On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN

Gregor Leander

DTU Mathematics
Technical University of Denmark
G.Leander@mat.dtu.dk

Abstract. We discuss complexities of advanced linear attacks. In particular, we argue why it is often more appropriate to examine the median of the complexity than the average value. Moreover, we apply our methods to the block ciphers PUFFIN and PRESENT. For PUFFIN, a 128 bit key cipher, we present an attack which breaks the cipher for at least a quarter of the keys with a complexity less than 2^{58}. In the case of PRESENT we show that the design is sound. The design criteria are sufficient to ensure the resistance against linear attacks, taking into account the notion of linear hulls. Finally, we show that statistical saturation attacks and multi dimensional linear attacks are almost identical.

1 Introduction

Block ciphers are probably one of the most studied objects in cryptography in general. The security of block cipher seems well understood and quite a number of secure and efficient block ciphers are available today. The AES is of course the most studied and analyzed block cipher at present, but many other interesting proposals have been made. Recently, there has been a trend to design block ciphers that are not suitable for every environment, but rather tailored to special platforms and/or purposes. What all those designs have in common is that nowadays a detailed analysis against known cryptanalytic methods is almost mandatory when presenting a new design.

One of those known attacks is of course Matsui's linear attack [1]. However, despite its discovery more than 15 years ago, linear cryptanalysis seems to be less understood in comparison to, for example, differential attacks. In particular, for advanced linear attacks, such as attacks using so called linear hulls or multidimensional cryptanalysis, we still do not understand completely how to estimate their running time correctly. Concerning linear attacks using linear hulls Murphy [2] points out very fundamental problems when estimating the impact of those attacks.

Besides the well known cryptanalytic methods, some new attacks appeared recently, and among those are the so called statistical saturation attack. In a nutshell, the main idea of statistical saturation attacks is to use a poor diffusion in a block cipher by fixing certain input bits in the plaintext and disregarding some

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 303–322, 2011.

of the output bits. While this method currently provides the best attacks against the lightweight block cipher PRESENT, a method to estimate its complexity correctly is missing.

This lack of a deeper understanding is especially surprising as the study of block ciphers is one of the classical fields of cryptography. One would expect that the necessary tools to precisely –and formally correctly – formulate attack complexities have been already developed. However, this is apparently not always the case.

1.1 Our Contributions

In Section 3 of this paper we discuss in detail the problems pointed out by Murphy [2] on linear hulls. However, we do not quite agree with the conclusion that linear hulls do not exist. On the contrary, we explain why linear hulls (when defined correctly) always exist and always have to be taken into account when statements about the attack complexity are made. In order to be able to make meaningful statements about the attack complexity we explain why a paradigm shift from discussing average complexities to discussing medians of complexities is necessary. As we will explain, this holds not only for attacks that make use of linear hulls, but actually also for linear attacks based on a single linear trail. We present methods on how one can, in many cases, compute good approximations of the median of the complexities.

As an example, we apply our methods to cryptanalyze the block cipher PUFFIN (see Section 4). We present an attack on full round PUFFIN, a block cipher with a 128 bit key, that allows us to recover 4 bits of the last round key for at least a quarter of the keys with a complexity below 2^{58}.

In Section 5 we use our methods to understand the resistance of PRESENT to linear cryptanalysis. Most interestingly we show that the design principle of PRESENT is sound, in the sense that any sbox and any bit permutation fulfilling the design criteria of PRESENT yield to a cipher secure against this type of linear attacks. In order to do so, we present a link between optimal bit permutations and central digraphs which is interesting in itself. Central digraphs are classical combinatorial objects (see for example [3]) and the link allows us to answer natural questions about optimal permutations. Most importantly, this link allows us to classify all optimal bit permutations. This classification then allows us to get a deeper understanding of the block cipher PRESENT.

Finally, in Section 6 we solve the problem of estimating the biases (or capacities) in statistical saturation attacks. Using a theorem on the Fourier transformations of restrictions of Boolean functions, we demonstrate that statistical saturation attacks are in principle identical to multi dimensional linear attacks. In particular, this link allows us to evaluate the bias used in statistical saturation attacks using well studied tools and well established theory, a major drawback of statistical saturation attacks so far. Furthermore, we believe that this link makes it possible to apply statistical saturation attacks to other ciphers.

2 Preliminaries

In this section, we fix our notation and recall known identities between the bias of a function, the correlation and the Fourier transformation. After doing so, we recall the basic concept of linear hulls and statistical saturation attacks.

2.1 Bias, Correlation and Fourier Transformation

We denote by $\mathbb{F}_2$ the binary field with two elements and by $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. The canonical inner product on $\mathbb{F}_2^n$ is denoted by $\langle \cdot, \cdot \rangle$, i.e.

$$\langle (a_0, \ldots, a_{n-1}), (b_0, \ldots, b_{n-1}) \rangle := \sum_{i=0}^{n-1} a_i b_i.$$

We note that all linear mappings, i.e. all linear functions, $l : \mathbb{F}_2^n \to \mathbb{F}_2$ can be described as $\ell(x) = \langle a, x \rangle$ for a suitable $a \in \mathbb{F}_2^n$. Given a vector $a \in \mathbb{F}_2^n$ we denote by $\mathrm{wt}(a)$ its *Hamming weight*, i.e. $\mathrm{wt}(a) = |\{0 \leq i < n \mid a_i = 1\}|$. Given a (vectorial Boolean) function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ the *Fourier coefficient* of F at the pair $(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^m$ is defined by

$$\widehat{F}(a, b) = \sum_x (-1)^{\langle b, F(x) \rangle + \langle a, x \rangle}.$$

Given the probability p of the linear approximation $\langle a, x \rangle$ of $\langle b, F(x) \rangle$, i.e.

$$p = \frac{\mathrm{wt}(\langle b, F(\cdot) \rangle + \langle a, \cdot \rangle)}{2^n}$$

the *bias* $\epsilon_F(a, b)$ of the linear approximation $\langle a, x \rangle$ of $\langle b, F(x) \rangle$ is defined as

$$p = \frac{1}{2} + \epsilon_F(a, b)$$

which can be rewritten as

$$\epsilon_F(a, b) = \frac{\mathrm{wt}(\langle b, F(\cdot) \rangle + \langle a, \cdot \rangle)}{2^n} - \frac{1}{2}.$$

