Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT

Jorge Nakahara Jr.[1], Pouyan Sepehrdad[1], Bingsheng Zhang[2,⋆], and Meiqin Wang[3,⋆⋆]

[1] EPFL, Lausanne, Switzerland
[2] Cybernetica AS, Estonia and University of Tartu, Estonia
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
{jorge.nakahara,pouyan.sepehrdad}@epfl.ch, b.zhang2009@gmail.com, mqwang@sdu.edu.cn

Abstract. The contributions of this paper include the first linear hull and a revisit of the algebraic cryptanalysis of reduced-round variants of the block cipher PRESENT, under known-plaintext and ciphertext-only settings. We introduce a pure algebraic cryptanalysis of 5-round PRESENT and in one of our attacks we recover half of the bits of the key in less than three minutes using an ordinary desktop PC. The PRESENT block cipher is a design by Bogdanov *et al.*, announced in CHES 2007 and aimed at RFID tags and sensor networks. For our linear attacks, we can attack 25-round PRESENT with the whole code book, $2^{96.68}$ 25-round PRESENT encryptions, 2^{40} blocks of memory and 0.61 success rate. Further we can extend the linear attack to 26-round with small success rate. As a further contribution of this paper we computed linear hulls in practice for the original PRESENT cipher, which corroborated and even improved on the predicted bias (and the corresponding attack complexities) of conventional linear relations based on a single linear trail.

Keywords: block ciphers, RFID, linear hulls, algebraic analysis, systems of sparse polynomial equations of low degree.

1 Introduction

This paper describes linear (hull) and algebraic cryptanalysis of reduced-round versions of the PRESENT block cipher, a design by Bogdanov *et al.* aimed at restricted environments such as RFID tags [3] and sensor networks. For the linear

⋆ This author is supported by Estonian Science Foundation, grant #8058, the European Regional Development Fund through the Estonian Center of Excellence in Computer Science (EXCS), and the 6th Framework Programme project AEOLUS (FP6-IST-15964).

⋆⋆ This author is supported by 973 Program of China (Grant No.2007CB807902) and National Outstanding Young Scientist fund of China (Grant No. 60525201).

J.A. Garay, A. Miyaji, and A. Otsuka (Eds.): CANS 2009, LNCS 5888, pp. 58–75, 2009.

case, our analysis include linear hulls of reduced-round variants of PRESENT, which unveils the influence of the linear transformation in the clustering effect of linear trails. The computation of linear hulls also served to determine more accurately the overall bias of linear relations, and consequently, more precise complexity figures of the linear attacks.

Previous known analyses on (reduced-round) PRESENT, including the results in this paper, are summarized in Table 4 along with attack complexities.

Our efficient attacks reach 25-round PRESENT under a known-plaintext setting and 26-round with small success rate, and 15 rounds under a ciphertext-only setting. The algebraic attacks, on the other hand, can recover keys from up to 5-round PRESENT in a few minutes, with only five known plaintext-ciphertext pairs.

This paper is organized as follows: Sect. 2 briefly details the PRESENT block cipher; Sect. 3 presents our algebraic analysis on PRESENT; Sect. 4 describes our linear cryptanalysis of reduced-round PRESENT; Sect. 5 describes our linear hull analysis of PRESENT; Sect. 6 concludes the paper.

2 The PRESENT Block Cipher

PRESENT is an SPN-based block cipher aimed at constrained environments, such as RFID tags and sensor networks. It was designed to be particularly compact and competitive in hardware. PRESENT operates on 64-bit text blocks, iterates 31 rounds and uses keys of either 80 or 128 bits. This cipher was designed by Bogdanov *et al.* and announced at CHES 2007 [3]. Each (full) round of PRESENT contains three layers in the following order: a bitwise exclusive-or layer with the round subkey; an S-box layer, in which a fixed 4×4-bit S-box (Table 5) is applied sixteen times in parallel to the intermediate cipher state; a linear transformation, called pLayer, consisting of a fixed bit permutation. Only the xor layer with round subkeys is an involution. Thus, the decryption operation requires the inverse of the S-box (Table 5) and of the pLayer. After the 31st round there is an output transformation consisting of an exclusive-or with the last round subkey. One full round of PRESENT is depicted in Fig. 1. Our

