBS with both a 64 and 80-bit key. We shall consider the 80-bit variant only.

4.2.2 Linear transformation

The linear transformation of MIBS, henceforth denoted P, operates on 4-bit words. The matrix operation is detailed and constructed in [12, 4.2], but no rational is provided for the final permutation on nibbles [10]; which amounts to nothing more than reordering the rows of the example from [12] used by [10]. Let I be a 32-bit input to the linear layer: $I = x_1||x_2||x_3||x_4||x_5||x_6||x_7||x_8$ then the P can be described by the following matrix multiplication:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

Where the entries are scalars in $\mathbb{F}_2$. The linear branch number (Definition 23) of P is 5 (with a bundle size of 4) [4] [10]. Since the square matrix has full rank, the transformation is a bijection[12] and the inverse transformation is:

[12] Alternatively it is a permutation of the rows of a matrix defining a bijection [12]

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix}$$

4.2.3 Prior work

MIBS has received substantially less cryptanalysis than PRESENT. The authors of MIBS has conducted some analysis of the cipher, to evaluate the security, in an attempt to upper bound the bias of any linear trail. They hypothesis the best 4-round approximation, with a correlation of 2^{-7}, which is used to conclude that any trail over 32 rounds will have a correlation no greater than 2^{-56} and that full MIBS should therefore remain secure against linear cryptanalysis. The primary source of cryptanalysis of MIBS has been the work by Asli Bay, Serge Vaudenay et al. [4] [3] claiming to attack 13 rounds of MIBS-80 using differential cryptanalysis [4] and 19 rounds of MIBS-80 using multidimensional linear cryptanalysis with a data complexity of $2^{57.874}$ [3]. However we have been unable to verify these results (the claimed trails do no appear consistent over the permutation).

5 Estimating the linear hull of PRESENT

In this section we explore methods for estimating the hull of PRESENT and estimate the correlation distribution for parities using one of these methods. Throughout this chapter we evaluate the algorithms proposed by considering the linear hull of PRESENT over 22 rounds, between the following input/output selection patterns (in hexadecimal):

$$(\alpha, \beta) = (\texttt{0x8000000000000000}, \texttt{0x8000000000000000})$$

Using the number of trails and estimated squared correlation as a measurement for success. For some of these strategies ('breadth first' and 'neighbouring trails') the number of trails is relevant for complexity since key

enumeration is done separately for each individual trail. For the iterative estimation it is largely irrelevant.

All benchmarks are on a single core of an Intel i7-2600K @ 3.40 GHz.

5.1 Key-enumeration on PRESENT

In this section we aim to give a rough idea about the cost of key-enumeration for PRESENT, using the method following from Lemma 20. In the optimal case where both key and trail fit in D1 cache (verified using 'cachegrind'), the average number of trails per second for a single key is 10^7 trails/s. This is the relevant measurement, since in practice we instantiate the search with a number of keys, then add the trails found during the search to the correlation estimation for each such key. During our analysis we seek to enumerate 100 000 keys per trail, in practice the cost of key-enumeration will usually dominate the running time of the estimation, observe also that key-enumeration is an embarrassingly parallel problem.

5.2 Terminology

In the following section we will use the notion of forward/backward trails:

- Backward trails.
  A backward trail is a trail (Defintion 17) constructed over some number of rounds of the decryption operation.

- Forward trails.
  A forward trail is a trail, over some number of rounds of the encryption.

This will allow us to search to the 'middle' of the iterated cipher and construct trails over the entire cipher by concatenating forward and backwards trails.

5.3 Breadth first trail search

The primary observation is that the linear potential of a trail only decreases as more rounds are added. Hence it seems a reasonable assumption that the best $n + 1$ round trails, have the best n round trails as a prefix.

However it is possible that the greedy choices made when pruning the pool does not yield the most optimal trails over all rounds; since such choices assume that good trails remain decent in every round. The problem of pruning a globally optimal trail based on low absolute correlation contribution at a particular round becomes more prevalent when the ratio of the pool size to the total number of trails becomes 'too' small. In particular we cannot expect to find the best trail in the hull by letting the pool size be 1.

5.3.1 Algorithm

The search maintains a pool of trails, adding a round to each trail in each round and pruning based on linear potential. The algorithm consists of two searches:

1. The first $\lfloor \frac{r-1}{2} \rfloor$ rounds of the PRESENT in the forward direction.
2. The last $\lceil \frac{r-1}{2} \rceil$ rounds of PRESENT, searching backwards.

After collecting each set of 'partial trails', we insert these into buckets according to which S-boxes they activate in the meeting round (remembering to apply the permutations). Leaving us with a set of buckets for each direction. We then match the buckets pairwise and enumerate all pairs of partials trails (one from each bucket); such that the forward and backward trail activate the same S-boxes. For each such pair we check if the forced approximation over the substitution layer has non-zero correlation.

5.3.2 Evaluation

The results for the strategy using been different pool size parameters can be seen in Table 1. The running time is solely for the trail search (not including key enumeration). Notice that we get comparable results when the size of the pool is increased in the number of rounds, but the running time is lower.

5.4 Exploring neighbouring trails

Following Ohkumas observation [16] that Single-bit trails trails dominate the linear hull of PRESENT it seems a viable strategy to try and explore trails that have a low Hamming distance from Hamming weight 1 trails;

Rounds	Pool size	Trails	$E(C^2)$	Time
22	1000000	523 077 464	$2^{-62.9522}$	82.79s
22	2000000	1 421 369 806	$2^{-62.9392}$	171.17s
22	3000000	2 737 607 739	$2^{-62.9353}$	274.60s
22	4000000	4 328 261 485	$2^{-62.9304}$	396.60s
22	$1000 \cdot 2^r$	216 090 757	$2^{-62.9904}$	17.30s
22	$2000 \cdot 2^r$	921 763 481	$2^{-62.9452}$	50.08s
22	$3000 \cdot 2^r$	1 687 145 980	$2^{-62.9377}$	85.09s
22	$4000 \cdot 2^r$	2 561 741 402	$2^{-62.931}$	126.16s
22	$8000 \cdot 2^r$	7 978 414 360	$2^{-62.9145}$	379.91s

Table 1: Evaluation of breadth-first search

which activate exactly 1 S-box in every round. We call these new trails 'neighbouring trails'.