The relation between the Fourier transformation of F and the bias of a linear approximation is derived using

$$\mathrm{wt}(\langle b, F(\cdot) \rangle + \langle a, \cdot \rangle)) = 2^{n-1} - \frac{\widehat{F}(a, b)}{2}, \tag{1}$$

which implies

$$\epsilon_F(a, b) = -\frac{\widehat{F}(a, b)}{2^{n+1}}.$$

Moreover, due to scaling reasons, it is often helpful to talk about the *correlation coefficient* of F. This is defined by

$$C_F(a, b) = 2\epsilon_F(a, b) = -\frac{\widehat{F}(a, b)}{2^n}$$

Given a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the value used to determine the complexity of both multidimensional linear attacks and statistical saturation attacks is

$$\mathrm{Cap}(F) = \sum_{y \in \mathbb{F}_2^m} \frac{(2^{-n}|\{x \in F_2^n \mid F(x) = y\}| - 2^{-m})^2}{2^{-m}}$$

which is called *capacity* in [4]. In [5] the *squared euclidian distance* was used which is defined as

$$D(F) = \sum_{y \in \mathbb{F}_2^m} \left(2^{-n}|\{x \in F_2^n \mid F(x) = y\}| - 2^{-m}\right)^2$$

and differers from $\mathrm{Cap}(F)$ by a factor of 2^m, i.e. $D(F) = 2^{-m}\,\mathrm{Cap}(F)$.

There is an important, and well known, relation between the capacity (or the squared Euclidian distance) and the Fourier transformation of F which we will use below (see for example [6]).

Lemma 1

$$\mathrm{Cap}(F) = 2^{-2n} \sum_{b \neq 0} \left(\widehat{F}(0,b)\right)^2 = \sum_{b \neq 0} \left(\widehat{C}_F(0,b)\right)^2$$

2.2 Linear Trails, Correlations and Linear Hull

Consider a mapping $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ given as the composition of mappings, i.e. $F = F_n \circ F_{n-1} \circ \cdots \circ F_1$. The correlation $C_F(a,b)$ can in this case be computed using linear trails. A linear trail consists of an input mask a and output mask b and a vector $U = (u_1, \ldots, u_{r-1})$ with $u_i \in \mathbb{F}_2^n$. The correlation of the trail is defined as

$$C_F(a,b,U) = C_{F_1}(a,u_1)C_{F_2}(u_1,u_2)\cdots C_{F_{r-1}}(u_{r-2},u_{r-1})C_{F_r}(u_{r-1},b).$$

Now, using correlation matrices [7] or the Fourier transformation of composite mappings [8], one can prove that

$$C_F(a,b) = \sum_{U \in (\mathbb{F}_2^n)^{r-1}} C_F(a,b,U).$$

In contrary to the piling-up lemma [1], no assumption of any kind has to be made for this equation to hold. In the case where F corresponds to a key-alternating iterative block cipher, that is when all F_i are the same up to an addition of a round key, one can rewrite the previous equation as

$$C_F(a,b) = \sum_{U \in (\mathbb{F}_2^n)^{r-1}} (-1)^{s_U} |C_F(a,b,U)|, \tag{2}$$

where the signs $s_U \in \{0,1\}$ depend on the sign of $C_F(a,b,U)$ and on the round keys. More importantly, the value $|C_F(a,b,U)|$ is independent of the round keys and the only influence of changing the keys is a change of the signs s_U. Again, no assumption is necessary for Equation 2 to hold. What we understand as the *linear hull* is in fact nothing other than Equation 2.

An assumption that will be necessary to understand the distribution of the biases in a linear attack is the following.

Assumption 1. *The signs s_U in Equation 2 are independently and uniformly distributed with respect to the key.*

Note that assuming independent round keys does not necessarily imply Assumption 1. This is only the case when all trails U with non-zero correlation $C_F(a,b,U)$ are linearly independent. In any case, it is important to verify experimentally for each cipher at hand, that Assumption 1 holds, before it can be applied.

Another important result we are going to use (cf. Theorem 1 in [6] and Theorem 7.9.1 in [7]) is the following.

Proposition 1. *Let F be the encryption function of a key alternating block cipher and assume that all round keys are independent. The average squared bias (resp. correlation) between an input and an output mask is the sum of the squared biases (resp. correlations) over all linear trails between the input and the output mask, i.e.*

$$\frac{1}{|K|} C_F(a,b)^2 = \sum_{U \in (\mathbb{F}_2^n)^{r-2}} C_F(a,b,U)^2$$

2.3 Statistical Saturation Attacks

In this section we briefly outline the idea of statistical saturation attacks. We refer to [5] for details. Given an encryption function

$$e : \mathbb{F}_2^n \to \mathbb{F}_2^n$$

statistical saturation attacks study the distribution of e when some of its inputs are fixed. While in general one can imagine the restriction to the coset of any subspace $E \subset \mathbb{F}_2^n$ for the inputs and any subspace $E' \subset \mathbb{F}_2^n$ for the output, for simplicity we restrict ourselves to the case where one fixes the last s bits in the inputs and considers only the first t bits of the output. Thus we write

$$e : \mathbb{F}_2^r \times \mathbb{F}_2^s \to \mathbb{F}_2^t \times \mathbb{F}_2^u \tag{3}$$

$$e(x,y) = \left(e^{(1)}(x,y), e^{(2)}(x,y)\right) \tag{4}$$

where $r+s = t+u = n$ and $e^{(1)}(x,y) \in \mathbb{F}_2^t$, $e^{(2)}(x,y) \in \mathbb{F}_2^u$. For convenience we denote by h_y the restriction of e by fixing the last s bits to y and considering only the first t bits of the output, that is

$$h_y : \mathbb{F}_2^r \to \mathbb{F}_2^t$$
$$h_y(x) = e^{(1)}(x, y) \tag{5}$$

In a statistical saturation attack one considers the capacity of h_y, and the attack complexity is usually a constant times $1/\operatorname{Cap}(h_y)$.

Applying Lemma 1 to h_y we can rewrite the capacity of h_y in terms of Fourier coefficients of h_y.

$$\operatorname{Cap}(h_y) = 2^{-2r} \sum_{b \in \mathbb{F}_2^t} \left(\widehat{h}_y(0, b)\right)^2 = \sum_{b \in \mathbb{F}_2^t} \left(C_{h_y}(0, b)\right)^2 . \tag{6}$$

One fundamental problem in statistical saturation attacks is that a useful method to estimate this capacity was missing. However, in Section 6 we show that in fact statistical saturation attacks are closely related to multi-dimensional linear attacks and in particular this link provides the missing method to estimate the capacity of h_y.