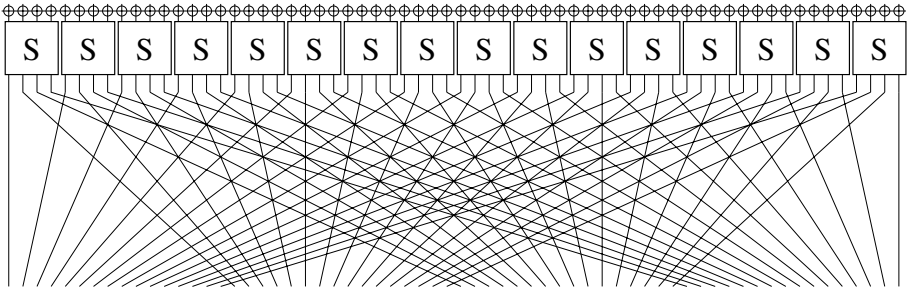

Fig. 1. One full round of PRESENT

attacks are independent of the key schedule algorithm. Further details about the key schedule, for each key size, can be found in [3].

3 Algebraic Analysis

Algebraic cryptanalysis is attributed to C.E. Shannon, who mentioned in [27] that breaking a good cipher should require "as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type". Contrary to linear and differential attacks, that require a large number of chosen or known plaintexts (which makes it roughly impractical in reality), algebraic cryptanalysis requires a comparatively small number of text pairs. Any algebraic attack consists of two distinct stages: the adversary writes the cipher as a system of polynomial equations of low degree over $\mathbb{GF}(2)$ or $\mathbb{GF}(2^k)$ [7,23]. Then, it solves the corresponding system which turns out to be overdefined and sparse. The methods already proposed for solving polynomial system of equations are Gröbner basis including Buchberger algorithm [4], F4 [15], F5 [16] and algorithms like ElimLin [9], XL [6] and its family [7], and Raddum-Semaev algorithm [26]. Converting these equations to Boolean expressions in Conjunctive Normal Form (CNF) [9] and deploying various SAT-solver programs is another strategy [14]. Algebraic attacks since the controversial paper of [7] has gotten considerable attention, has been applied to several stream ciphers (see [8]) and is able to break some of them but it has not been successful in breaking real life block ciphers, except Keeloq [11,18].

In this paper we deploy ElimLin algorithm proposed by Courtois against DES [9] and F4 algorithm by Faugére [15] and we break up to 5-round PRESENT for both key sizes. Then we compare our results using these two approaches. Courtois and Debraize in [10] have already proposed a Guess-then-Algebraic attack on 5-round PRESENT only for the 80-bit key version. In fact, our main focus in this paper is a comparison between the efficiency of ElimLin algorithm which uses simple linear algebra and the recent so called efficient implementation of F4 algorithm under PolyBori framework. Although there exist other types of attacks for larger number of rounds, we believe this result is interesting, because we can recover many key bits with a relatively few known plaintext-ciphertext pairs. Moreover, the designers of PRESENT in [3] have mentioned that they were unsuccessful to obtain any satisfactory result in reasonable time using algebraic cryptanalysis (F4 algorithm under MAGMA [21]) to break two rounds of a smaller version of the cipher having only seven S-boxes per round compared to the real PRESENT cipher having sixteen S-boxes per round.

3.1 ElimLin Algorithm and Attack Description

The *ElimLin* algorithm stands for *Eliminate Linear* and is a technique for solving systems of multivariate polynomial equations of degree mostly 2, 3 or 4 over a finite field, specifically $\mathbb{GF}(2)$. Originally, it was proposed in [9] to attack DES and was reported to break 5-round DES.

ElimLin is composed of two distinct stages, namely: *Gaussian Elimination* and *Substitution*. All the linear equations in the linear span of initial equations are found. Subsequently, one of the variables is nominated in each linear equation and is substituted in the whole system. This process is repeated up to the time when no linear equation is obtained in the linear span of the system.