Since all trails with selection patterns having Hamming weight 1 can be exhaustively enumerated for 22 rounds, we can find such neighbouring trails by replacing a section of the trail with one of higher Hamming weight, thus obtaining trails that are 'close' to the original. We call the number of rounds replaced from the original trail, the 'leap' of the search.

5.4.1 Algorithm

Care must be taken to avoid 'duplicate trails', considering the same trail twice. A description of the simplified algorithm follows.

1. $F \leftarrow$ All forward Hamming weight 1 trails in the hull.
2. $B \leftarrow$ All backward Hamming weight 1 trails in the hull.
3. Set $S \leftarrow \emptyset$
4. For each $U \in F$:
   (a) For all $0 < n \leq r_{fw} - l$:
      i. Replace the section $[n : n + l]$ of the trail U using sub-search, yielding a new set of partial trails Q
      ii. $S \leftarrow S \cup Q$

5. Match S with B obtaining trails from the hull
6. Set $S \leftarrow \emptyset$
7. For each $U \in B$:
   (a) For all $0 < n \leq r_{bw} - l$:
      i. Replace the section $[n : n + l]$ of the trail U using sub-search, yielding a new set of partial trails Q
      ii. $S \leftarrow S \cup Q$
8. Match F with $S \setminus B$ obtaining trails from the hull

The number of found trails can be adjusted by pruning the sub-search or adjusting the number of rounds substituted (the 'leap'). Observe that 'leaping' less than 4 rounds (2 times full diffusion in PRESENT) will yield few additional results. This strategy is made efficient by two observations:

1. The search for neighbouring trails, can be done before matching the partial forward/backward trails.
2. The results of the high Hamming weight sub-search can be cached and reused at all round offsets.

5.4.2 Evaluation

The 'Leap' column shows how many rounds of the Hamming weight 1 trails we replace. The Hamming weight ('HW') is the maximum Hamming weight of any selection pattern of resulting trails.

Observe that adding the trails with Hamming weight 4 sections, does not yield a significant improvement over Hamming weight 3, because the magnitude of correlation contribution of such trails is significantly smaller. The greatest squared correlation contribution for any such trail is 2^{-110}. For 6 rounds the number of Hamming weight 4 trails becomes so large, that the neighboring trails herein begins to have a small effect on the estimated potential, however the running time of the key-enumeration phase makes this number of trails impractical (requiring at least ≈ 2027803 CPU hours on key enumeration alone).

Although significantly slower than the breadth first search, observe that the squared correlations in Table 2 are better than those of Table 1, while the

Rounds	HW	Leap	Trails	$E(C^2)$	Time
22	2	4 Rounds	545 180 856	$2^{-62.9145}$	488.70s
22	2	5 Rounds	706 485 734	$2^{-62.9104}$	635.17s
22	2	6 Rounds	1 007 846 304	$2^{-62.9099}$	909.34s
22	3	4 Rounds	969 391 374	$2^{-62.9129}$	873.08s
22	3	5 Rounds	3 103 530 089	$2^{-62.9078}$	2780.90s
22	3	6 Rounds	9 700 781 818	$2^{-62.9067}$	8719.26s
22	4	4 Rounds	1 112 503 702	$2^{-62.9129}$	1005.36s
22	4	5 Rounds	9 844 531 610	$2^{-62.9078}$	8893.94s
22	4	6 Rounds	73 009 116 065	$2^{-62.9059}$	66282.76s

Table 2: Evaluation of neighboring trail strategy

number of trails is comparable: a crucial advantage of this method is that it finds trails with a significantly higher linear potential (more 'dominant' trails) than the breadth first search, this is important since the running time of the key-enumeration phase is linear in the number of trails, hence the running time including key-enumeration is lower.

The reason why the method surpasses a simple breadth first search is that the breath first method depends on local choices leading to optimal trails. Where as this method considers a dominant trail and replaces some section hereof with what would otherwise be a terrible local choice (e.g activating 3 S-boxes in a round), yet the remainder of the trail is low Hamming weight and so the total number of active S-boxes remain small.

5.4.3 Comparison of trails

Here we note that the trails found by the two strategies discussed in the last sections are often quite different and observe artifacts in the trail due to the search method employed; in particular the set of trails for one of the strategies is not just a superset of the other.

The trails in Table 3 are typical trails from the respective strategies, both have an estimated squared correlation of 2^{-102}, despite the trail in from the breadth first strategy activating 30 S-boxes and the neighbouring trail 26 S-boxes. The 'breadth first trail' is not covered by the neighboring strategy, since we only replace a single section of the original trail, while this trail has many selection patterns over a wide range of rounds with a higher Hamming

Round	Breadth first trail (input masks)	Neighbouring trail (input masks)
1	8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0
3	0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0
4	0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0
5	0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0	0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0
6	0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0	0 2 0 0 0 2 0 0 0 2 0 0 0 0 0 0
7	0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 4 4 4 0 0 0 0 0 0 0 0 0
8	2 0 0 0 0 0 0 0 0 0 0 2 0 0 0	0 0 0 0 0 0 0 0 0 e 0 0 0 0 0 0
9	8 0 0 8 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0
10	9 0 0 0 9 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0
11	0 0 0 0 0 0 0 0 0 0 0 0 8 8 0	0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0
12	0 0 0 c 0 0 0 c 0 0 0 0 0 0 0	0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0
13	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0
14	0 0 0 0 c 0 0 0 0 0 0 0 c 0 0	0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15	0 8 0 8 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0
16	0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0	0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0
17	0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 0	0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0
18	0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0
19	0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0
20	0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0	2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
21	0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0	8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
22	2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Table 3: Comparison of trails for PRESENT

weight (rounds 8-15). The 'neighbouring trail' trail is not discovered during the breadth-first search, since the breadth first search makes greedy local choices and a trail activating 6 S-boxes over 2 rounds (rounds 6-7) will get pruned from the pool.