3 On the Linear Hull Effect

Linear hulls have been studied already in [9] and since then have been used in a number of papers. The main idea is to consider several, sometimes a lot of, linear trails with the same input and output mask to decrease the complexity of linear attacks using Matsui's Algorithm 2. However, as Murphy [2] pointed out nicely, there are (at least) two problems often appearing in the literature. In this section we first recall those two very fundamental problems, and discuss their impact on the complexity of linear attacks. While our starting point is clearly the work of Murphy, our conclusions are quite orthogonal. More precisely, we think that a better statement than Murphy's conclusion that *there is no linear hull effect* is that *there is always a linear hull effect and we always have to deal with this*. We furthermore propose methods that allow us to make meaningful statements about the running time of linear attacks. In particular we show the following.

Theorem 2. *Under Assumption 1 in a linear attack using a single trail with squared bias ϵ^2, at least half of the keys yield to a squared bias of at least ϵ^2. Thus, the complexity of this linear attack is less than c/ϵ^2 in more than half of the cases, where c is a small constant.*

In a linear attack using many linear trails with the same squared bias $\epsilon_i^2 = \epsilon^2$ at least one quarter of the keys yield to a squared bias of at least $0.46 \cdot \sum_i \epsilon_i^2$. Thus, the complexity of this linear attack is less than $2.2c/(\sum_i \epsilon_i^2)$ in more than a quarter of the cases, where c is a small constant.

The first problem Murphy points out comes from an incorrect formalization. Consider the simplest case of two linear trails with the same input and output mask, but different intermediate masks. The piling-up lemma is used to estimate the bias (say ϵ_1, ϵ_2) for each of the equations. We assume that $\epsilon_1 \neq 0$ and $\epsilon_2 \neq 0$.

Denoting by α the input mask, by β the output mask and by γ_1, γ_2 the two key masks, we end up with the following two equations

$$\langle \alpha, p \rangle + \langle \beta, c \rangle = \langle \gamma_1, K \rangle \text{ and } \langle \alpha, p \rangle + \langle \beta, c \rangle = \langle \gamma_2, K \rangle$$

where the first equation holds with a bias ϵ_1 and the second with a bias ϵ_2. So far, so good, but now consider any fixed (extended) key K. There are mainly two cases to consider. First, one could have a key such that $\langle \gamma_1, K \rangle = \langle \gamma_2, K \rangle$. In this case one gets the same equation twice, implying that $\epsilon_1 = \epsilon_2$. On the other hand, a different key could yield $\langle \gamma_1, K' \rangle = \langle \gamma_2, K' \rangle + 1$, and in this case we conclude that $\epsilon_1 = -\epsilon_2$. As it is very likely that there is at least one key for each of the cases, the only possible choice for the biases is $\epsilon_1 = \epsilon_2 = 0$, contradicting our assumption. What went wrong? Where does the mistake come from? The main point here is that, by applying the piling-up lemma, one implicitly assumes independent and uniform distribution in each round. However, having the first trail at hand, we already know that the inputs are non-uniformly distributed, and thus this assumption is wrong. It is important to notice the difference to the approach using correlation matrices, that is Equation 2. *Here no assumption about independent or uniform distribution is involved.* Thus, one actually always deals with the expression

$$C_F(a, b) = \sum_{U \in (\mathbb{F}_2^n)^r} C_F(a, b, U).$$

but separating the different trails is not possible. This is why the correct conclusion is that there is always a linear hull effect and we have to deal with this.

So far, we recaptured the first of the two problems pointed out by Murphy, and apply the (known) theory of correlation matrices to overcome it. Now, let us have a closer look at Murphy's second point, which we present slightly differently. Assume someone found *one* linear trail with a non-zero bias ϵ. Usually, the conclusion is that an attack based on this bias (lets say using Matsui second algorithm) is a constant times $1/\epsilon^2$. But what if the attacker overlooked another trail with the same absolute bias. The correct correlation is given by

$$C_F(u, v) = ((-1)^{s_1} + (-1)^{s_2})2\epsilon,$$

where the value of s_1 and s_2 depend on the extended key. Then, assuming a random behavior of the signs, the total bias would be zero for half of the keys.

There are two important remarks to make. First, the absolute bias is actually key dependent and secondly *the average complexity is formally infinite.* Again, to make the point very clear, the original *attack did not take any linear hull effect into account and this is the reason why the claim about the attack complexity is wrong.*

Now the attacker tries to do better and can actually show that all other trails have a (much) smaller bias. Still, for some (maybe only one) keys the biases could cancel and the average complexity is again infinite.

Thus, due to the linear hull effect, which is always there, estimating the average complexity of the attack seems very difficult (and often turns out to be infinite).

In the example, where there are exactly two trails with the same bias, the attack will still work for half of the keys (and for this half even faster than estimated by looking at only one trail). This leads directly to a natural way to overcome this problem. Namely, instead of studying the average complexity, the median of the complexities should be studied.

Definition 1. *The median of the complexities $\tilde{C}$ is the value such that, for half of the keys the complexity of the attack is less than or equal to $\tilde{C}$. More generally, one could study the complexity C_p defined as the complexity such that the probability that for a given key the attack complexity is lower than C_p, is p.*

Note that $\tilde{C} = C_{1/2}$.

Studying the complexity C_p instead of the average complexity has several advantages. First, given the median of the biases $\tilde{\epsilon}$, the median of the complexity is simply $c/\tilde{\epsilon}^2$ (as the inverse is a monotone function). Secondly, the median (or general C_p) is actually a more interesting value to know than the average complexity, especially in the case where the latter is infinite.

Finally, let us see what happens when we ignore or overlook trails in the computation of $C_F(u, v)$. This was exactly where the trouble started for the average complexity. Let us assume we take n trails with correlations γ_i into account. Furthermore, let us denote by γ_p the correlation such that the probability that a given key yields a correlation larger or equal to γ_p is p, given the n trails. That is

$$\mathrm{Prob}_K\left(\left|\sum_i (-1)^{s_i}\gamma_i\right| > |\gamma_p|\right) = \frac{1}{2}.$$

Denoting the correlations of the remaining trails by η_j, we get

$$C_F(u, v) = \left(\sum_i (-1)^{s_i}\gamma_i\right) + \left(\sum_j (-1)^{s'_j}\eta_j\right).$$

With a probability of 1/2 the sum $\sum_j (-1)^{s'_j}\eta_j$ has the same sign as the sum $\sum_i (-1)^{s_i}\gamma_i$. Thus

$$\mathrm{Prob}_K(|C_F(u, v)| > |\gamma_p|) \geq \frac{p}{2}. \tag{7}$$

This inequality implies that the probability of having a complexity less than a given bound, might actually be smaller than estimated due to linear trails that have not been taken into account. However, this probability drops by at most a factor of 2.