3.2 F4 Algorithm under PolyBori Framework

F4 is currently the most efficient algorithm for computing the Gröbner basis of an ideal. The most efficient implementation of F4 is available under PolyBori framework [2] running alone or under SAGE algebra system. PolyBori is a C++ library designed fundamentally to compute Gröbner basis applied to Boolean Polynomials. The ring of Boolean Polynomials is a quotient ring over $\mathbb{GF}(2)$, where the field equation for each variable is $x^2 = x$. A Python interface is used, surrounding the C++ core, announced by the designers to be used from the beginning to facilitate "parsing of complex polynomial systems" and "sophisticated and easy extendable strategies for Gröbner base computation" [2]. It uses zero-suppressed binary decision diagrams (ZDDs) [17] as a high level data structure for storing Boolean Polynomials which results in the monomials to be stored more efficiently with respect to the space they occupy in memory and making the computational speed faster compared with other computer algebra systems. We used polybori-0.4 in our attacks.

3.3 Algebraic Representation of PRESENT

It is a straightforward procedure to demonstrate that every 4×4-bit S-box has at least 21 quadratic equations. The larger the number of equations, the weaker the S-box. In fact, the S-box of PRESENT has exactly 21 equations. Writing the whole 80-bit key variant of PRESENT as a system of quadratic equations for 5 rounds, we obtained 740 variables and 2169 equations. In our attack, we fix some of the key bits and we recover the remaining unknown ones. In fact, we introduce an attack on both PRESENT with key sizes of 80 and 128 bits. Notice that for both key sizes one pair is not enough to recover the key uniquely and we need at least two pairs.

The summary of our results is in Table 1. All the timings were obtained under a 2Ghz CPU with 1Gb of RAM and we used an efficient implementation of ElimLin available in [12]. As it is depicted in Table 1, the timing results of ElimLin and PolyBori are comparable except the time in which PolyBori crashed[1] due to probably running out of memory. As our experiments revealed, in all cases ElimLin used much less memory compared to F4 under PolyBori which turns out to be due to the Gröbner basis approach of increasing the degree of polynomials in the intermediate stages.

[1] In Appendix, we give the intermediate results of ElimLin for one of the cases in which PolyBori crashes.

Table 1. Algebraic attack complexities on reduced-round PRESENT

# rounds	#key bits	#key bits fixed	full key (hours)	# plaintexts	notes
5	80	40	0.04	5 KP	ElimLin
5	80	40	0.07	5 KP	PolyBori
5	80	37	0.61	10 KP	ElimLin
5	80	37	0.52	10 KP	PolyBori
5	80	36	3.53	16 KP	ElimLin
5	80	36	Crashed!	16 KP	PolyBori
5	80	35	4.47	16 KP	ElimLin
5	80	35	Crashed!	16 KP	PolyBori
5	128	88	0.05	5 KP	ElimLin
5	128	88	0.07	5 KP	PolyBori

Since the time complexity of the experiment depends on the system instance, Table 1 represents the average time complexity. We had some instances revealing that the times it takes to recover 45 bits of the key is much less than that for 44 bits. This seems very surprising at the first glance, but it can be justified by considering that the running time of ElimLin implementation in [12] is highly dependable on the sparsity of equations. So, our intuition is that as we have picked distinct plaintext and key randomly in each experiment, by pure chance the former system of equations turns out to be sparser than the latter and it is also probable that more linear equations are generated due to some combination of plaintexts and keys randomly picked.

In [1], Albrecht and Cid compared their result with exhaustive key search over PRESENT assuming that checking an arbitrary key takes at least 2 CPU cycles which seems ambitious implying that recovering 45 bits of the key should take at least more than 9 hours, while we could recover the key in less than two hours using only five KP in our best attack.

We tried to break 6 rounds of PRESENT by ElimLin and F4, but ElimLin did not give us any satisfactory result and PolyBori crashed after a while due to probably running out of memory for 6-round PRESENT. In [10], the results are compared with F4 implementation under MAGMA which is specified not to yield any satisfactory results in reasonable time. Although PolyBori crashes in much fewer cases, we could not get anything better by using F4 under PolyBori compared to ElimLin in this specific case.