5.5 Iterative estimation of the hull

Rather than explicitly finding individual trails in the hull and enumerating keys over these afterwards, we consider the cipher instantiated with a fixed key and estimate the correlation between (α, β) for this fixed key. This allows us to consider a entirely new approach for estimating the correlation over

the key space: rather than enumerating trails, we inductively estimate the hull between α and a number of intermediate masks. This has the effect of 'collapsing the trails', allowing us to consider the influence of an exponential (in the round count) number trails. These intermediate selection patterns lie in a predefined 'mask-set':

Definition 24. *Mask-set. A mask-set $\mathcal{S}$, is a set of selection patterns from $\mathbb{F}_2^n$ for some fixed $n \geq 1$. Usually we shall consider the hull defined by trails with selection patterns (or 'masks') from $\mathcal{S}$.*

We start by defining the mask-set to estimate the linear hull over. For the search over PRESENT we let $\mathcal{S}$ be all selection patterns with Hamming weight less than some constant c and consider the hull of all trails in $\{\alpha\} \times \mathcal{S} \ldots \times \mathcal{S} \times \{\beta\}$, where $\beta \in \mathcal{S}$. Correctness of the algorithm below follows from the observation that we can 'inductively' expand an estimation of the hull for n rounds to an estimation over $n+1$ rounds:

Lemma 25. *Let $C_F(\alpha, \beta_1), C_F(\alpha, \beta_2), \ldots, C_F(\alpha, \beta_k)$ be the correlations between α and all output masks $\beta_1, \ldots, \beta_k$ for the Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. Let $G : \mathbb{F}_2^m \to \mathbb{F}_2^o$ be any Boolean function, then for any $\beta \in \mathbb{F}_2^o$:*

$$C_{G \circ F}(\alpha, \beta) = \sum_{\beta_i} C_F(\alpha, \beta_i) \cdot C_G(\beta_i, \beta)$$

This follows directly from Definition 18 and Lemma 19. We can obtain an estimate for the correlation, by restricting the selection patterns those of the mask-set.

By letting F be the prior n rounds of the cipher and G the keyed round function we can therefore estimate the correlation between α and all output masks in $\mathcal{S}$ (in particular $\beta \in \mathcal{S}$) after $n+1$ rounds (proceeding by induction to the full number of rounds). Recall that the correlation matrix (Definition 15) for the composition of two functions is the matrix product of their corresponding correlation matrices (Lemma 16). Hence by constructing the correlation matrices for the keyed round functions (and an initial matrix with the single non-zero entry $M_{(\alpha,\alpha)} = \pm 1$), with entries equal to zero for all selection patterns not in the mask-set, we could estimate the correlations with multiple output parities as the matrix product of these. Therefore this work bares some resemblance with that of Abdelraheem [2], discussed earlier, although we avoid the explicitly expressing the matrices.

5.5.1 Algorithm

With these observations in mind we can estimate the linear hull, by calculating the key dependent correlation for a number of keys and all low Hamming weight trails iteratively. Let P be the permutation in PRESENT, $(\alpha_{init}, \beta_{final})$ the approximation for which we want to estimate the hull and $K_0||\ldots||K_r$ be the expanded keys for r rounds. Then the algorithm can be described as:

1. Enumerate linear approximations over the S-box
2. Set $pool_{current} \leftarrow \emptyset$ (an empty map)
3. Set $pool_{new} \leftarrow \emptyset$ (an empty map)
4. If $\langle \alpha_{init}, K_0 \rangle = 0$, then $pool_{current}[\alpha_{init}] = 1$
5. If $\langle \alpha_{init}, K_0 \rangle = 1$, then $pool_{current}[\alpha_{init}] = -1$
6. For each round:
   (a) Calculate the correlations over the keyed S-boxes.
   (b) For each $\alpha \in pool_{current}$:
      i. Let $C = pool_{current}[\alpha]$
      ii. Estimate the substitution layer using the keyed S-boxes. Yielding an approximation over the layer (α, β) with correlation c
      iii. If $w_1(\beta) > hw$ $(\beta \notin \mathcal{S})$, then continue to next α
      iv. Let $\beta' = P(\beta)$
      v. If $\beta' \notin pool_{new}$, then $pool_{new}[\beta'] \leftarrow C \cdot c$
      vi. Else if $\beta' \in pool_{new}$, then $pool_{new}[\beta'] \leftarrow pool_{new}[\beta'] + C \cdot c$
   (c) $pool_{current} \leftarrow pool_{new}$
   (d) $pool_{new} \leftarrow \emptyset$
7. Return $pool_{current}[\beta_{final}]$

As noted earlier the algorithm computes the correlation between α_{init} and all possible output selection masks $\beta \in \mathcal{S}$. This further reduces complexity if we wish to estimate the hull for multiple linear approximations with a shared

input selection mask α. For a given Hamming weight hw, round count r and block size n, the algorithm has a memory complexity of:

$$O(search(hw)) = \sum_{i=1}^{hw} \binom{n}{i}$$

And time complexity:

$$O(search(hw, r)) = r \cdot \sum_{i=1}^{hw} \binom{i}{n}$$

5.5.2 Evaluation

The expected squared correlation can be estimated by replacing the correlations over the S-Box with their squared and applying the method for the expanded key equal to zero:

Lemma 26. *Let $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ be all the correlations (with duplicates) of the linear approximations over the S-boxes enforced by the trail U_i. For a key-alternating cipher, under assumption of independent round keys, the squared correlation can be estimated by squaring the correlations of the relevant S-box approximations. Since:*

$$E(C^2) = \sum_{U_i} C_F(U_i)^2 = \sum_{U_i} (c_{i_1} \cdot c_{i_2} \cdot \ldots \cdot c_{i_k})^2 = \sum_{U_i} c_{i_1}^2 \cdot c_{i_2}^2 \cdot \ldots \cdot c_{i_k}^2$$

We can count the number of trails in a similar manner, by replacing the correlation with a counter, which we add to the new pattern whenever a non-zero approximation is found over the round function.