Coming back to the case where an attacker just considered one trail with bias ϵ. As we saw, it is not possible to conclude *anything* meaningful about the average, but using the above considerations, one can conclude that for at least half of the keys the data complexity is below c/ϵ^2.

One important point not discussed so far is how to estimate the medians of the correlations. Here we consider two cases. First, in an attack where only a

few trails are used, one can easily compute the median by running through all possible values for the signs. This gets infeasible when there are too many trails. However, in the case where one deals with many trails with the same absolute correlation, one can estimate the median nicely by using a normal approximation, as explained below. In Section 4 we furthermore show by example, how one can estimate the median in the case of many trails with different absolute values.

3.1 Many Trails with the Same Absolute Value

As already done before (see [10]) in the case of many linear trails with the same absolute value, the distribution of the correlation

$$C_F(a,b) = \sum_{U \in (\mathbb{F}_2^n)^{r-2}} (-1)^{s_U} |C_F(a,b,U)| = \sum_i (-1)^{s_i} 2\epsilon_i,$$

where ϵ_i are the absolute biases of the trails, can be approximated by a normal distribution. This approximation implicitly makes use of Assumption 1. Clearly, this assumption has to be justified for each cipher by experiments (and we do so below for the block cipher PUFFIN). Denoting by X the random variable corresponding to the bias, we will approximate its distribution by

$$X \sim \mathcal{N}(0, \sum \epsilon_i^2),$$

that is, X is normally distributed with mean zero and variance $\sigma^2 := \sum \epsilon_i^2$. The probability density function is thus given by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}x^2}.$$

Again, when trying to compute the average complexity of the attack, that is, the mean value of the random variable c/X^2, it turns out that this value is formally infinite. Therefore, and for reasons outlined above, we focus on the median of the squared biases corresponding to the random variable $Y := X^2$. Denoting by F the cumulative distribution function of X, the cumulative distribution function G of $Y = X^2$ can be computed as

$$\begin{aligned} G(t) &= \text{Prob}(Y \leq t) = \text{Prob}(-\sqrt{t} \leq X \leq \sqrt{(t)}) \\ &= F(\sqrt{t}) + F(-\sqrt{t}) - 1 = 2F(\sqrt{t}) - 1. \end{aligned}$$

Using the relation to the normal distribution (or ask maple) we can simplify $G(t)$ to

$$G(t) = \text{erf}(\sqrt{\frac{t}{2\sigma^2}})$$

where erf is the Gauss error function. The median $\tilde{\epsilon}^2$ of Y is by definition the value $\tilde{\epsilon}^2$ such that $G(\tilde{\epsilon}^2) = \frac{1}{2}$. We get

$$\text{erf}(\sqrt{\frac{\epsilon_m^2}{2\sigma^2}}) = \frac{1}{2}$$

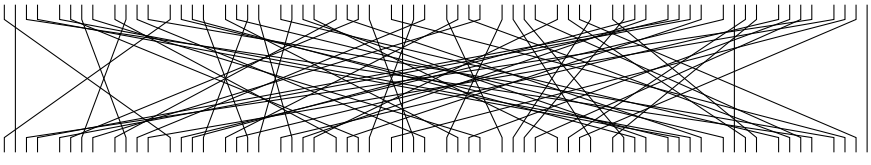
and using the approximation $\text{erf}^{-1}(1/2) \approx 0.48$ we conclude that $\epsilon_m^2 \approx 0.46\sigma^2$.

For completeness, we can furthermore compute the mean of Y as $E[Y] = \sigma^2$ which naturally corresponds to Proposition 1 (without using the normal approximation).

Thus, using the normal approximation and Equation 7 we can conclude that for a quarter of the keys the attack has a complexity lower than a small constant times $1/(0.46\sigma^2) \approx 2.21/\sigma^2$. A similar calculation shows that fraction of approximately 0.317 of the keys lead to an attack complexity of a small constant times σ^2.

4 Linear Hulls and PUFFIN

This section applies the ideas outlined above to the block cipher PUFFIN [11]. PUFFIN, a very PRESENT like SP-network, is a 64 bit block cipher with 32 rounds and a 128 bit master key. The only components we are interested in here are the linear layer and the sbox-layer. The linear layer is the following bit permutation.



The sbox-layer consists of 16 parallel executions of a single 4 bit sbox given by the following table.

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

The main difference to PRESENT is that all components are involutions, thus allowing to save area when implementing the decryption circuit. For more details on PUFFIN we refer to [11].

We show how one can estimate quite precisely the distribution of the biases and thus in particular estimate the median of the attack complexity. Applying Theorem 2, our results indicate that, for at least a quarter of the keys, PUFFIN can be broken with a complexity less than 2^{58}. Because everything in our attack, except the estimation of the attack complexity, is a standard application of Matsui's second algorithm, we skip some details of the attack.

4.1 Linear Trails in PUFFIN

We focus only on trails, where all intermediate masks have Hamming weight one, i.e we have exactly one active sbox in each round. As it turns out, the highest median of biases can be expected for the input and output mask $e_1 =$ `0x2000000000000000`, that is between the first (counting from zero) bit of the plaintext and the first bit of the ciphertext. The number of all such trails together

with their absolute squared correlation can easily be computed for up to 31 rounds. Those results are summerized below. An entry t in this table at level ℓ and round r means that there are 2^t one bit trails, each with an absolute bias of 2^{-r-l}. Note that there is exactly one trail with maximal absolute bias of 2^{-r} and this trail is not included in the table.