4 Linear Analysis

Linear cryptanalysis (LC) typically operates under a known-plaintext (KP) or a ciphertext-only (CO) setting, and its origin dates back to the works of Matsui on DES [22]. The main tool for this attack is the *linear relation*, which consists of a linear combination of text and key bits, holding with a relatively high parity deviation from the uniform parity distribution. The effectiveness of a linear relation is measured by a parameter called bias, denoted ϵ, which is the

absolute value of the difference between the parity of the linear relation from 1/2. The higher the bias, the more attractive the linear relations are, since they demand less plaintext-ciphertext pairs. These relations form the core of a linear distinguisher, namely, a tool that allows one to distinguish a given cipher from a random permutation, or to recover subkey bits.

Our linear analysis of the PRESENT cipher started with a study of the Linear Approximation Table (LAT) [22] (Table 6) of its 4×4 S-box (Table 5). In Table 6, the acronym IM stands for the Input Mask (leftmost column); OM for the Output Mask (top row); the entries for non-zero masks are either 0, 2, −2, 4 or −4, that is, the S-box is linearly 4-uniform. Thus, the largest entry for non-trivial (non-zero) bitmasks corresponds to a bias of $4/16 = 2^{-2}$. Thus, entries in the LAT correspond to $16 \cdot \epsilon$, except for the sign; a negative entry implies that the parity is closer to '0', while the absence of a sign means the parity is closer to '1'.

One-round and multiple-round linear approximations were derived by combining the LAT with the bit permutation pLayer that follows each S-box layer. Our analysis indicated that the most promising linear relations shall exploit

- one-bit-input-to-one-bit-output bitmasks in order to minimize the number of active S-boxes per round;
- the pLayer bit permutation following the S-box layer has order three, that is, if we denote this permutation by P, then $\mathsf{P}(\mathsf{P}(\mathsf{P}(X))) = \mathsf{P}^3(X) = X$, for all text blocks X; this fact motivated us to look for iterative relations across three rounds. Particular bit positions of pLayer, though, have much smaller order, such as the leftmost bit in a block which is unaffected by pLayer, that is, it is a fix-point. There are four such fix-points in the pLayer. Thus, the branch number [13] of pLayer is just two. This means that diffusion is quite poor in PRESENT. Due to the fix-points of pLayer, and the LAT profile of the S-box, iterative linear relations exist for any number of rounds.

Taking the order of pLayer into account, it is straightforward to find 3-round iterative linear relations with only three active S-boxes (there cannot be less than one active S-box per round due to the SPN structure). Nonetheless, the S-box design minimizes the bias of single-bit linear approximations. The bias for each such approximation is $2/16 = 2^{-3}$, which gives a maximum bias of $2^{2-3-3-3} = 2^{-7}$ for any 3-round linear relation.

Let us denote a 64-bit mask by

$$\Gamma = \gamma_0\gamma_1\gamma_2\gamma_3\gamma_4\gamma_5\gamma_6\gamma_7\gamma_8\gamma_9\gamma_{10}\gamma_{11}\gamma_{12}\gamma_{13}\gamma_{14}\gamma_{15}$$

where $\gamma_i \in \mathbb{Z}_2^4$, $0 \leq i \leq 15$, that is, a nibble (4 bits). An interesting example of 1-round linear relation for PRESENT is

$$8000000000000000_x \xrightarrow{1r} 8000000000000000_x \tag{1}$$

where the linear approximation $8 \xrightarrow{\mathsf{S}} 8$ for the S-box was used for the leftmost nibble (the leftmost S-box), with bias 2^{-3}, and $\xrightarrow{1r}$ means one round transition. Note that this bias is not the highest possible, but the non-zero bit position in

the mask is a fix-point for the pLayer. Thus, (1) is an iterative linear relation with a single active S-box. If we denote a text block as $X = (x_0, x_1, \ldots, x_{63})$, then (1) can be expressed simply as