As Table 4 clearly shows, this method considers the influence of a significantly larger portion of the hull (number of trails) than those constructing individual trails could ever do. From Table 4 we can also see that the trails in the Hamming weight 1 mask-set (which activate exactly 1 S-box in each round) dominate the hull of PRESENT.

Rounds	HW	Trails	$E(C^2)$	Time
22	1	$\approx 10^{7.5200}$	$2^{-63.019}$	0.00s
22	2	$\approx 10^{14.3235}$	$2^{-62.8617}$	0.07s
22	3	$\approx 10^{21.7632}$	$2^{-62.8564}$	2.40s
22	4	$\approx 10^{29.6627}$	$2^{-62.8545}$	111.14s
22	5	$\approx 10^{37.7888}$	$2^{-62.8544}$	1540.77s

Table 4: Evaluation of iterative hull estimation

5.5.3 Correlation distribution over the key space

The distribution of the correlation between input and output selection pattern has been estimated over the key space by sampling 100000 cipher keys at random, applying the key schedule and using the method described above to estimate the linear hull for a fixed expanded key. The result for $\mathcal{S} = \{\alpha \in \mathbb{F}_2^{64} : w_1(\alpha) \leq 4\}$, over 22 rounds of full PRESENT can be seen in Figure 2.

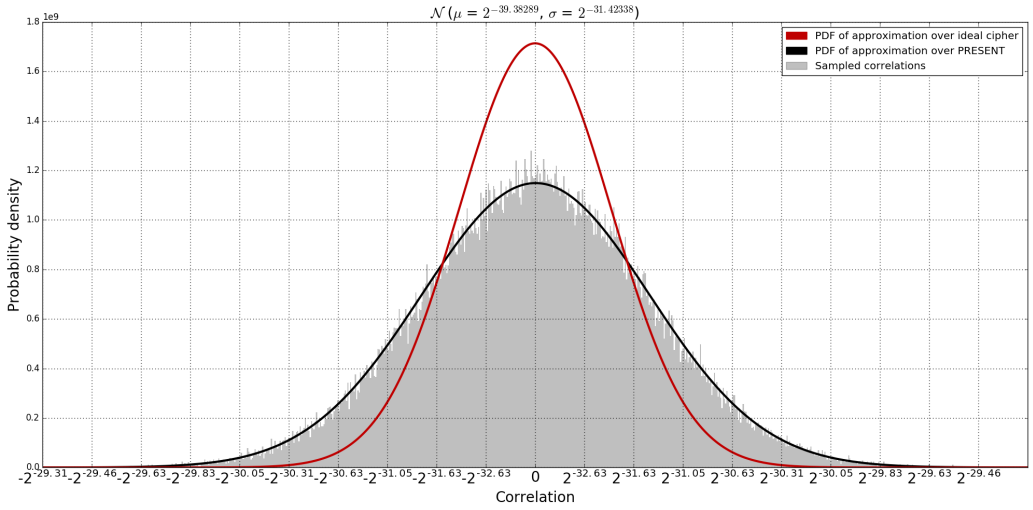

Figure 2: Estimated density of the correlation over the key space

5.5.4 Heuristic for finding approximations

Since the algorithm computes not just the correlation between α and β, but between α and any possible output mask in the mask-set, using the

observation from Lemma 26 we can use the algorithm as a heuristic for choosing linear approximations over PRESENT: by enumerating candidate input selection patterns and extracting the best output masks, based on estimated squared correlation. For the input parity `0x8000000000000000`, the output parity with the greatest linear potential is `0x8000000080008000` when considering the hull of all trails with selection patterns having at most Hamming weight 5. When only considering Hamming weight 1 trails, the approximation considered in this section has the greatest linear potential.

6 Estimating the linear hull of MIBS

6.1 Mask-set search

Like the iterative hull estimation applied to PRESENT, we explicitly define a mask-set to consider. We start by finding the set $\mathcal{Q}$ of inputs to P^{-1} activating at most 4 S-boxes over the F-function:

$$\mathcal{Q} = \{w_4(P^{-1}(I)) \leq 4\}$$

For MIBS we consider the mask-set consisting of selection patterns which activate at most 4 S-boxes each round. This restricts all the selection patterns of any trail to the set:

$$\mathcal{S} = \{L||R : R \in \mathcal{Q}\}$$

Observe that due to the Feistel construction, the left part of the intermediate selection patterns becomes restricted exactly like the right. However the number of intermediate masks still remain impractical, since $|\mathcal{Q}| = 3739171$ and hence the maximum number of intermediate selection patterns becomes $\approx 10^{13}$; which is beyond our available memory. Therefore we must prune the estimation process, discussed later.

Suppose $L||R$ is an input selection pattern to the current round. Let R' be the output selection pattern over S^{-1} for the input selection pattern $P^{-1}(R)$. Then we can prune to the mask-set while choosing approximations over the S-Box, since:

$$R||(L + R') \in \mathcal{S} \iff L + R' \in \mathcal{Q}$$

We use this observation by computing $\mathcal{Q}$ and inserting the elements into a trie with 4-bit words. As we choose approximations over S^{-1} we xor the output mask of the S-box approximation with the corresponding nibble from the left 32-bits of the state and walk the trie. If no node was found, we backtrack and choose another approximation over this S-Box. In our search we considers trails with input selection pattern:

$$\Gamma = \{L||0^{32} \ : L \in \mathcal{Q}\}$$

This has the effect of 'skipping' the first round, since the linear approximation over the substitution layer of the F-function becomes trivial. Note however that this is not necessarily optimal, since the right state may cancel with nibbles from the left in the following round. The trail described in [10] is not of this form.