Round\Level	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
4	2.00	-	1.00	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	2.81	-	2.58	-	-	-	-	-	-	-	-	-	-	-	-	-	-
6	3.32	-	4.00	2.00	1.00	-	-	-	-	-	-	-	-	-	-	-	-
7	3.81	-	5.13	3.32	4.09	2.58	-	-	-	-	-	-	-	-	-	-	-
8	4.17	-	6.00	4.58	5.95	4.46	3.32	-	-	-	-	-	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
28	7.70	-	13.41	12.69	17.88	18.04	21.51	22.05	24.47	25.18	26.89	27.63	28.84	29.50	30.34	30.87	31.41
29	7.80	-	13.61	12.90	18.19	18.36	21.91	22.48	24.98	25.72	27.52	28.30	29.58	30.30	31.22	31.81	32.43
30	7.89	-	13.80	13.09	18.47	18.66	22.30	22.89	25.48	26.25	28.12	28.94	30.30	31.07	32.06	32.71	33.41
31	7.99	-	13.99	13.29	18.76	18.95	22.67	23.28	25.95	26.75	28.70	29.55	30.99	31.80	32.86	33.57	34.34

4.2 Approximation of the Bias Distribution

In the case of PUFFIN we use a normal approximation to approximate the many linear trails for levels greater than 1. Denote by σ^2 the sum of squares of all those biases, i.e.

$$\sigma^2 = \sum_i \epsilon_i^2,$$

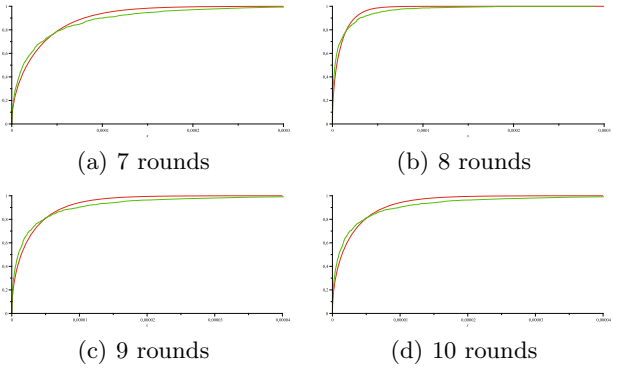
where ϵ_i runs through all one bit linear trails of level greater than 1 and smaller than 19. We denote the bias of the unique trail with maximal absolute bias by η. In order to incorporate this one maximal biased trail as well, we expect that the cumulative distribution function G of the total bias as is given by

$$G(t) = \frac{1}{2}\left(F(t-\eta) + F(t+\eta)\right), \tag{8}$$

where F is the cumulative distribution function of $\mathcal{N}(0, \sigma^2)$.

In order to justify the approximation, that is the implicit assumption on the random behavior of the signs, we experimentally compute the bias for 1000 keys for rounds 7 to 10. The results are shown in Figure 1.

Note that one reason for the small difference is that the estimates for very small biases are wrong, due to a naturally limited amount of samples. Apart from this small error, the distributions are quite close and in particular the median is predicted very precisely. Using Equation 8 one can easily compute the median numerically (using for example Maple). It turns out that the base two logarithm of the medians of the squared biases is almost an affine function. A good approximation of the median of the square bias for r rounds is given by $2^{-1.71r-3.13}$. In particular, applying Theorem 2 this implies that for a quarter of the keys, the data complexity of attacking r rounds of PUFFIN is proportional to $2^{1.71(r-1)+3.13}$. Experimental results for 7 to 12 round attacks (again using 1000 randomly chosen keys per round) indicate that using $4 \cdot 2^{1.71(r-1)+3.13}$ gives full gain, that is it recovers four key bits of the last round key successfully, in over 40% of the cases. In particular for $r = 32$, that is for the full PUFFIN, we get a complexity of about 2^{58}.



(a) 7 rounds (b) 8 rounds

(c) 9 rounds (d) 10 rounds

Fig. 1. Theoretical estimates vs. experimental bias for 7 to 10 rounds. The experimental data is based on 1000 randomly chosen keys for each round.

We expect that this is not the optimal attack. For example partial decryption of two instead of only one round might further reduce the data complexity (at the cost of increasing the computational complexity). Another improvement is likely obtained by applying the below mentioned statistical saturation attack (however estimating the exact data requirements is difficult). As the main objective of this section was to demonstrate how one can estimate the distribution of biases in the case of many linear trails, and use this finally to allow meaningful statements about the behavior of a linear attack, those improvements are out of scope of this paper and we leave them as a topic for further investigation.

5 Linear Hulls and PRESENT

PRESENT is a 64-bit block cipher developed by A. Bogdanov et al. [12] and was designed to be particular suitable for low-cost devices like RFID-tags. There are two versions, a 80 bit key version, called PRESENT-80 and a 128 bit version PRESENT-128. PRESENT is an substitution-permutation-network with 31 rounds and one final key exclusive-or at the end.

In the substitution layer (called sBoxLayer in PRESENT) a single 4-bit to 4-bit sbox is applied 16 times in parallel. The action of the sbox in hexadecimal notation is given by the following table.

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

The permutation layer (called pLayer in PRESENT) is given as a simple bit permutation. Bit i of the current state is moved to bit position $P(i)$, where

$$P(i) = \begin{cases} 16 \times i \bmod 63 & \text{for } 0 \leq i \leq 62 \\ 63 & \text{for } i = 63 \end{cases}$$

The sbox in PRESENT $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ has been chosen to fulfill several criteria (see [12] for details) to ensure resistance against differential and linear cryptanalysis. We call a sbox fulfilling these criteria optimal.

Compared to a general linear layer the permutation layer of PRESENT provides relatively low diffusion. However, as stated in [12], for a bit permutation the pLayer is optimal in the sense that full dependency is reached already after a minimal number of rounds. After three rounds each of the 64 input bits influence each of the 64 output bits. For convenience we call such a permutation optimal.For more details on PRESENT we refer to [12].

5.1 Linear Attacks on PRESENT

Several papers discussed linear attacks on PRESENT. In [10] linear hulls where used to attack up to 25 rounds. Moreover, in [6] a multi dimensional linear attack on up to 26 rounds of PRESENT is described. The main reason why linear cryptanalysis works against up to 26 rounds of PRESENT is the relative high number of linear trails with only one active sbox per round. As the number of those linear trails was not discussed in [12] an important question is how sound the design of PRESENT is, when taking linear hulls into account. For $a, b \in \{0, \ldots, 63\}$ we denote by $N(a, b)$ the number of linear trails starting with the input bit a, ending with the output bit b and with exactly one active sbox per round.

