More on linear hulls of PRESENT-like ciphers and a cryptanalysis of full-round EPCBC–96

Stanislav Bulygin[1,2]

[1] Technische Universität Darmstadt, Department of Computer Science
Hochschulstraße 10, 64289 Darmstadt, Germany
Stanislav.Bulygin@cased.de

[2] Center for Advanced Security Research Darmstadt - CASED
Mornewegstraße 32, 64293 Darmstadt, Germany

Abstract. In this paper we investigate the linear hull effect in the light-weight block cipher EPCBC. We give an efficient method of computing linear hulls with high capacity. We then apply found hulls to derive attacks on the *full* 32 rounds of EPCBC–96 and 20 rounds of EPCBC–48. Using the developed methods we revise the work of J.Y. Cho from 2010 and obtain an attack based on multidimensional linear approximations on 26 rounds of PRESENT–128. The results show that designers of block ciphers should take seriously the threat coming from the linear hull attacks and not just limit themselves to proving bounds based solely on linear characteristics.

Keywords: PRESENT, EPCBC, linear cryptanalysis, linear hull, multidimensional linear cryptanalysis

1 Introduction

Light-weight cryptography has been an extensive field of study recently. The need for cryptographic solutions for small devices, such as RFID tags, smart cards, and sensors, is eminent. In particular, the need for light-weight block ciphers has been covered by a range of new proposals. The block cipher PRESENT [6] is the most remarkable representative.[1] There is a series of ciphers that are inspired by the design of PRESENT, e.g. [5,8,13,21]. For example PRINTcipher [13] is designed specifically for applications in integrated circuit printing. Then, EPCBC [21] is designed for encrypting Electronic Product Code (EPC) information.

Naturally, once cipher proposals appear, there is a need for their security evaluation. The field of cryptanalysis deals with this. Differential [4] and linear [16] cryptanalysis are the classical tools for analyzing block ciphers. In linear cryptanalysis an attacker tries to find biased linear approximations for non-linear components of a cipher (e.g. an S-Box) and then use them to find biased linear approximation for the entire cipher. One is then able to use these biased approximations to recover certain subkey bits. Afterwards, the remaining key bits are recovered by brute force. Linear cryptanalysis is in fact one of the oldest method of cryptanalysis that caught mass attention since the appearance of the papers of Matsui [16,17] where he proposed a theoretical break of the DES cipher. Since the appearance of these classical papers, the method of linear cryptanalysis has been applied to a wide range of other ciphers and the method itself has been extended and improved, e.g. [3,7,10]. In the context of light-weight cryptanalysis, quite a few results are known for PRESENT. In particular, the so-called linear hull effect is observed to a large extend for this cipher. Several

[1] PRESENT is a part of ISO/IEC 29192-2:2011 standard.

papers study the effect [1,15,18,19]. The bottom line here is that security estimates for linear cryptanalysis are usually given in terms of linear characteristics, whereas a linear hull is a sort of combination of all linear characteristics with a certain input and output masks. It has been shown previously that using linear hulls, at least for the case of PRESENT, gives much more power to an attacker than is predicted by a simple linear characteristic analysis usually undertaken by designers. Using this effect and also advanced methods, such as multidimensional linear cryptanalysis it is possible to break up to 25 rounds or PRESENT–80 [9].

In this paper we investigate the linear hull effect for a PRESENT-like cipher EPCBC [21]. We provide an efficient method of computing capacities of linear hulls of EPCBC. We then apply these to attack the full 32 rounds of EPCBC–96. For the smaller cipher EPCBC–48 by using similar methods we can break 16 out of 32 rounds. By making some additional assumptions we can break 20 rounds, however. In the analysis we use recent remarks on estimating complexity of a linear attack made in [15]. We also observe certain inaccuracies that are usually made when analyzing the time complexity of the attack. Afterwards, we move to analyzing PRESENT cipher itself. We revise the results of Cho [9] and show that his attack on 26 rounds of PRESENT–80 does not really work. By employing our methods and the ones of [9] we present an attack on 26 rounds of PRESENT–128. An attack on 26 rounds was also presented in [18], but it has a rather low success probability. Therefore, to our knowledge this is the first attack on 26 rounds of PRESENT–128 with high success rate.

The outline of the paper is as follows. In Section 2 we briefly recall the specifications of the ciphers PRESENT and EPCBC. Section 3 provides a short summary of the linear cryptanalysis. Then in Section 4 we address the problem of efficient finding of certain classes of linear hulls and give the results for PRESENT and EPCBC. In Section 5 we discuss the methods of estimating data and time complexity of the attack. The applications of the method are in Section 6 and 7, where we attack EPCBC and PRESENT resp. We conclude in Section 8. Appendix A contains the correlation table for the PRESENT S-Box.

2 PRESENT and EPCBC block ciphers

2.1 PRESENT

The block cipher PRESENT [6] was proposed at CHES 2007 and is a light-weight block cipher, which has been now internationally standardized [12]. It is a substitution-permutation network encrypting 64-bit blocks with either 80- or 128-bit keys. Both versions of PRESENT have 31 rounds. Schematically, the process of PRESENT encryption and the key schedule can be described as in Figure 1. The S-Box layer consists of consecutive application of 16 identical 4-bit S-Boxes S defined by Table 1.

Table 1. PRESENT S-Box.

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

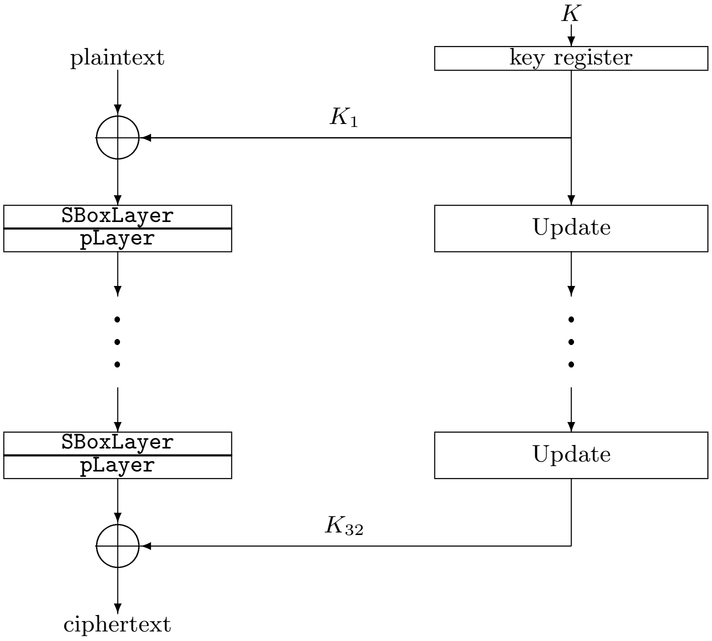

Fig. 1. Schematic description of PRESENT.