6.1.1 Algorithm

Pre-computation phase:

1. Set $\mathcal{Q} \leftarrow \emptyset$
2. For each input $I \in \mathbb{F}_2^{32}$:
   (a) Compute $w = w_4(P^{-1}(I))$
   (b) If $w \leq 4$, then $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{I\}$
3. Save $\mathcal{Q}$

Naive implementation of correlation estimation:

1. Let $\mathcal{S} = \{L||R : R \in \mathcal{Q}\}$
2. Set $pool_{current} \leftarrow \emptyset$ (an empty map)
3. Set $pool_{new} \leftarrow \emptyset$ (an empty map)
4. Set $pool_{current}[\alpha] = 1$
5. For each round:
   (a) For each mask $\alpha \in pool_{current}$:

i. Split into $L||R = \alpha$
ii. Compute $R' = P^{-1}(R)$
iii. For each approximation over the inverted substitution layer of the F-function with R' as input selection pattern, yielding R'' as output selection pattern with a correlation C:
   A. Let $\beta = R||(L \oplus R'')$
   B. Let $c = C \cdot pool_{current}[\alpha]$
   C. If $\beta \notin \mathcal{S}$, continue with next such approximation.
   D. If $\beta \in pool_{new}$ then $pool_{new}[\beta] \leftarrow pool_{new}[\beta] + C$
   E. If $\beta \notin pool_{new}$ then $pool_{new}[\beta] \leftarrow C$

(b) Set $pool_{current} \leftarrow pool_{new}$

(c) Set $pool_{new} \leftarrow \emptyset$

In the actual implementation membership of $\mathcal{S}$ is checked simultaneously with the approximation of the substitution layer (as described above). Due to memory constraints we have to limit the pool size to ≈ 100000, by only keeping the masks with the highest squared correlation after each round; which is calculated separately from the correlation to ensure the hull under consideration remains the same for all keys. We obtain a 'hybrid' of the breadth-first search and the iterative estimation.

Over 10 rounds, I have estimated the linear potential of all input masks in $\alpha \in \Gamma$ and any output selection pattern $\beta \in \mathcal{S}$. The best of which is shown in the appendix (Table 7). For key enumeration we can consider the hull over MIBS for 12 rounds, with the following two approximations:

$$(\alpha, \beta) = (\texttt{0x8888800800000000}, \texttt{0x80888080d000d000})$$

With an estimated squared correlation of $2^{-61.98772}$

$$(\alpha, \beta) = (\texttt{0x2222200200000000}, \texttt{0x2022202060006000})$$

With an estimated squared correlation of $2^{-61.98785}$

6.1.2 Evaluation

If we plot a histogram over the correlations estimated over 100 000 keys (like done PRESENT earlier), we observe that the linear hull of MIBS looks

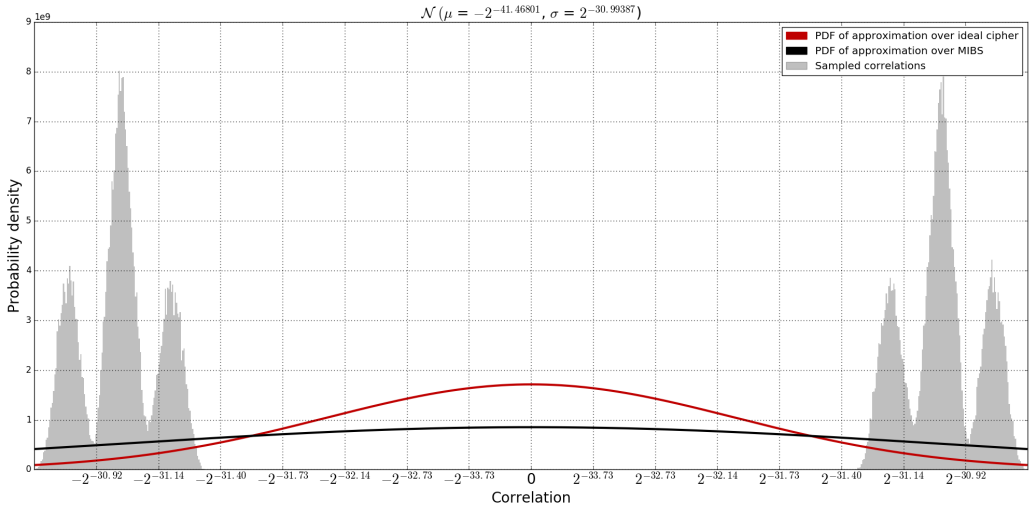

Figure 3: Distribution of the correlation over 12 rounds of MIBS

significantly different (See Figure 3), namely that the correlation does not approximate a normal distribution.

The peaks in the distribution (Figure 3) indicates that the hull contains a few very dominant trails; since destructive interference seems to be minimal (which would otherwise 'smooth out' the peaks observed). A distinguisher can therefore be constructed by checking if the sampled correlation lies close to these 6 distinct peaks. Furthermore the average squared correlation is $2^{-61.98771}$, which is very closed to the estimated squared correlation given above. When we repeat the enumeration over 14 rounds we obtain the results seen in Figure 4, we again get a distribution which is close to normal. The significantly different distributions indicate the possibility of constructing a successful distinguisher between 12 and 14 rounds of MIBS, which could lead to key recovery (as discussed earlier) over 13 rounds of MIBS. Note that even though the normal distribution for 14 rounds has variance less than 2^{-64}, we still expect the wrong key distribution of the correlation to be $\mathcal{N}(0, 2^{-64})$ [5]; at worst we will underestimate the power of our distinguisher.