$$p_0 \oplus c_0 = k_0^1 \oplus k_0^2, \tag{2}$$

where k_0^1 and k_0^2 are the leftmost (most significant) bits of the first two round subkeys. For a distinguish-from-random attack, the complexity for a single round is $N = 8 \cdot (2^{-3})^{-2} = 2^9$ KP and equivalent parity computations (2). Notice, though, that if the plaintext is composed of ASCII text, then the most significant bit of every plaintext byte is always zero, and the attack actually requires 2^9 ciphertexts only (CO). Iterating (1) for up to 14 rounds requires $8 \cdot (2^{13-3*14})^{-2} = 2^{61}$ CO and an equivalent amount of parity computations. If we allow a lower success probability, we can attack up to 15-round PRESENT using $4 \cdot (2^{14-3*15})^{-2} = 2^{64}$ CO, and equivalent number of parity computations. But, since the codebook is exhausted, the KP or CO settings are the same.

Other two 1-round iterative linear relations with the bias 2^{-3}, also based on fix-points of pLayer are

$$0000000000200000_x \xrightarrow{1r} 0000000000200000_x, \tag{3}$$

and

$$0000040000000000_x \xrightarrow{1r} 0000040000000000_x. \tag{4}$$

A linear relation based on the fourth fix-point of pLayer is not effective since the LAT entry is zero.

An example of 2-round (non-iterative) linear relation for PRESENT with maximum bias is

$$1000000000000000_x \xrightarrow{1r} 0000800000008000_x \xrightarrow{1r} 0808080808080000_x, \tag{5}$$

with bias $2^{2-2-2-2} = 2^{-4}$, and three active S-boxes. The local S-box approximations used were $1 \xrightarrow{\mathsf{S}} 5$ and $8 \xrightarrow{\mathsf{S}} 14$, both with bias 2^{-2}. Reducing the number of active S-boxes to only two across two rounds would decrease the bias to $2^{1-3-3} = 2^{-5}$. Thus, the trade-off of three active S-boxes versus the bias, across two rounds, is the best possible. The attack complexity is $N = 8 \cdot (2^{-4})^{-2} = 2^{11}$ KP and equivalent parity computations.

For three rounds, one of the simplest, most-biased and iterative linear relations we have found is

$$\begin{aligned} &0800000000000000_x \xrightarrow{1r} 4000000000000000_x \xrightarrow{1r} \\ &0000800000000000_x \xrightarrow{1r} 0800000000000000_x, \end{aligned} \tag{6}$$

where the S-box linear approximations were $8 \xrightarrow{\mathsf{S}} 8$ and $4 \xrightarrow{\mathsf{S}} 4$, both with bias 2^{-3}. The overall bias is $2^{2-3-3-3} = 2^{-7}$. Relation (6) is an example that demonstrates a trade-off between the number of active S-boxes per rounds versus the overall bias of linear relations involving single-bit-input-single-bit-output masks.

Relation (6) allows to mount a distinguish-from-random linear attack on 3-round PRESENT with $N = 8 \cdot (2^{-7})^{-2} = 2^{17}$ KP and equivalent number of parity computations (less than one encryption) and negligible memory. For six rounds, the attack complexity becomes $N = 8 \cdot (2^{1-2*7})^{-2} = 2^{29}$ KP and equivalent parity computations. For nine rounds, the complexity becomes $N = 8 \cdot (2^{2-3*7})^{-2} = 2^{41}$ KP and equivalent number of parity computations. For twelve rounds, the attack complexity becomes $N = 8 \cdot (2^{3-4*7})^{-2} = 2^{53}$ KP and equivalent parity computations. For fifteen rounds, if we allow a smaller success rate, then $N = 4 \cdot (2^{4-5*7})^{-2} = 2^{64}$ KP are required, and an equivalent number of parity computations (which is about the effort of one-round computation). Actually, in the 12-round case, the first and last round S-box approximations can be improved, leading to

$$\begin{array}{l} 7700000000000000_x \xrightarrow{1r} 0000C00000000000_x \xrightarrow{1r} 0800000000000000_x \xrightarrow{12r} \\ 0800000000000000_x \xrightarrow{1r} 4000000000000000_x \xrightarrow{1r} 8000000080008000_x, \end{array} \tag{7}$$