The linear diffusion layer `pLayer` is implemented by a bit permutation defined by the map P sending a bit at position i to the bit at position $P(i)$:

$$P(i) = \begin{cases} 16i \bmod n-1, & \text{for } 0 \le i \le b-2, \\ b-1, & \text{for } i = b-1, \end{cases} \tag{1}$$

where $b = 64$ is the block length of PRESENT.

The key schedule of PRESENT–80 is organized as follows. The master key $K = k_{79}k_{78}\dots k_1k_0$ is loaded into an 80-bit register. At round i the register is updated as follows:

1. $[k_{79}k_{78}\dots k_1k_0] \leftarrow [k_{18}k_{17}\dots k_1k_0k_{79}\dots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] \leftarrow S([k_{79}k_{78}k_{77}k_{76}])$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] \leftarrow [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \mathit{round\_counter}$

The leftmost 64 bits of the result of this update are loaded to the i-th round key $K_i = [k_{79}k_{78}\dots k_{17}k_{16}]$. The key schedule for the version with 128 bit keys is organized in a similar fashion.

2.2 EPCBC

EPCBC [21] was proposed at CANS 2011 and is designed specifically for encrypting Electronic Product Codes (EPCs). EPCBC has key length $k = 96$ and has two versions that differ in the block length, either 48 or 98 bits. We denote them by EPCBC-b with $b = 48, 96$. Both versions have 32 rounds. The structure of EPCBC is largely inspired by PRESENT and follows the same structure, see Figure 1. Namely, EPCBC is also an SP-network, employs the same S-Box defined in Table 1 and uses the same bit permutation P defined in (1), of course used with the corresponding b. However, the key schedule of EPCBC is different from the one of PRESENT and uses more non-linearity. The key schedule for $b = 96$ exactly mimics the encryption function. For $b = 48$ the key schedule is using a sort of Feistel-like structure, see Algorithm 1.

Algorithm 1 EPCBC–48 key schedule

```
(LKeystate, RKeystate) = 96-bit key
Subkey[0] ← LKeystate
for $i = 0 \to 7$ do
    temp ← LKeystate ⊕ RKeystate
    for $j = 0 \to 3$ do
        RKeystate ← sBoxLayer(RKeystate)
        RKeystate ← pLayer(RKeystate)
        RKeystate ← RKeystate ⊕ $(4i + j)$
        Subkey$[4i + j + 1]$ ← RKeystate
    end for
    LKeystate ← RKeystate
    RKeystate ← temp
end for
```

3 Linear cryptanalysis

In this section we briefly outline the main notions and ideas of (multi-dimensional) linear cryptanalysis. For a more thorough description, see e.g. [14]. The main idea of linear cryptanalysis is to find biased linear relations between inputs/outputs of a cipher. These, in turn, make it possible to recover certain key bits of a cipher.

Definition 1. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a Boolean function. For non-zero vectors $\alpha, \beta \in \mathbb{F}_2^n$ consider a linear form*

$$l_{\alpha,\beta,F}(x) = \langle \alpha, x \rangle + \langle \beta, F(x) \rangle,$$

defined for $x \in \mathbb{F}_2^n$. A bias *of this linear form is a number $-1/2 \leq \epsilon(\alpha \to \beta) \leq 1/2$, such that*

$$Pr(l_{\alpha,\beta,F}(x) = 0; x \leftarrow \mathbb{F}_2^n) = 1/2 + \epsilon(\alpha \to \beta).$$

Correlation *of $l_{\alpha,\beta,F}(x) = 0$ is simply $c(\alpha \to \beta) := 2\epsilon(\alpha \to \beta)$.*
The vectors α and β are called input *and* output *masks of F resp.*

In our setting F will be a round function of a cipher. More specifically, we will deduce a biased relation for F from those of S-Boxes that constitute the substitution layer of F. A table of correlations of the PRESENT S-Box that is also used in EPCBC is given in Appendix A. If we have an iterative block cipher, the next notion comes in hand.

Definition 2. *Let $F := F_{r-1} \circ \cdots \circ F_0,\ F_i : \mathbb{F}_2^n \to \mathbb{F}_2^n, 0 \leq i \leq r-1$. And let $\{(\alpha_i, \alpha_{i+1})\}_{i=0,\ldots,r-1}$ be a sequence of masks to $F_i, i = 0, \ldots, r-1$. This sequence is called a linear trail of F with the input mask $\alpha_0 =: \alpha$ and output mask $\alpha_r =: \beta$.*
All linear trails with the same input mask α and output mask β constitute a linear hull *with these masks.*

Correlation of a linear trail is usually computed from correlations of individual approximations for each F_i by applying the so-called Matsui's Piling-Up lemma. Under the assumption that individual approximations are independent, one has

$$c(\alpha \to \alpha_1 \to \cdots \to \alpha_{r-1} \to \beta) = \prod_{i=0}^{r-1} c(\alpha_i \to \alpha_{i+1}).$$

Capacity of the trail is simply $C(\alpha \to \alpha_1 \to \cdots \to \alpha_{r-1} \to \beta) := (c(\alpha \to \alpha_1 \to \cdots \to \alpha_{r-1} \to \beta))^2$. One has to be a bit more careful, though, in computing correlations of linear

hulls. Under the assumption that round keys of an alternating block cipher are independent, it is known (cf. e.g. [9]) that capacity of a linear full (averaged over all keys) is the sum of capacities of its composing trails.
Now a basic idea of a linear attack is to 1.) find a linear trail/hull with large capacity; 2.) collect a certain number of plaintext/ciphertext pairs; 3.) by guessing at some subkey bits partially decrypt the ciphertexts and observe if pairs plaintext/decrypted ciphertexts satisfy a linear relation predicted by the trail/hull; 4.) keep counters for the "hits" in 3. and identify correct guesses for the subkey bits; 5.) brute force the remaining (sub-)key bits. We will address computing data/time/memory complexity of this attack in Section 5. Depending on how many rounds are taking part in partial encryptions/decryptions, one may talk about 1R-, 2R-, etc. attacks.

Remark 1. Throughout this paper we will use the fact that (average) capacity of a hull is a sum of capacities of its constituents. This fact holds if certain assumptions on independence hold: independence of round keys, trails, see also Assumption 1 in [15]. Strictly speaking, computations based on this assumption need practical validation for reduced-round versions of a cipher. Still, these assumptions are known to hold for PRESENT and other similar ciphers, e.g. PUFFIN [15]. Therefore, it is plausible to assume its validity also for the EPCBC block cipher.