6.2 Using multiple approximations

Lastly I briefly mention the possibility of using multiple approximations

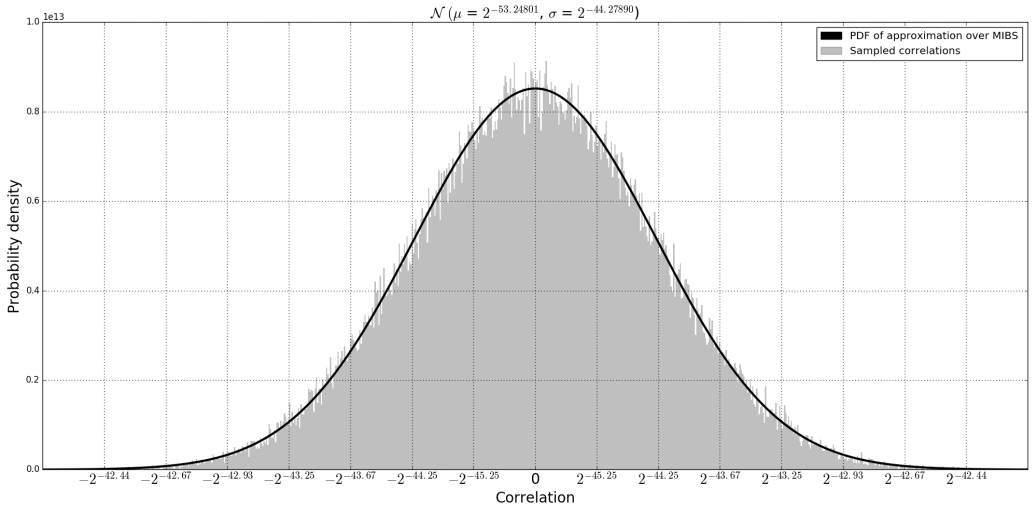

Figure 4: Distribution of the correlation over 14 rounds of MIBS

when constructing linear distinguishers ('multidimensional linear cryptanalysis'). If we plot the correlation distributions for both MIBS approximations using a heatmap we obtain a diagram like the one found in Figure 5. We expect the distribution for an ideal cipher to be a bivariate normal distribution with mean 0 and identity covariance matrix; which is clearly not what we observe in Figure 5. Hence a distinguisher could implement binary hypothesis testing by 'drawing boundaries' around the 4 clusters. Using more approximations over the cipher may decrease the false positive rate of the distinguisher leading to lower computational complexity or the ability to distinguish over additional rounds.

7 Conclusions

We began with a discussion about the high level concept of distinguishers between block ciphers and saw that linear approximations can be used to construct distinguishers by considering the distributions of the correlation over the key space. Using key enumeration, we have exhibited different methods for estimating the distribution using the dominant linear trails in the hull between the parities. We have explored 'traditional' trail searching strategies, where search is followed by key enumeration and the 'iterated' estimation

where we estimate the hull for a concrete instantiation of the cipher (applied to MIBS and PRESENT). We saw that the iterative/inductive method significantly outperformed the ones considering individual trail for PRESENT.

We have estimated the distribution of the correlation for approximations over both PRESENT and MIBS, observing correlation distributions over 22 rounds of PRESENT and 12 rounds of MIBS which deviates from ones expected over an ideal cipher.

References

[1] G. Leander C. Paar A. Poschmann M.J.B. Robshaw Y. Seurin C. Vikkelsoe A. Bogdanov, L.R. Knudsen. PRESENT: An ultra-lightweight block cipher. 2007.

[2] Mohamed Ahmed Abdelraheem. *Estimating the Probabilities of Low-Weight Differential and Linear Approximations on PRESENT-Like Ciphers*, pages 368–382. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[3] Aslı Bay, Jialin Huang, and Serge Vaudenay. *Improved Linear Cryptanalysis of Reduced-Round MIBS*, pages 204–220. Springer International Publishing, Cham, 2014.

[4] Asli Bay, Jorge Nakahara, and Serge Vaudenay. *Cryptanalysis of Reduced-Round MIBS Block Cipher*, pages 1–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[5] Andrey Bogdanov and Elmar Tischhauser. On the wrong key randomisation and key equivalence hypotheses in matsui's algorithm 2. In *FSE*, pages 19–38. Springer, 2013.

[6] Joo Yeon Cho. *Linear Cryptanalysis of Reduced-Round PRESENT*, pages 302–317. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[8] Joan Daemen and Vincent Rijmen. Probability distributions of correlation and differentials in block ciphers. *Journal of Mathematical Cryptology*, 1(3), jan 2007.

[9] ISO. Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers. Standard, International Organization for Standardization, Geneva, CH, January 2012.

[10] Maryam Izadi, Babak Sadeghiyan, Seyed Saeed Sadeghian, and Hossein Arabnezhad Khanooki. MIBS: A new lightweight block cipher. In *Proceedings of the 8th International Conference on Cryptology and*

Network Security, CANS '09, pages 334–348, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] Masayuki Kanda. Practical security evaluation against differential and linear cryptanalyses for feistel ciphers with spn round function. april 2001.

[12] Masayuki Kanda, Youichi Takashima, Tsutomu Matsumoto, Kazumaro Aoki, and Kazuo Ohta. *A Strategy for Constructing Fast Round Functions with Practical Security Against Differential and Linear Cryptanalysis*, pages 264–279. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[13] Gregor Leander. Small scale variants of the block cipher PRESENT. 2010. g.leander@mat.dtu.dk.

[14] Mitsuru Matsui. *Linear Cryptanalysis Method for DES Cipher*, pages 386–397. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[15] Mitsuru Matsui and Atsuhiro Yamagishi. *A New Method for Known Plaintext Attack of FEAL Cipher*, pages 81–91. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[16] Kenji Ohkuma. Weak keys of reduced-round PRESENT for linear cryptanalysis, 2009.