where the S-box approximations were $7 \xrightarrow{\mathsf{S}} 4$ with bias 2^{-2} in the 1st round; $C \xrightarrow{\mathsf{S}} 8$ with bias 2^{-2} in the 2nd round; $8 \xrightarrow{\mathsf{S}} 8$ with bias 2^{-3} in the 15th round; $4 \xrightarrow{\mathsf{S}} B$, with bias 2^{-2} in the last round. The notation $\xrightarrow{xr}$ means an x-round transition. The overall bias is $2^{-2-2-2-3\cdot 12-3-2+16} = 2^{-31}$. A distinguish-from-random attack using the 16-round relation (7) costs $N = 4 \cdot (2^{-31})^{-2} = 2^{64}$ KP.

Additional 16-round linear approximations can be derived taking into account other fix-points of pLayer. For instance, using (3):

$$\begin{array}{l} 00000000A0A00000_x \xrightarrow{1r} 0000000000A00000_x \xrightarrow{1r} 0000000000200000_x \xrightarrow{13r} \\ 0000000000200000_x \xrightarrow{1r} 0020000000200020_x, \end{array} \tag{8}$$

where the S-box approximations were $A \xrightarrow{\mathsf{S}} 2$ with bias 2^{-2} in the 1st and 2nd rounds; $2 \xrightarrow{\mathsf{S}} B$ with bias 2^{-2} in the last round. The overall bias is $2^{-3\cdot(2+13)-2+16} = 2^{-31}$.

Further, using (1), we have

$$\begin{array}{l} CC00000000000000_x \xrightarrow{1r} C000000000000000_x \xrightarrow{1r} 8000000000000000_x \xrightarrow{13r} \\ 8000000000000000_x \xrightarrow{1r} 8000800080000000_x, \end{array} \tag{9}$$

where the S-box approximations were $C \xrightarrow{\mathsf{S}} 8$ with bias 2^{-2} in the 1st and 2nd rounds; $8 \xrightarrow{\mathsf{S}} E$ with bias 2^{-2} in the last round. The overall bias is $2^{-3\cdot(13+2)-2+16} = 2^{-31}$.

A 1R key-recovery attack can be applied at the top end of (9) would require guessing the subkeys on top of four S-boxes, because $CC00000000000000_x$ has four active bits. It means a complexity of $2^{64+16}/4 = 2^{78}$ 1-round computations, or $2^{78}/17 \approx 2^{73.91}$ 17-round computations. The memory complexity is a 16-bit counter and the success rate [28] is about 0.37. Recovering subkeys at the bottom

end requires guessing only twelve subkey bits since 8000800080000000_x has only three active bits. It means $2^{64+12} \cdot 3/(16 \cdot 17) \approx 2^{69.50}$ 17-round computations. The memory complexity is a 12-bit counter and the success rate is about 0.63. Applying this attack at both ends (2R attack) requires $2^{64+16+12} \cdot 7/(16 \cdot 18) \approx 2^{86.64}$ 18-round computations (applies only to 128-bit keys). The success rate is about 0.03.

For the remaining 100 key bits, we use (8), which has the same bias as (9). So, the effort to recover further 24+12 key bits is $2^{64+24+12} \cdot 8/(16 \cdot 18) \approx 2^{96.83}$ 18-round computations. The remaining 64 key bits can be found by exhaustive search.

5 Linear Hulls

The concept of linear hulls was first announced by Nyberg in [24]. A linear hull is the LC counterpart to differentials in differential cryptanalysis. Therefore, a linear hull stands for the collection of all linear relations (across a certain number of rounds) that have the same (fixed) input and output bitmasks, but involves different sets of round subkeys according to different linear trails. Consequently, the bias of a linear hull stands for the actual bias of a linear relation involving a given pair of input and output bitmasks. When there is only a single linear trail between a given pair of input and output bitmasks, the concepts of linear relation and linear hull match.