In a series of works [9,10,11] the notion of a linear hull has been generalized to using several input/output masks combined to obtain more powerful hulls. In essence, one considers masks that come from a linear subspace of $\mathbb{F}_2^n$ of dimension m. In [10] the *advantage* of a multi-dimensional attack is obtained. The advantage being the number of subkey bits targeted by the attack. In Section 7 we will use the following formula of the advantage using χ^2-statistics:

$$adv = \frac{NC - 4\Phi^{-1}(2P_s - 1)^2}{4M}, \tag{2}$$

where N is the amount of data available, C the capacity of the approximation, P_s is the success probability, $M = 2^m - 1$ is the number of participating masks, and finally $\Phi(x)$ is the probability density function of the standard normal distribution.

4 Finding good linear hulls efficiently

At the heart of every linear attack is the question of finding linear trails with high correlation. Moreover, we are interested in finding such trails with the same input/output masks. Such trails then constitute a linear hull. Under certain assumptions, correlation of a linear hull is composed of correlations of individual trails, see Section 3. As is quite known [9,15,18,19] and will be shown again in this paper, it is sometimes the case that it is possible to find linear hulls with much larger correlation than any of its individual trails. Since providing bounds on correlations of individual trails is usually quite manageable, designers tend to rely on estimating this figure when arguing about security against linear attacks. The question of bounding correlation of linear hulls is much more involved, since it has to do with finding a number of linear trails that constitute a hull. This number seems to be in general hard to compute or even bound.

Usually for finding good linear trails and hulls variations of Matsui's branch-and-bound algorithm are used. It is even possible to formulate and solve the problem of finding good trails as a Mixed Integer Linear Program (MILP). So the problem of finding trails/hulls can be solved either by a dedicated solver that is based on the branch-and-bound algorithm specific to the cipher in question, or it can be solved via using MILP and an available general purpose solver, e.g. IBM ILOG CPLEX. The main difficulty with this approach is the fact that both optimization methods in general have exponential complexity. Thus applying them to a large number of rounds may be challenging.

It is known that the S-Box used in PRESENT and also in EPCBC allows many 1-to-1 masks for linear trails passing through the S-Box; a 1-mask being a mask which binary representation has Hamming weight 1. Namely, from the correlation table, cf. Table 4, it follows that $|c(2^i \to 2^j)| = 2^{-2}$ for $1 \leq i, j \leq 3$ except for the case $i = 3, j = 2$.

So we see that 10 out of possible 16 1-to-1 masks hold for the PRESENT S-Box. In this case it is quite reasonable to look for linear trails among those that have only one active S-Box per round. This question has been studied extensively in [9,15,18,19]. Also a question of computing correlations of linear hulls is addressed in [5]. In [9] a formula for computing correlation of specific trails is given. We propose a general algorithm for PRESENT-like ciphers that solves this problem of finding all linear trails with one active S-Box per round. In particular PRESENT and EPCBC fall into this category, as well as the block cipher PUFFIN, albeit the latter being broken [15]. In some sense we generalize the work of [9] applied to PRESENT

In order to construct a trail that has only one active S-Box per round we start with an S-Box in the first round and some input mask $mask_{in}$. We then look for 1-masks that can be obtained from $mask_{in}$ with non-zero correlation. Each such 1-mask $mask_{out}$ goes to a 1-mask $mask'_{in}$ of some S-Box at round two after an application of the permutation P. Now if $mask'_{in}$ is of the form $2^i, 1 \leq i \leq 3$, then we obtain several possibilities to continue the trail by using one of the output 1-masks. The process goes on like this until we hit the penultimate round. There we go with an output 1-mask in the round $R - 1$ to an input 1-mask at round R. This mask in turn can end in an output mask $mask_{out}$. By applying a quite simple counting argument we may compute all trails that start and end at certain S-Boxes and have certain masks. The algorithm is presented in Algorithm 2. The algorithm is run as $\#Trails(In, Out, c_{in}, c_{out})$. In and Out give positions of active S-Boxes in the first and the last rounds resp. Now, c_{in} and c_{out} are absolute values of correlations $c(mask_{in} \to 2^i)$ and $c(2^j \to mask_{out})$ for some $0 \leq i \leq 3$ and $0 \leq j \leq 3$ that start and end the trail, resp. These masks start and end a trail. The c_{in}, c_{out} may attain two values: 2^{-1} and 2^{-2}. Note that all other parts of a trail are coming from a 1-to-1 masks which all have absolute correlation 2^{-2}. Therefore, due to Matsui's Piling-up lemma, we have that absolute correlation of a trail is $c_{in} \cdot c_{out} \cdot 2^{-2(R-2)}$. The $trail\_count[i]$ is a counter of the number of paths one can get from the root nodes from S, defined before the big for-loop, to a position i after a certain (current) number of rounds. At the end of the current round a counter $trail\_count\_next$ is formed from $trail\_count$ by using propagation rules for 1-to-1 masks through an S-Box (lines 18–23) and summing up the corresponding values in $trail\_count$ (lines 25–27). This latter process is illustrated in Figure 2.

Based on the assumptions from Remark 1 we can compute capacity of the linear hull with inputs/outputs given by In and Out. It is

$$C(In, Out) = \sum_{c_{in}, c_{out} \in \{2^{-1}, 2^{-2}\}} 2^{-4(R-2)} c_{in}^2 {c_{out}}^2 \#Trails(In, Out, c_{in}, c_{out}).$$

Accordigly, the correlation is $c(In, Out) = \sqrt{C(In, Out)}$.
In Table 2 we list the best linear hulls we found for $b = 48, 64, 96$, i.e. PRESENT and EPCBC.

Algorithm 2 $\#Trails$:Finding all trails that have given correlation and one S-Box per round

```
Require:
        - Position of the input active S-Box and its mask: $In = (sbox_{in}, mask_{in})$
        - Position of the output active S-Box and its mask: $Out = (sbox_{out}, mask_{out})$
        - Input correlation starter $c_{in}$
        - Output correlation starter $c_{out}$
        - Number of rounds $R$
Ensure: Number of trails with absolute correlation $c_{in} \cdot c_{out} \cdot 2^{-2R+4}$
Begin
$I_{c_{in}} := \{0 \leq i \leq 3|\ |c(mask_{in} \rightarrow 2^i)| = c_{in}\}$
$O_{c_{out}} := \{0 \leq i \leq 3|\ |c(2^i \rightarrow mask_{out})| = c_{out}\}$
$trail\_count[i] = 0, 0 \leq i \leq b-1$
$S := \{i|\ i = 4 \cdot sbox_{in} + j, j \in I_{c_{in}}\}$
$trail\_count[i] = 1, i \in S$
for $round = 1, \ldots, R-1$ do
    $D := \{P(i)|\ i \in S\}$
    $trail\_count\_next[i] = 0, 0 \leq i \leq b-1$
    $T := \{\}$
    for all $i \in D$ do
        $s = \lfloor i/4 \rfloor$
        if $i\ mod\ 4 = 3$ then
            $s\_out := \{4s+1, 4s+3\}$
        end if
        if $1 \leq (i\ mod\ 4) \leq 2$ then
            $s\_out := \{4s+1, 4s+2, 4s+3\}$
        end if
        $T := T \cup s\_out$
        for all $j \in s\_out$ do
            $trail\_count\_next[j] += \ trail\_count[P^{-1}(i)]$
        end for
    end for
    $S := T$
    $trail\_count := trail\_count\_next$
end for
$E := \{P^{-1}(i)|i \in O_{c_{out}}\}$
$Result := \sum_{j \in E} trail\_count[j]$
return $Result$
End
```