Appendices

		Output selection pattern															
Input selection pattern		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	-2	0	2	0	-2	-4	-2	2	0	-2	0	2	0	2	-4
	2	0	0	-2	-2	-2	2	-4	0	0	4	2	-2	-2	-2	0	0
	3	0	2	2	0	2	0	0	2	-2	4	0	2	0	2	-2	-4
	4	0	-2	-2	4	-2	0	0	2	0	-2	2	0	-2	0	-4	-2
	5	0	0	-2	2	2	-2	0	0	2	2	0	4	-4	0	2	2
	6	0	-2	4	2	0	-2	0	-2	0	2	4	-2	0	2	0	2
	7	0	4	0	0	0	-4	0	0	-2	-2	2	-2	-2	-2	2	-2
	8	0	2	2	4	0	2	-2	0	-2	0	0	2	2	-4	0	2
	9	0	0	2	-2	-4	-4	-2	2	0	0	-2	2	0	0	-2	2
	A	0	-2	0	-2	-2	0	2	-4	-2	0	2	4	0	-2	0	-2
	B	0	0	4	0	-2	2	2	2	4	0	0	0	-2	-2	2	-2
	C	0	0	0	0	2	-2	2	-2	2	2	-2	-2	0	-4	-4	0
	D	0	2	0	-2	2	0	-2	0	4	-2	4	2	2	0	-2	0
	E	0	4	-2	2	-4	0	2	-2	2	2	0	0	2	2	0	0
	F	0	2	2	0	0	2	-2	-4	0	-2	-2	0	-4	2	-2	0

Table 5: Linear approximation table for the MIBS S-box

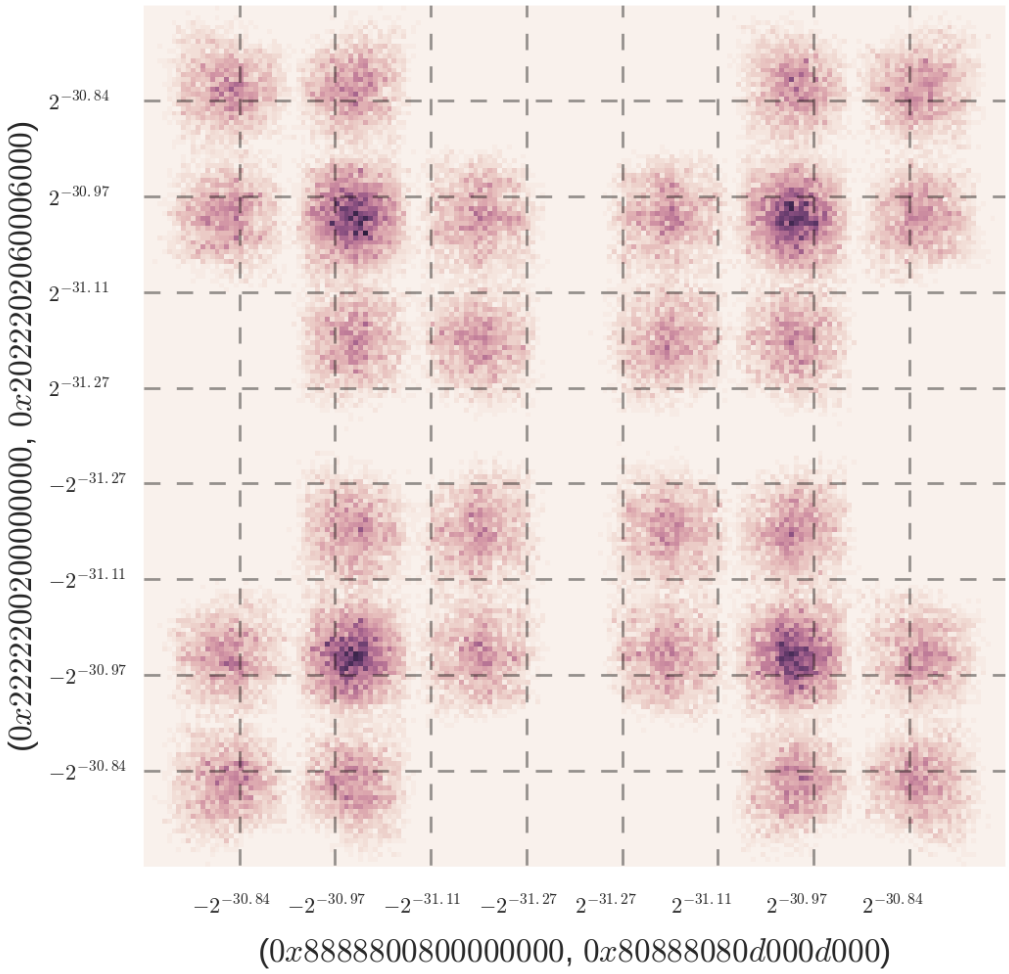

Figure 5: Heatmap of correlations for both MIBS approximations (12 rounds)

		Output selection pattern															
Input selection pattern		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	-4	0	-4	0	0	0	0	0	-4	0	4
	2	0	0	2	2	-2	-2	0	0	2	-2	0	4	0	4	-2	2
	3	0	0	2	2	2	-2	-4	0	-2	2	-4	0	0	0	-2	-2
	4	0	0	-2	2	-2	-2	0	4	-2	-2	0	-4	0	0	-2	2
	5	0	0	-2	2	-2	2	0	0	2	2	-4	0	4	0	2	2
	6	0	0	0	-4	0	0	-4	0	0	-4	0	0	4	0	0	0
	7	0	0	0	4	4	0	0	0	0	-4	0	0	0	0	4	0
	8	0	0	2	-2	0	0	-2	2	-2	2	0	0	-2	2	4	4
	9	0	4	-2	-2	0	0	2	-2	-2	-2	-4	0	-2	2	0	0
	A	0	0	4	0	2	2	2	-2	0	0	0	-4	2	2	-2	2
	B	0	-4	0	0	-2	-2	2	-2	-4	0	0	0	2	2	2	-2
	C	0	0	0	0	-2	-2	-2	-2	4	0	0	-4	-2	2	2	-2
	D	0	4	4	0	-2	-2	2	2	0	0	0	0	2	-2	2	-2
	E	0	0	2	2	-4	4	-2	-2	-2	-2	0	0	-2	-2	0	0
	F	0	4	-2	2	0	0	-2	-2	-2	2	4	0	2	2	0	0