5.2 On the Choice of Sbox and Permutation in PRESENT

Here we are interested in understanding the influence of the choice of the sbox and the bit permutation on the maximal number of trails $N(a, b)$. As explained below, using classification results on optimal sboxes and central digraphs, we can conclude that

- Given the PRESENT bit permutation, the PRESENT sbox is among the 8% worst of all optimal sboxes.
- Given the PRESENT sbox, the PRESENT bit permutation was the worst among roughly 2^{21} optimal permutations tested.

It should be noted that the PRESENT bit permutation is a natural choice, also reflected by the fact that it corresponds to what is known as the standard example in central digraphs (see below). However, the sbox in PRESENT was selected among all possible optimal sboxes as one with the smallest hardware circuit. We leave it as a topic for further investigation to explore, if there is a correlation between the size of the hardware circuit and the number of trails.

Furthermore, using those classification results we conclude that there is not a particular good bit permutation nor a particular good sbox. The number of one bit linear trails is mainly determined by the combination of both.

Most importantly, the results outlined below imply that the design principle of PRESENT is sound, in the following sense.

Fact 3. *For no combination of an optimal sbox and an optimal bit permutation, a linear attack based on one bit trails seems possible on* 31 *rounds.*

Influence of the Sbox. In this section we fix the bit permutation to the one used in PRESENT and compute the maximal number of trails for various choices of the sbox. Up to adding constants before and after the sbox, which clearly does not change any of these criteria and furthermore does not change the number of linear one bit trails, there are exactly 8064 such sboxes (see for example [13]). For each possible sbox fulfilling these criteria we computed the maximal number of one bit trails over all input/output bit combination for 31 rounds. It turns out that there are only 31 possible values for this maximum, ranging from 0 to approx 2^{39}. In Table 1 the number of sboxes (out of the 8064 possible ones) for each of the 30 possible values for the maximum of trails is shown. The PRESENT sbox has a maximal trail number of approx. $2^{38.17}$ and thus is among the 8 percent worst optimal sboxes with this respect.

Table 1. Maximal number of trails ($\log_2 \max_{a,b} N(a,b)$) vs number of sboxes (out of 8064) with the given trail number. In all cases the PRESENT bit permutation was used.

Maximal Number of Trails	$-\infty$	0	3.000	6.965	7.754	9.509	9.965	10.75	12.26
Number of sboxes	96	1056	192	48	48	192	768	48	48
Maximal Number of Trails	15.00	16.47	17.42	19.42	21.47	21.71	22.86	24.29	25.25
Number of sboxes	144	48	48	816	96	48	48	96	864
Maximal Number of Trails	25.30	25.54	25.75	26.03	26.04	26.08	26.33	26.34	26.37
Number of sboxes	96	96	192	96	96	96	96	96	96
Maximal Number of Trails	26.60	27.00	38.17	39.47					
Number of sboxes	96	1728	384	192					

Influence of the Bit Permutation. Different choices of an optimal bit permutation might result in different resistance against linear attacks. Below, we discuss the influence of the bit permutation on the number of one bit linear trails.

The first problem one encounters is, that it is actually not straight forward to find a permutation of 64 bits that, in a PRESENT style SP-network, yield to full dependency after three rounds. Optimal permutations are very rare and a naive trial to construct such objects is likely to fail. However, there is an interesting link to well studied objects in graph theory, namely central digraphs, that allows us to overcome this obstacle.

Definition 2. *Let n be an integer and D be a directed graph with n vertices. D is said to be a central digraph if there is a unique oriented path of length two between any two of its vertices.*

It is known (see [3]) that central digraphs exist only for $n = k^2$ and necessarily every vertex has in and out degree 4.

Any optimal bit permutation gives rise to a central digraph as follows. Thinking about the 16 sboxes as 16 vertices, we add a directed arc from vertex i to

vertex j if and only if there is an output bit of sbox i that gets mapped to an input bit of sbox j. Clearly, each of the 16 vertices has in and out degree 4. Being an optimal bit permutation now translates to the fact that each vertex (that is each sbox) can be reached from each vertex (that is any other sbox) in exactly two steps. A counting argument shows that such a path has to be unique and therefore the resulting graph is indeed a central digraph.

On the other hand, the converse construction, works as well. That is, given a central digraph of order 16 we can easily construct a optimal bit permutation. Note that this correspondence is unique only up to a permutation of the input and output bits of each sbox, as clearly permuting the input and output bits of an sbox does not change the corresponding graph (neither the optimality of the permutation). We thus have the following theorem.

Theorem 4. *Up to a permutation of the input and output bits of each sbox there is a one to one correspondence between optimal permutations of* 64 *bits and central digraphs of order* 16*.*

It is interesting to note that the PRESENT bit permutation actually corresponds to what is known as the "standard example" in terms of central digraphs. The vertex set of the standard example consists of all pairs (x, y) with $1 \leq x, y \leq k$ and we let $(x, y) \to (x', y')$ precisely when $y = x'$.

Using this link a couple of interesting questions can be easily answered. For example, one might want to avoid optimal permutations where a bit coming from one sbox gets mapped to the same sbox in the next round. However, it is well known (see for example [3]) that every central digraph on k^2 vertices has exactly k loops. This translate to the property that an optimal permutation on 64 bits will map exactly 4 input bits (coming from 4 distinct sboxes) back into the source sbox. Furthermore, for implementation reasons, one might want to chose a optimal permutation that is an involution. Again, it follows from the theory of central digraphs, that such a permutation does not exist.

For our purpose, the most interesting fact about central digraphs is that a classification of all central digraphs of order 16 is known (see [14]) and the actual number of (non-isomorphic) central digraphs of order 16 is reasonable small. Up to isomorphism there are precisely 3492 central digraphs of order 16. Thus, this link allows us to compare a great variety of choices for the optimal bit permutation.

We fixed the sbox to the one specified by PRESENT and computed the number of trails for all the 3492 possible central digraphs. Here, we randomly assigned the 4 incoming vertices and the 4 outgoing vertices to input and output bits of the sbox in 1000 different ways for each of the 3492 central digraphs. The result is shown in Table 2.

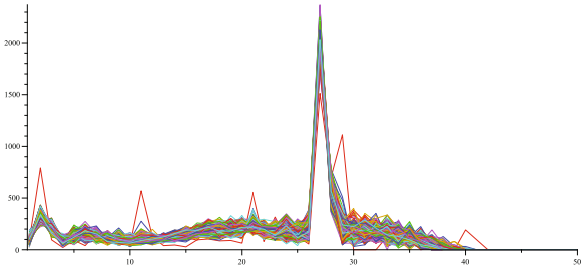
Again, the PRESENT permutation gives a maximal number of trails of approx $2^{38.17}$ and is therefore the worst of all $3492 \cdot 1000 \approx 2^{21}$ cases.