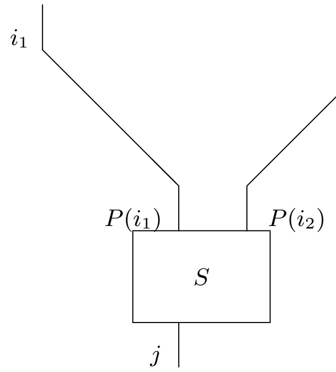

Fig. 2. #paths to $\#j$ = (#paths to i_1) + (#paths to i_2)

These are the best linear hulls that are composed of linear trails with exactly one active S-Box. Note that the numbers in the penultimate column is only an estimate for correlation, since it may happen that there are some linear trails in a hull that have more than one active S-Box and thus are not counted here. We expect contribution of such trails to be negligible, so we expect the estimate to be very close to the actual value. The table lists the overall number of optimal hulls obtained and then positions of active S-Boxes in the first and the last rounds, as well as the input/output masks for one particular hull, as an example. We also list the number of trails with the given positions/masks provided by the starter in the example hull. For example $(+,+)$ means that the values of $c_{in} = c_{out} = 2^{-1}$ are used in Algorithm 2 and $(-,-)$ means that $c_{in} = c_{out} = 2^{-2}$ are used. Note that all optimal hulls follow the same pattern for the numbers in the column "# trails". The last column provides the largest values (or a bound) of the correlation known before. For $b = 48, 96$ we use the original EPCBC paper [21] and for $b = 64$ we use numbers on one-S-Box-hulls from [18,19]. The numbers are provided for the last two rounds, where we could obtain hulls with correlations $> 2^{-b}$. In fact we could easily obtain hulls with smaller correlations for larger number of rounds, but this does not seem to be relevant for the attack.

5 Computing complexity

In this section we estimate complexity of our linear hull attack. The two main questions here are computing data and time complexity. For the former it has been shown, see [15], that some care has to be taken in computing data complexity. For the latter we point out in this paper that computations that are usually done are not quite accurate and need adjustment.

Data complexity. First we deal with the problem of computing data complexity of the attack. It has been shown in [15] that in fact if one tries to compute *average* complexity, one fails, since it is strictly speaking equals to infinity. Therefore, one should instead talk about *median* complexity, or more generally complexity C_p as defined below. We quote here Definition 1 of [15]:

Definition 3. *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 defines complexity C_p to be the complexity such that the probability that for a given key the attack complexity is lower than C_p, is p. In particular, $\tilde{C} = C_{1/2}$.*

Table 2. The best linear hulls for $b = 48, 64, 96$

b	R	overall no.	Input S-Box	Output S-Box	Input mask	Output mask	# trails	starter	$\log_2(corr)$	known $\log_2(corr)$
48	14	4	5	4	[1,0,1,1]	[1,1,0,1]	498	$(+,+)$	-20.81	-24
							439	$(-,+)$		
							249	$(+,-)$		
							219	$(-,-)$		
48	15	4	5	4	[1,0,1,1]	[1,1,0,1]	887	$(+,+)$	-22.39	-26
							781	$(-,+)$		
							443	$(+,-)$		
							389	$(-,-)$		
48	16	4	5	4	[1,0,1,1]	[1,1,0,1]	1579	$(+,+)$	-23.98	-28
							1389	$(-,+)$		
							788	$(+,-)$		
							692	$(-,-)$		
64	23	16	5	5	[1,1,1,0]	[1,0,1,1]	0	$(+,+)$	-30.57	-31.77
							33,116,290	$(-,+)$		
							0	$(+,-)$		
							53,583,010	$(-,-)$		
64	24	16	5	5	[1,1,1,0]	[1,0,1,1]	0	$(+,+)$	-31.88	-33.08
							86,699,300	$(-,+)$		
							0	$(+,-)$		
							140,281,700	$(-,-)$		
96	29	8	5	8	[1,0,1,1]	[1,1,1,1]	2,955,558	$(+,+)$	-44.23	-50
							2	$(-,+)$		
							1,232,503	$(+,-)$		
							0	$(-,-)$		
96	30	8	5	8	[1,0,1,1]	[1,1,1,1]	5,420,564	$(+,+)$	-45.80	-52
							2	$(-,+)$		
							2,260,439	$(+,-)$		
							0	$(-,-)$		

In the following theorem we combine ideas of [15] and e.g. [14,19] to compute the data complexity of the linear attack.

Theorem 1. *Let $\mathcal{L}$ be a linear hull (approximation) used in the attack and C be its capacity. Let $m = m(\mathcal{L})$ be the advantage of the attack (i.e. the number of subkey bits guessed at) and let p be the success probability of the attack. Then the* data complexity *of the attack (i.e. the number of known plaintext/ciphertext pairs) for the portion of p^* of the keys is*

$$N_{p^*} = \frac{\Phi^{-1}(p^{1/2^m})^2}{2C \cdot (erf^{-1}(1-p^*))^2} = \frac{1}{C} \cdot \Big(\frac{erf^{-1}(2p^{1/2^m}-1)}{erf^{-1}(1-p^*)}\Big)^2, \tag{3}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution and $erf(x) = 2\Phi(x\sqrt{2}) - 1$ is the Gauss error function.

Proof. Let X be a random variable corresponding to the correlation of the linear hull $\mathcal{L}$. Further, we assume that X is distributed normally with the mean 0 and variance $\sigma^2 : X \sim \mathcal{N}(0, \sigma^2)$, where $\sigma^2 = \sum \gamma_i^2$ with γ_i being correlations of linear trails that compose the hull $\mathcal{L}$ (cf. Remark 1). As we see, the mean value of X is 0 and since the number of plaintexts needed for the attack (data complexity) is proportional to $1/X^2$, We conclude, as in [15], that this number is ∞. Therefore, we compute for the given $0 < p^* < 1$ the number of plaintexts needed for the attack for the portion of p^* of all keys. This means that our complexity estimate will be valid for $2^k \cdot p^*$ keys, where k is the key length. This corresponds to the C_p-notion from Definition 3.