Table 6: Linear approximation table for the PRESENT S-box

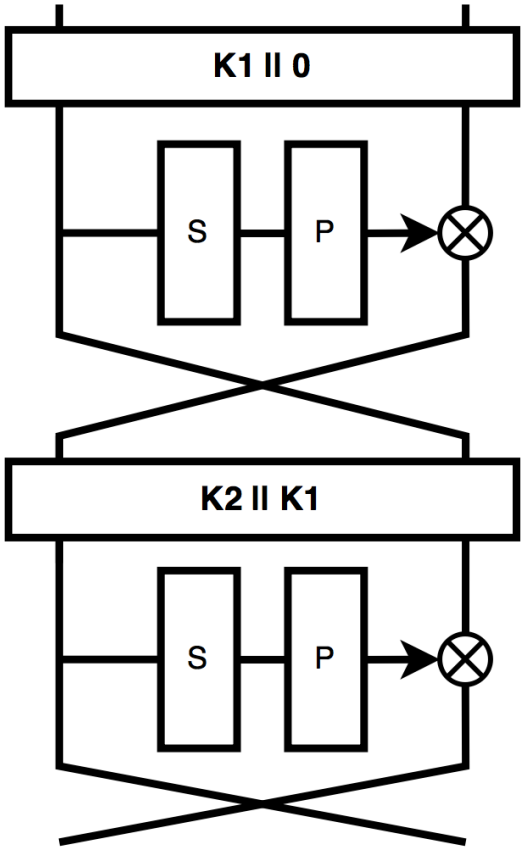

Figure 6: Feistel with SPN F-function expressed as a key-alternating cipher

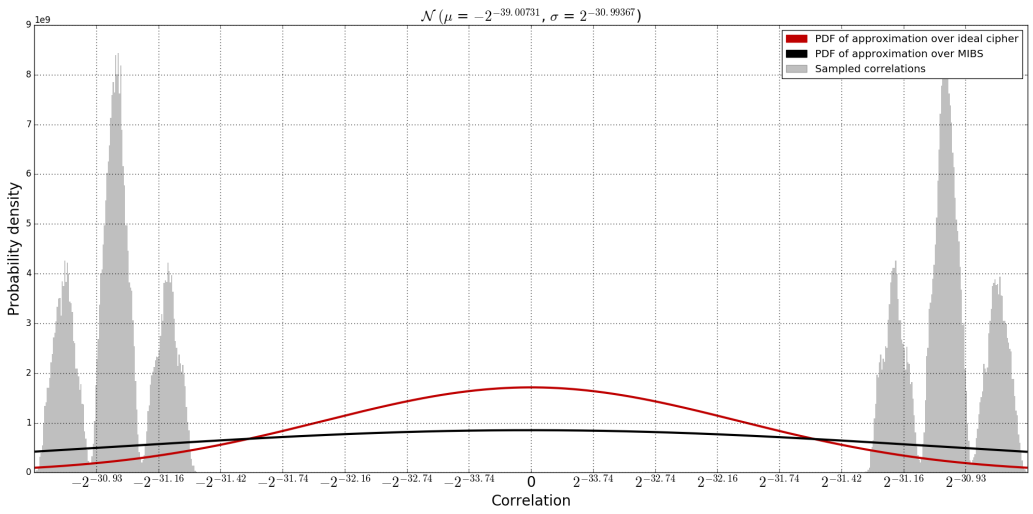

Figure 7: Estimated distribution of correlation for approximation $(\texttt{0x2222200200000000}, \texttt{0x2022202060006000})$ over 12 rounds of MIBS

Input selection pattern	Best $E(C^2)$
0x8888800800000000	$2^{-51.987767}$
0x2222200200000000	$2^{-51.988001}$
0xccccc00c00000000	$2^{-51.988170}$
0x282220a800000000	$2^{-51.993612}$
0x828880a200000000	$2^{-51.993663}$
0xbcbbb07c00000000	$2^{-51.994307}$
0xcbccc07b00000000	$2^{-51.994319}$
0x88828a0200000000	$2^{-51.994363}$
0x82828aa800000000	$2^{-51.994363}$
0x22822a0200000000	$2^{-51.994366}$
0x82288aa200000000	$2^{-51.994366}$
0xbbcbb70b00000000	$2^{-51.994366}$
0xcbbcc77b00000000	$2^{-51.994366}$
0x88288a0800000000	$2^{-51.994368}$
0x28822aa800000000	$2^{-51.994368}$
0xcccbc70b00000000	$2^{-51.994368}$
0xcbcbc77c00000000	$2^{-51.994368}$
0xbbbcb70c00000000	$2^{-51.994368}$
0xbcbcb77b00000000	$2^{-51.994368}$
0x22282a0800000000	$2^{-51.994368}$
0x28282aa200000000	$2^{-51.994368}$
0xccbcc70c00000000	$2^{-51.994368}$
0xbccbb77c00000000	$2^{-51.994368}$
0x8822800200000000	$2^{-51.994368}$
0x2288200800000000	$2^{-51.994368}$
0x822280a800000000	$2^{-51.994368}$
0x288820a200000000	$2^{-51.994368}$
0xccbbc00b00000000	$2^{-51.994368}$
0xbbccb00c00000000	$2^{-51.994368}$
0xcbbbc07c00000000	$2^{-51.994368}$
0xbcccb07b00000000	$2^{-51.994368}$

Table 7: Candidate input selection masks for MIBS