Influence of both Components. In a next step, instead of fixing the sbox and varying the permutation we varied both, that is we run through all 8064 possible sboxes and all 3492 possible central digraphs, again with 1000 randomly

Table 2. Maximal number of trails ($\lceil \log_2 \max_{a,b} N(a,b) \rceil$) vs number of optimal permutations (out of $3492 \cdot 1000$) with the given trail number. In all cases the PRESENT sbox was used.

♯Trails	1	2	3	4	5	6	7	8	9	10
♯permu.	2	5	7	8	16	21	46	51	86	144
♯Trails	11	12	13	14	15	16	17	18	19	20
♯permu.	234	351	559	990	1780	3260	5951	11187	21033	39284
♯Trails	21	22	23	24	25	26	27	28	29	30
♯permu.	71712	125520	205411	313402	431188	524990	553858	494911	359864	205508
♯Trails	31	32	33	34	35	36	37	38		
♯permu.	87875	26257	5344	941	184	18	1	1		

chosen permutations for the input and output bits of each sbox. As this is far too much data to be included in the paper, we only give parts of in the picture below.



There are two important observations to make. First, all curves follow pretty much the same pattern. This is to say there is no specially good or bad choice of sbox or bit permutation, only the combination of both can be good or bad. Second, in non of the $3492 \cdot 8064 \cdot 1000 \approx 2^{34}$ cases the number of trails was high enough to allow a linear attack based on one bit trails for 31 rounds.

6 Understanding Statistical Saturation Attacks

In this section we show how the capacity of statistical saturation attacks can be explained using tools from linear cryptanalysis. The main technical ingredient is an identity between the Fourier transform of a Boolean function and the biases of its restrictions (cf. Theorem V.1 in [15], see also Proposition 9 in [16])

Proposition 2 (Theorem V.1 in [15]). *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. Furthermore, let E and E' be subspaces of $\mathbb{F}_2^n$ such that $E \cap E' = \{0\}$ and whose direct sum equals $\mathbb{F}_2^n$. For every $a \in \mathbb{F}_2^n$ let h_a be the restriction of f to the coset $a + E$ (h_a can be identified with a function on $\mathbb{F}_2^k$ where k is the dimension of E. Then*

$$\sum_{u \in E^{\perp}} \left(\widehat{f}(u)\right)^2 = |E^{\perp}| \sum_{a \in E'} \left(\widehat{h}_a(0)\right)^2. \tag{9}$$

Here $E^{\perp}$ is the orthogonal space of E.

Recall that we consider the encryption function $e : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and its restrictions by fixing the last s bits of the input and considering only the first t bits of its output, that is the function $h_y(x)$ defined by Equation 5. For statistical saturation attacks we are interested in the capacity given by Equation 6. Using the proposition above, we now state the main result of this Section.

Theorem 5. *With the above notation, the average capacity in statistical saturation attacks where the average is taken over all possible fixations is given by*

$$\begin{aligned}\overline{\mathrm{Cap}(h_y)} &= 2^{-s} \sum_{y \in \mathbb{F}_2^s} \mathrm{Cap}(h_y) = 2^{-2n} \sum_{a \in \{0\} \times \mathbb{F}_2^s, b \in \mathbb{F}_2^t \times \{0\}} \left(\widehat{e}(a,b)\right)^2 \\ &= \sum_{a \in \{0\} \times \mathbb{F}_2^s, b \in \mathbb{F}_2^t \times \{0\}} \left(C_e(a,b)\right)^2.\end{aligned}$$

Proof. By definition

$$\overline{C(h_y)} = 2^{-s} \sum_{y \in \mathbb{F}_2^s} \mathrm{Cap}(h_y) = 2^{-s} \sum_{a \in \{0\} \times \mathbb{F}_2^s, b \in \mathbb{F}_2^t} 2^{-2r} \left(\widehat{h}_a(0,b)\right)^2 \tag{10}$$

Applying identity (9) to all component function $\langle b, e\rangle$ (and its restrictions $\langle b, h_y\rangle$) where we choose $E = \mathbb{F}_2^r \times \{0\}$ and $E' = E^{\perp} = \{0\} \times \mathbb{F}_2^s$ yields

$$\sum_{u \in \{0\} \times \mathbb{F}_2^s} \left(\widehat{e}(u,b)\right)^2 = 2^s \sum_{a \in \{0\} \times \mathbb{F}_2^s} \left(\widehat{h}_a(0,b)\right)^2$$

Using this we deduce from (10).

$$\overline{\mathrm{Cap}(h_y)} = 2^{-2s-2r} \sum_{u \in \{0\} \times \mathbb{F}_2^s, b \in \mathbb{F}_2^t \times \{0\}} \left(\widehat{e}(u,b)\right)^2$$

as claimed. □

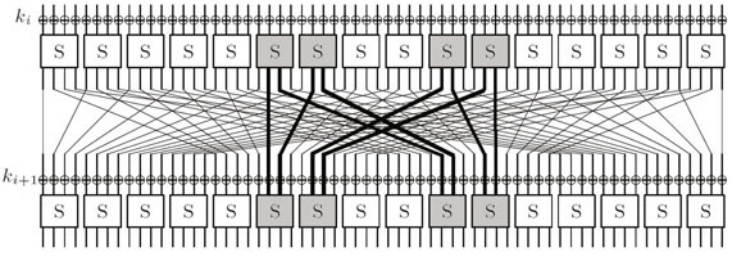
In general, statistical saturation attacks work well if one can identify subspaces $U, U'' \in \mathbb{F}_2^n$ (where U corresponds to output masks and U' to input masks) such that the sum $\sum_{u \in U', U \in E} \left(\widehat{e}(u,b)\right)^2$ is big. Moreover, Theorem 5 allows us to estimate the capacity, which is a first step in estimating the attack complexity of a statistical saturation attack.

From this point of view, statistical saturation attacks are very closely related to multi dimensional linear attacks. Especially, the statistical saturation attack on PRESENT presented in [5] and the multi dimensional linear cryptanalysis on PRESENT presented in [6] are in principle the same attack.

6.1 Statistical Saturation Attacks on PRESENT

For a description of PRESENT we refer to Section 5 and for more details to [12].