So we are interested in computing $Prob(c/X^2 \leq N_{p^*})$ for some small constant $c \geq 1$ and appropriately chosen N_{p^*}. Note that this probability is exactly equal to p^*. We have

$$Prob\Big(\frac{c}{X^2} \leq N_{p^*}\Big) = 1 - Prob\Big(X^2 \leq \frac{c}{N_{p^*}}\Big) = 1 - G\Big(\frac{c}{N_{p^*}}\Big),$$

where G is the cumulative distribution function of X^2. Now,

$$G(x) = Prob(X^2 \leq x) = Prob(-\sqrt{x} \leq X \leq \sqrt{x}) = F(\sqrt{x}) - (1 - F(\sqrt{x})) = 2F(\sqrt{x}) - 1,$$

where F is the cumulative distribution function of X. The relation between F and Φ is as follows:

$$F(x) = \Phi\Big(\frac{x}{\sigma}\Big) = \frac{1}{2}\Big(1 + erf\Big(\frac{x}{\sigma\sqrt{2}}\Big)\Big).$$

Combining the above equalities we obtain

$$p^* = 1 - G\Big(\frac{c}{N_{p^*}}\Big) = erf\Big(\frac{1}{\sigma}\sqrt{\frac{c}{2N_{p^*}}}\Big).$$

Therewith we have

$$N_{p^*} = \frac{c}{2\sigma^2 \cdot (erf^{-1}(1-p^*))^2} = \frac{c}{2C \cdot (erf^{-1}(1-p^*))^2}.$$

Now we have to compute c. We have $Prob(x \geq \sqrt{c/N_{p^*}}$ or $x \leq -\sqrt{c/N_{p^*}}) = p^*$. The probability of the linear approximation given by $\mathcal{L}$ for a portion p^* is $1/2 \pm \gamma/2$ with $\gamma = \sqrt{c/N_{p^*}}$ for a correct key and $1/2$ for an incorrect key. The probability for one subkey bit guess is known to be given by [19]:

$$p_S = 1 - \Phi(-\gamma\sqrt{N_{p^*}}) = \Phi(\gamma\sqrt{N_{p^*}}) = \Phi(\sqrt{c}).$$

The success probability to recover all m subkey bits is ($2^m - 1$ incorrect key guesses are to be identified):

$$p = p_S^{2^m-1} = \Phi(\sqrt{c})^{2^m-1} \approx \Phi(\sqrt{c})^{2^m}$$

for large enough m. Therefore $c = (\Phi^{-1}(p^{1/2^m}))^2$ and finally we obtain (3).

Time and memory complexity. For estimating time complexity we take the time of one trial encryption as a measuring unit, so that the brute force approach takes 2^k and on average 2^{k-1} units. Since we aim at recovering m subkey bits, a naive approach would yield time complexity $N \cdot 2^m \cdot \omega$, where N is the number of plaintexts and $0 < \omega < 1$ is the factor to adjust this number to the number of trial encryptions. For example if we do a $2R-$attack on a 32-round cipher, we would set $\omega = 2/32 = 1/16$. There is a nice time/memory trade-off approach described in [9] . The idea for a $2R$-attack (the first and the last round) is as follows. Let p and c be parts of plaintext/ciphertext that are targeted by the subkey guesses. We denote by k_0 and k_R the parts of the first and the last subkeys that are targeted in guessing. Therefore $(p||c) \in \mathbb{F}_2^m$ and $k = (k_0||k_R) \in \mathbb{F}_2^m$. Now one is interested in studying the distribution of $z_k = (P(S(p+k_0))||S^{-1}(P^{-1}(c+k_R)))$ for all N pairs of plaintext/ciphertext. So one first precomputes the frequencies of $(p||c)$ for all N pairs and stores them in the table Q. Constructing such a table costs $N \cdot \omega'$ encryptions and around $2^m \cdot \log(\log N - m)$ bits of memory. Here ω' is another adjusting parameter; it is clear that $\omega' < 1$, but it is a matter of estimation to decide the more or less exact value. Once we have Q and given the "key" $k \in \mathbb{F}_2^m$ we can compute z_k in the time $2^m\omega''$ (ω'' is yet another adjusting parameter). Thus, for all keys that yields the time complexity $2^{2m}\omega''$. Then computing the correlation of an approximation given by z_k takes some constant time t_b, thus $2^m t_b$ in total. So the overall complexity of recovering m subkey bits using a linear approximation is around

$$T \approx N\omega' + 2^{2m}\omega'' + t_b 2^m \approx \max\{N\omega', 2^{2m}\omega''\}. \tag{4}$$

Now comes an interesting issue. After we have recovered the m subkey bits (either from some first or last rounds or both), one has to recover the k-bit *master* key, i.e. the key that is actually used for the encryption and which is used to obtain the round keys. The usual estimation of time complexity undertaken at this point (in papers on both linear and differential cryptanalysis) is that one has to recover the "remaining" $k - m$ key bits and thus the time complexity is 2^{k-m}. For most ciphers this estimation is an underestimation, however, if we talk about $\alpha R-$attacks, where $\alpha > 1$.

In order to see this, let us consider a $2R-$attack on PRESENT where a bits are recovered in the subkey K_1 and a bits in the subkey K_{32}, so that $m = 2a$. Knowledge of these m bits is *not* equivalent to knowledge of m master key bits. What is true, is that we know a master key bits that come from K_1, see Section 2. Can we efficiently compute more master key bits from the a subkey bits of K_{32}? In general, no. This is due to the fact that PRESENT, as well as many other ciphers, has non-linearity built into the key schedule. For example, subkey bits $K_2[76 \dots 79]$ depend non-linearly on $K_1[15 \dots 18]$. So knowledge of just one value of any bit there does not, in general, yield value of any other bit. In principle, one could express bits from K_1 via non-linear polynomials (of high degree) in the bits of K_{32}. However, one should expect such polynomials to be close to random and therefore, deducing values of K_1 by knowing some a values of K_{32} is virtually impossible. We will see another specific example later. One could, of course, try to run the following scenario. Obtain a master key bits from K_1, set a system of algebraic equations representing the key schedule, set some a variables that represent bits

of K_{32} to known values. Now try to solve this system of non-linear equations targeting to recover the remaining $80 - a$ master key bits. This is a form of an algebraic attack. Since we know that algebraic attacks on block ciphers perform quite poorly when the number of rounds is large, we should not expect such an attack to be faster than 2^{80-a}. So, the time complexity of the last search step is 2^{80-a} and not 2^{80-2a} as would normally be stated.
Assume we run a $2R-$attack targeting the first and the last round, and $m = m_1 + m_R$, where m_i is the number of subkey bits recovered in the round $i = 1, R$ in the previous phase. Now assume w.l.o.g. that $m_1 \geq m_R$ we have that the overall time complexity of the linear attack is

$$T_{attack} \approx \max\{N\omega', 2^{2m}\omega'', 2^{k-m_1}\}. \tag{5}$$

We have that $T_{attack} < 2^k$ if $N \lesssim 2^k$ and $2m \lesssim k$.

Remark 2. 1. Interestingly enough, the above "2^{80-m}-statement" is true for the case of DES [17]. The reason lies again in the structure of the key schedule of DES: it is not only linear, but actually subkey bits of the round keys *are* always some master key bits. In his paper [17] Matsui showed that m subkey bits he recovered from rounds 15 and 16 are actually different m master key bits. Therefore, indeed, the final search phase takes here 2^{56-m} encryptions.

2. The overlook often made, fortunately, usually has little impact on the actual complexity of the attack, since the attack is usually dominated by the first two terms in (5). One important exception will be discussed in Section 7.1 where we look at the attack of [9] on 26 rounds on PRESENT–80.
3. It may seem that the discussion above implies that it does not make sense to run a $2R-$attack on the first and the last round, since some subkey bits recovered at this stage are "useless" for the final search stage. This is not true, however. If instead we would run a $2R$-attack on the first two or the last two rounds, the advantage of such an attack would usually be higher, which leads to higher data complexity. This is something we cannot afford when the attack approaches its limits on the data available.

6 Linear cryptanalysis of EPCBC

6.1 A cryptanalysis of the full-round EPCBC–96

We describe a linear hull attack on the full 32 rounds of EPCBC–96. For the attack we use one of the hulls from Table 2 that attain maximum capacity. Namely, we take a linear hull, which input is active in the S-Box no. 5 and output is active in S-Box no. 8. The input mask is $[1, 0, 1, 1]$ and the output mask is $[1, 1, 1, 1]$. As we see from Table 2 this linear hull over 30 rounds has capacity $2^{-45.80 \cdot 2} = 2^{-91.60}$. We now run a $2R$-attack, where we target the first and the last rounds. It turns out that this hull is one of those that attain minimal data complexity for $p^* = 1/8$ among all linear hulls with one active S-Box per round. Indeed, using formula (3) with $m = (3 + 4) \cdot 4 = 28$ we have that $N_{p^*} = 2^{95.65} < 2^{96}$, where the latter is the size of the codebook. In Table 3 we list some other hulls that attain minimal data complexity.

As to the time complexity, we turn to formula (5). From there we see that $T_{attack} \approx N_{p^*}\omega'$, since the term $N_{p^*}\omega'$ dominates the other two. We conservatively estimate an operation to fill an appropriate element in the matrix Q to be equal to two rounds of EPCBC–96 encryption. Therewith $\omega' \approx 2/32 = 2^{-4}$. So we estimate the overall time complexity of the attack to be

Table 3. Linear hulls attaining minimum data complexity for $2R$-attack on 32 rounds with $p^* = 1/8, p = 0.95$

Input S-Box	Input mask	Output S-Box	Output mask
5	[1, 1, 0, 1]	8	[1, 1, 1, 1]
5	[1, 1, 0, 1]	11	[1, 1, 1, 1]
5	[1, 1, 0, 1]	20	[1, 1, 1, 1]
5	[1, 1, 0, 1]	23	[1, 1, 1, 1]
6	[1, 1, 0, 1]	8	[1, 1, 1, 1]
6	[1, 1, 0, 1]	11	[1, 1, 1, 1]
6	[1, 1, 0, 1]	20	[1, 1, 1, 1]
6	[1, 1, 0, 1]	23	[1, 1, 1, 1]

$T_{attack} \approx 2^{92}$ EPCBC–96 encryptions. In any case, since $\omega' < 1$, the overall time complexity is always $< 2^{96}$.

Now, note that we have chosen $p^* = 2^{-3}$ to get our attack to work; for larger p^* the attack on 32 rounds does not work anymore. Still, for a fraction of 1/8 of all 2^{96} keys we expect our attack to have data complexity $< 2^{96}$ plaintext/ciphertext pairs. In [19] the term "weak keys" is used in this context. We argue that the statement is actually stronger. Since there is no way to efficiently describe the class of $2^{96-3} = 2^{93}$ keys that enable the attack, we have nothing better as to assume that one out of eight possible encryption keys is susceptible to the attack. Of course, this probability is rather high, which renders EPCBC–96 insecure against the linear hull attack.

We can break 31 rounds with $p^* = 1/2$ by using the same hull, but over 29 rounds. Using (3) we obtain $N_{p^*} = 2^{94.90}$. So we have an attack for half of all possible keys.

6.2 A cryptanalysis of 20 rounds of EPCBC–48

Now we describe an attack on reduced-round EPCBC–48. Since the codebook of this cipher is only 2^{48} we cannot hope to break the full 32 rounds with our approach. Still, we are able to run attacks on a considerable number of rounds.

First, let us describe an attack on 16 rounds. The idea is the same as for EPCBC–96 above. We use a linear hull over 14 rounds from Table 2. It has input active S-Box no. 5 with the input mask $[1, 0, 1, 1]$ and the active output S-Box no. 4 with the output mask $[1, 1, 0, 1]$. It has capacity $2^{-20.81 \cdot 2} = 2^{-41.62}$. The $2R$-attack on 16 rounds with $p^* = 1/2$ yields data complexity $N_{p^*} = 2^{47.83}$. The time complexity can be estimated, similarly to the 96-bit case, to be $T_{attack} \approx 2^{46}$. So this is the result we should have expected by observing linear hulls of EPCBC–48.

Under some additional assumptions we can do better, however. We can obtain an attack on 20 rounds, using the following observation on the key schedule of EPCBC–48. The keys for the rounds 17–20 are obtained in a predetermined manner from the value $temp = LKeystate \oplus RKeystate$ that is derived from the earlier key schedule operations, see Algorithm 1. Now, assume that we know this 48-bit vector *temp*. The cipher must still provide 48-bit security. Now, since we know *temp*, we know round keys for the rounds 17–20, and, therefore, we are able to decrypt a ciphertext after 20 rounds to obtain an output of round 16. After this we run our $2R-$attack as described above. As has been shown, we can beat the 48-bit security with this attack. This attack may be considered as a strong form of a related key scenario, where we assume known the difference between certain round keys.