In this section we take a closer look at the statistical saturation trails used in [5] and explain its capacity using the above link to linear attacks. A picture of the trail used in [5] is given below.



In this trail the bits (counting from right to left, starting with 0)

$$S = \{21, 22, 25, 26, 37, 38, 41, 42\}$$

are fixed. At the output the same set of bits is used to compute the bias. In light of Theorem 5 this corresponds to taking $e : \mathbb{F}_2^{64} \to \mathbb{F}_2^{64}$ and its restrictions by fixing the 8 bits of the trail and restricting the output to the 8 bits in the trail as well $h_y : \mathbb{F}_2^{56} \to \mathbb{F}_2^8$.

Defining $E = \text{span}\{e_i \mid i \in S\}$, where $e_i \in \mathbb{F}_2^{64}$ is the canonical basis vector with a single one at position i (counting from zero), Theorem 5 states that

$$\overline{\text{Cap}(h_y)} = \sum_{a,b \in E} \left(C_e(a,b)\right)^2 .$$

To compute this capacity, we have to compute the correlation coefficients $C_e(a,b)$ for $a, b \in E$.

Like in Section 5 we restrict to a, b of weight one. This was done as well in [10,6], the argument being that it can be expected that those correlation coefficients have a much higher absolute value. Again, this assumptions is confirmed by the experimental data, see below. Given $a, b \in E$ of weight one, we recalled in Section 5 that there are many possible linear trails, starting with the input mask a and ending in the output mask b where all intermediate masks have weight one as well. Recall that we denoted the number of these linear trails by $N(a,b)$. Furthermore it is easy to compute the exact number of such paths for any pair (a,b).

The correlation $C_e(a,b,U_i)$ of the linear trails U_i, using the fact that in each round the bias is 2^{-3}, is given by $C_e(a,b,U_i) = 2^{-2R}$. Applying Proposition 1, the average square correlation is given by

$$\left(\overline{C}_e(a,b)\right)^2 = 2^{-4R} N(a,b),$$

and

$$\overline{C(h_y)} = \sum_{a,b \in E} \left(\overline{C}_e(a,b)\right)^2 \approx 2^{-4R} \sum_{\substack{a,b \in E \\ \mathrm{wt}(a)=\mathrm{wt}(b)=1}} N(a,b) \tag{11}$$

We compared experimental computations of $C(h_y)$, averaged over 100 different keys and 10 different values of y for each key with the results of the approximation (11). Except for the first two rounds the experimental date follow quite closely the approximation.

Round	2	3	4	5	6	7	8	9
$\log_2 \sum N(a,b)$	5.00	6.00	7.32	8.64	9.97	11.34	12.72	14.10
approx. (11)	−11.00	−14.00	−16.68	−19.36	−22.03	−24.66	−27.28	−29.90
experimental	−10.38	−13.82	−16.27	−18.90	−21.60	−24.13	−26.78	−29.26

The next observation that is immediate from looking at the numbers $N(a,b)$ is that this trail is likely to not be the best choice. Indeed, using the trail defined by fixing the same input bits as before, i.e. using $S = \{21, 22, 25, 26, 37, 38, 41, 42\}$ but this time restricting the output to the bits $S' = \{21, 23, 29, 31, 53, 55, 61, 63\}$ gives better results theoretically. Defining $E' = \mathrm{span}\{e_i \mid i \in S'\}$, the sum $\sum_{a \in E, b \in E'} N(a,b)$ is higher compared to the original trail (for example the capacity for 9 rounds is 2^{-29} instead of $2^{-29.9}$). Again, we verified this behavior experimentally and the experimental data confirm the approximations used quite nicely.

7 Conclusion and Further Work

We explained in detail why an estimate of the complexity of linear attacks is difficult and statements on the average complexity are often wrong. This is a very fundamental problem and we conclude that a paradigm shift from studying the average complexity to studying the median of the complexity is necessary. To simplify statements on the median, an important problem for further research is to give a general lower bound of the median in terms of the capacities of trails. In the case where the correlation for all trails have the same absolute value, this is not too difficult. However, as shown in Section 3 in this case an approximation by a suitable normal distribution provides nice results anyway.

Furthermore, we explained in Section 6 that statistical saturation attacks are almost identical to multidimensional linear attacks. This link allowed us to nicely estimate the average capacity of statistical saturation attacks. Of course, knowing only the average capacity for statistical saturation and multidimensional linear attacks suffers from the same problems as knowing the average in linear attacks. Namely, a useful statement on the running time is difficult. One important topic of further research is therefore to extend the ideas outlined in Section 3 to these cases.

Acknowledgment. The author likes to thank Sean Murphy for very valuable comments.

References

1. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
2. Murphy, S.: The Effectiveness of the Linear Hull Effect. Technical Report, RHUL-MA-2009-19 (2009)
3. Knuth, D.: Notes on central groupoids. J. Combin. Theory 8, 376–390 (1970)
4. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional extension of matsui's algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 209–227. Springer, Heidelberg (2009)
5. Collard, B., Standaert, F.X.: A statistical saturation attack against the block cipher present. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
6. Cho, J.Y.: Linear cryptanalysis of reduced-round present. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010)
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
8. Carlet, C.: Vectorial (multi-output) Boolean Functions for Cryptography. Cambridge University Press, Cambridge (to appear)
9. Nyberg, K.: Linear approximation of block ciphers. In: Santis, A.D. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
10. Ohkuma, K.: Weak keys of reduced-round present for linear cryptanalysis. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 249–265. Springer, Heidelberg (2009)
11. Cheng, H., Heys, H.M., Wang, C.: Puffin: A novel compact block cipher targeted to embedded digital systems. In: Fanucci, L. (ed.) DSD, pp. 383–390. IEEE, Los Alamitos (2008)
12. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: Present: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
13. Leander, G., Poschmann, A.: On the classification of 4 bit s-boxes. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 159–176. Springer, Heidelberg (2007)
14. Kündgen, A., Leander, G., Thomassen, C.: Switchings, extensions, and reductions in central digraphs (2010) (preprint)
15. Canteaut, A., Carlet, C., Charpin, P., Fontaine, C.: On cryptographic properties of the cosets of r(1, m). IEEE Transactions on Information Theory 47(4), 1494–1513 (2001)
16. Carlet, C.: Boolean Functions for Cryptography and Error Correcting Codes (to appear)