7 Linear cryptanalysis of the reduced-round PRESENT

7.1 Revising the work of [9]

In his paper [9] J.Y. Cho used multidimensional linear approximations and χ^2-statistics to attack 25 rounds of PRESENT–80. In particular he used a multidimensional linear approximation composed of linear trails with only one active S-Box per round, but as opposed to the linear hull approach there are several possible S-Boxes that can be active in the input and in the output. For his attack he used a configuration with input active S-Boxes $I = \{5, 9, 13\}$ and output active S-Boxes $O = \{5, 6, 7\}$. So his multidimensional linear approximation contains all trails

$$a_{ij} : \alpha_i \to \beta_j, i \in I, j \in O,$$

where $\alpha_i, \beta_j \in \mathbb{F}_2^4 \setminus \{0\}$ are non-zero input/output masks for the S-Boxes i and j resp. Capacity of this approximation is then computed as in [9]:

$$C = \sqrt{\sum_{i \in I, j \in O} c(a_{ij})^2}.$$

Note that we can use our method from Section 4 to compute capacities of such linear approximations. Capacities of Cho's approximations for different number of rounds are given in Table 1 of [9]. Using our method from Algorithm 2 adjusted appropriately, we can confirm the numbers there. For his $2R$-attack on 25 rounds, he used a 23-round approximation with capacity $2^{-52.77}$. Interestingly enough, Cho's approximation is not the one attaining maximal capacity. The maximal capacity for 23 rounds is $2^{-52.38}$ and is attained e.g. by the approximation A with $I = \{6, 9, 10\}$ and $O = \{7, 13, 15\}$. What makes Cho's approximation special is almost optimal capacity together with low number of active S-Boxes in the "built-upon" rounds: the one before the approximation and the one after to mount an attack on $1+23+1 = 25$ rounds. Indeed, three active S-Boxes in the input/output give rise to only four S-Boxes per built-upon round, whereas the approximation A yields eight S-Boxes and 16 overall. The $4 \cdot 2 = 8$ active S-Boxes in the attack yield an advantage $adv = 8 \cdot 4 = 32$. Now the data complexity of the attack is computed by formula (2) with $p_s = 0.95, adv = 32, M = 9 \cdot (2^8 - 1)$. So the attack on 25 rounds has data complexity $N = 2^{61.87}$. This is the lowest one can get with approximations involving three input/output S-Boxes as has been verified by our experiments.

After presenting an attack on 25 rounds, Cho went on with extending the attack to 26 rounds. He argued that using the entire codebook of 2^{64} plaintexts/ciphertexts, from formula (2) it follows that for the attack on 26 rounds using his approximation over 24 rounds with capacity $2^{-55.38}$ the advantage is equal to 16[2]. This means that the correct "subkey" is ranked within the position of $2^{32-16} = 2^{16}$ out of 2^{32} candidates with probability $p_s = 0.95$. So by running the attack on 26 rounds with 2^{64} plaintexts, one obtains 2^{16} candidates for the correct subkey with high probability. Now, in [9] it is argued that $80 - adv = 80 - 32 = 48$ key bits are left to be recovered. That led to time complexity figure of $2^{64} + 2^{48} \cdot 2^{16} = 2^{65}$. Here is the place where our discussion of Section 5 comes into play. Note that Cho's attack targets 16 bits of K_1 and 16 bits of K_{26}. As we have seen, this means that actually $80 - 16 = 64$ bits of the master key are to be recovered, not 48. This leads then to the time complexity estimate $2^{64} + 2^{64} \cdot 2^{16} > 2^{80}$. So the attack fails. Similarly the attack fails for PRESENT–128.

[2] He computed it to be equal to 8, but this is due to a mistake in (2), which was also pointed out in [10].

7.2 A cryptanalysis of 26 rounds of PRESENT–128

We are able to run a valid attack on 26 rounds, however it works only for PRESENT–128. What we do is we consider linear approximations that have four active S-Boxes in the input and the output. Clearly, therewith we obtain larger capacity, simply because we add up more trails. The problem to be addressed here is growing advantage due to a large number of active S-Boxes. We take a multidimensional approximation over 24 rounds with input active S-Boxes $I = \{5, 6, 9, 10\}$ and output active S-Boxes $O = \{5, 7, 13, 15\}$. This approximation has advantage $adv = 64$ with 32 bits active in rounds 1 and 26. This approximation has capacity $2^{-54.16}$. Still for 26 rounds one would need $2^{64.17}$ data to mount an attack (note that now $M = 16 \cdot (2^8 - 1)$). So we do the same trick Cho tried to use. For PRESENT–128 from (2) we see that by using the entire codebook we can have an advantage of 50 in the attack on 26 rounds. So we run the attack on 26 rounds and obtain a correct subkey bits among possible $2^{adv-50} = 2^{14}$ candidates with probability $p_s = 0.95$. Now combined with remaining $128 - 32 = 96$ bits of the master key to be searched this yields time complexity

$$T_{attack} \approx \max\{2^{64}\omega', 2^{128}\omega'', 2^{96} \cdot 2^{14}\} = 2^{128}\omega''.$$

Recall that $0 < \omega'' < 1$ is the adjusting parameter to scale the complexity down to trial encryptions. We estimate ω'' to be coming from 2-round encryptions, so that the overall time complexity is about $2^{124.5}$ PRESENT–128 encryptions.

8 Conclusions and future work

In this paper we considered linear cryptanalysis of the PRESENT-like block cipher EPCBC and PRESENT itself. We show that, although known for some time now, the linear hull effect still plays an important role in cryptanalysis even of recent proposals. As a result we could break the full EPCBC–96 for a fraction of 1/8 of all keys. Therefore, designers should be more careful in studying linear properties of a cipher, not just limit themselves to studying linear trails and bounding correlation of the best trail.

As a future work, we see investigating what other ciphers could be susceptible to these kinds of attacks. For example, the block cipher LBlock [22] and recently proposed TWINE [20] seem appropriate targets: both show certain structural similarities and both do not go beyond standard evaluations of security regarding linear cryptanalysis. Moreover, the S-Box of TWINE seems to be only slightly stronger w.r.t linear attacks than the one of PRESENT. Therefore, one should investigate the linear hull effect properly here.

Acknowledgements

The author is funded by the German Science Foundation (DFG) grant BU 630/22-1 and CASED. The author is also thankful to anonymous referees for their comments and to Johannes Buchmann for his support.

References

1. M.A. Abdelraheem, M. Agren, P. Beelen, G. Leander: “On the Distribution of Linear Biases: Three Instructive Examples”. In R. Safavi-Naini, R. Canetti (Eds.) CRYPTO 2012, LNCS 7417, pp. 50–67, 2012.

2. M, Agren, T. Johansson: “Linear Cryptanalysis of PRINTcipher - Trails and Samples Everywhere”. In D.J. Bernstein, S. Chatterjee (Eds.) INDOCRYPT 2011, LNCS 7107, pp. 114–133.
3. E. Biham, O. Dunkelman, N. Keller: “Linear cryptanalysis of reduced round Serpent”. In M. Matsui (Ed.) FSE 2001, LNCS 2355, pp. 16–27, 2001.
4. E. Biham, A. Shamir: “Differential Cryptanalysis of the Data Encryption Standard”. Springer, 1993.
5. A. Bogdanov, M. Knezević, G. Leander, D. Toz, K. Varici, I. Verbauwhede: “SPONGENT: The design space of lightweight cryptographic hashing”. IEEE Transactions on Computers, vol. 99, 2012.
6. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, C. Vikkelsoe: “PRESENT – An Ultra-Lightweight Block Cipher”. In P. Pailier, I. Verbauwhede (Eds.) CHES 2007, LNCS 4727, pp.450–466.
7. J. Borst, B. Preneel, J. Vandewalle: “Linear cryptanalysis of RC5 and RC6”. In L.R. Knudsen (Ed.) FSE 1999, LNCS 1636, pp. 16–30, 1999.
8. H. Cheng, H.M. Heys, C. Wang: “Puffin: A novel compact block cipher targeted to embedded digital systems”. In L. Fanucci (Ed.) DSD. pp. 383–390, 2008.
9. J.Y. Cho: “Linear Cryptanalysis of Reduced-Round PRESENT”. In J. Pieprzyk (Ed.) CT-RSA 2010, LNCS 5985, pp. 302–317, 2010.
10. M. Hermelin, J.Y. Cho, K. Nyberg: “Multidimensional extension of Matsui’s algorithm 2”. In O. Dunkelman (Ed.) FSE 2009, LNCS 5665, pp. 209–227, 2009.
11. M. Hermelin, K. Nyberg: “Linear Cryptanalysis Using Multiple Linear Approximations”, IACR Cryptology ePrint Archive, available at http://eprint.iacr.org/2011/093, 2011.
12. ISO/IEC 29192-2:2012. “Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers”, available at http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56552, 2011.
13. L. Knudsen, G. Leander, A. Poschmann, M.J.B. Robshaw: “PRINTCipher: A Block Cipher for IC-Printing”. In S. Mangard and F.-X. Standaert (Eds.) CHES 2010, LNCS 6225, pp.16–32, 2010.
14. L.R. Knudsen, M.J.B. Robshaw: “The Block Cipher Companion”. Springer, 2011.
15. G. Leander: “On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN”. In K.G. Paterson (Ed.) Eurocrypt 2011, LNCS 6632, pp. 303–322, 2011.
16. M. Matsui: “Linear cryptanalysis method for DES cipher”. In T. Helleseth, (Ed.) Eurocrypt 1993, LNCS 765, pp. 386–397, 1994.
17. M. Matsui: “The first experimental cryptanalysis of the Data Encryption Standard”. In Y.G. Desmedt, CRYPTO 1994, LNCS 839, pp. 1–11, 1994.
18. J. Nakahara Jr, P. Sepehrdad, B. Zhang, M. Wang: “Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT”. In J.A. Garay, A. Otsuka (Eds.) CANS 2009, LNCS 5888, pp. 58–75, 2009.
19. K. Ohkuma: “Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis”. In M.J. Jacobson Jr., V. Rijmen, R. Safavi-Naini (Eds.) SAC 2009, LNCS 5867, pp. 249–265, 2009.
20. T. Suzaki, K. Minematsu, S. Morioka, E. Kobayashi: “TWINE: A Lightweight Block Cipher for Multiple Platforms”. SAC 2012.
21. H. Yap, K. Khoo, A. Poschmann, M. Henricksen: “EPCBC – A Block Cipher Suitable for Electronic Product Code Encryption”. In D. Lin, G. Tsudik, X. Wang (Eds.) CANS 2011, LNCS 7092, pp. 76–97.
22. W. Wu, L. Zhang: “LBlock: A Lightweight Block Cipher”. In J. Lopez, G. Tsudik (Eds.) ACNS 2011, LNCS 6715, pp. 327–344.

A Correlation table of the PRESENT S-Box

The numbers in Table 4 are computed according to Definition 1, where F is the PRESENT S-Box with $n = 4$.

Table 4. Correlation table of the PRESENT S-Box.

α/β	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	0	0	0	0	-2^{-1}	0	-2^{-1}	0	0	0	0	0	-2^{-1}	0	2^{-1}
2	0	2^{-2}	2^{-2}	-2^{-2}	-2^{-2}	0	0	2^{-2}	-2^{-2}	0	2^{-1}	0	2^{-1}	-2^{-2}	2^{-2}
3	0	2^{-2}	2^{-2}	2^{-2}	-2^{-2}	-2^{-1}	0	-2^{-2}	2^{-2}	-2^{-1}	0	0	0	-2^{-2}	-2^{-2}
4	0	-2^{-2}	2^{-2}	-2^{-2}	-2^{-2}	0	2^{-1}	-2^{-2}	-2^{-2}	0	-2^{-1}	0	0	-2^{-2}	2^{-2}
5	0	-2^{-2}	2^{-2}	-2^{-2}	2^{-2}	0	0	2^{-2}	2^{-2}	-2^{-1}	0	2^{-1}	0	2^{-2}	2^{-2}
6	0	0	-2^{-1}	0	0	-2^{-1}	0	0	-2^{-1}	0	0	2^{-1}	0	0	0
7	0	0	2^{-1}	2^{-1}	0	0	0	0	-2^{-1}	0	0	0	0	2^{-1}	0
8	0	2^{-2}	-2^{-2}	0	0	-2^{-2}	2^{-2}	-2^{-2}	2^{-2}	0	0	-2^{-2}	2^{-2}	2^{-1}	2^{-1}
9	2^{-1}	-2^{-2}	-2^{-2}	0	0	2^{-2}	-2^{-2}	-2^{-2}	-2^{-2}	-2^{-1}	0	-2^{-2}	2^{-2}	0	0
a	0	2^{-1}	0	2^{-2}	2^{-2}	2^{-2}	-2^{-2}	0	0	0	-2^{-1}	2^{-2}	2^{-2}	-2^{-2}	2^{-2}
b	-2^{-1}	0	0	-2^{-2}	-2^{-2}	2^{-2}	-2^{-2}	-2^{-1}	0	0	0	2^{-2}	2^{-2}	2^{-2}	-2^{-2}
c	0	0	0	-2^{-2}	-2^{-2}	-2^{-2}	-2^{-2}	2^{-1}	0	0	-2^{-1}	-2^{-2}	2^{-2}	2^{-2}	-2^{-2}
d	2^{-1}	2^{-1}	0	-2^{-2}	-2^{-2}	2^{-2}	2^{-2}	0	0	0	0	2^{-2}	-2^{-2}	2^{-2}	-2^{-2}
e	0	2^{-2}	2^{-2}	-2^{-1}	2^{-1}	-2^{-2}	-2^{-2}	-2^{-2}	-2^{-2}	0	0	-2^{-2}	-2^{-2}	0	0
f	2^{-1}	-2^{-2}	2^{-2}	0	0	-2^{-2}	-2^{-2}	-2^{-2}	2^{-2}	2^{-1}	0	2^{-2}	2^{-2}	0	0