The linear hull effect accounts for a clustering of linear trails, with the consequence that the final bias may become significantly higher than that of any individual trail. Due to Nyberg [24], given the input and output masks a and b for a block cipher $Y = Y(X, K)$, the potential of the corresponding linear hull is denoted

$$ALH(a, b) = \sum_{c} (P(a \cdot X \oplus b \cdot Y \oplus c \cdot K = 0) - \frac{1}{2})^2 = \epsilon^2 \qquad (10)$$

where c is the mask for the subkey bits. Then, key-recovery attacks such as Algorithm 2 in [22] apply with

$$N = \frac{t}{ALH(a, b)} = \frac{t}{\epsilon^2}$$

known plaintexts, where t is a constant. An advantage to use linear hulls in key-recovery attacks, such as in Algorithm 2, is that the required number of known plaintexts can be decreased for a given success rate. Keliher *et al.* exploited this method to attack the Q cipher [19].

For PRESENT, in particular, it makes sense to choose input/output masks that affect only a few S-boxes, because it minimizes the number of key bits to guess in key-recovery attacks around the linear hull distinguisher. Moreover, minimizing the number of active S-boxes in the first round may also minimize the number of linear trails to look for, which speeds up our search program for all possible linear paths and the corresponding bias computation.

Our approach to determine linear hulls for PRESENT used a recursive algorithm (in ANSI C) employing a depth-first search strategy. It is a classical technique to find exhaustively all linear trails systematically and with low memory cost. Fixed input and output (I/O) bitmasks and a number of rounds were provided as parameters, and the algorithm computed all possible linear trails starting and ending in the given I/O masks, the corresponding biases and the number of active S-boxes. One objective of the linear hull search was to double check if the linear relations we derived in (1), (5) and (6) actually had the predicted biases (which have been confirmed).

An interesting phenomenon we have observed is the rate of decrease of the bias in linear hulls with some fixed input/output bitmasks of low Hamming Weight (HW), for increasing number of rounds. In particular, we have focused on a few cases, where the input and output masks are the same (iterative linear relations) and have low HW. We have studied all 64-bit input and output bit masks with HW = 1. Further, to optimize the search, we have focused only on the linear trails with the highest bias (single-bit trails), which we call the best trails (with a single active S-box per round). The best results we obtained concern the mask 0000000000200000_x (both at the input and at the output). Table 2 summarizes our experiments, where "computed bias" denotes the bias of the linear hull for the given number of rounds computed according to 10. For up to four rounds, all trails were found. For five rounds or more, only the trails with highest biases were accounted for. The values under the title "expected bias" indicate the bias as computed by the Piling-up lemma.

From Table 2, we observe that the bias for linear hulls in PRESENT does not decrease as fast, with increasing number of rounds, as in linear relations as dictated by the Piling-up lemma. Fig.2 compares the computed and the predicted bias values in Table 2. Our experimental results indicate that the linear hull effect is significant in PRESENT even for a small number of rounds. For five rounds or more, we could not determine all linear trails, but we looked for the ones with the highest bias values, so that their contribution to the overall ALH would be significant. We have searched for linear trails with r up to $r + 2$ active S-boxes across r rounds. Thus, the values for more than four rounds represent a lower bound on the overall bias of the linear hulls.

In Table 2, consider the linear hull across five rounds. We have found nine trails with bias 2^{-11} inside this linear hull. Repeating it three times, we arrive at 9^3 15-round linear trails. The ALH (0000000000200000_x, 0000000000200000_x) for 15 rounds is $(2^{-31})^2 \cdot 9^3 = 2^{-62+9.51} = 2^{-52.49}$. We extend this 15-round linear hull to a 17-round linear hull with 9^3 17-round linear trails by choosing an additional 1-round relation at the top and at the bottom ends of it:

$$\begin{aligned} &0000000000A00000_x \xrightarrow{1r} 0000000000200000_x \xrightarrow{15r} \\ &0000000000200000_x \xrightarrow{1r} 0020000000200020_x, \end{aligned} \tag{11}$$

where the ALH for the 17-round linear hull is $(2^{-33})^2 \cdot 9^3 = 2^{-66+9.51} = 2^{-56.49}$. This linear hull can be used to distinguish 17-round PRESENT from a random permutation with $2^{56.49} \cdot 8 = 2^{59.49}$ KP, and equivalent parity computations.

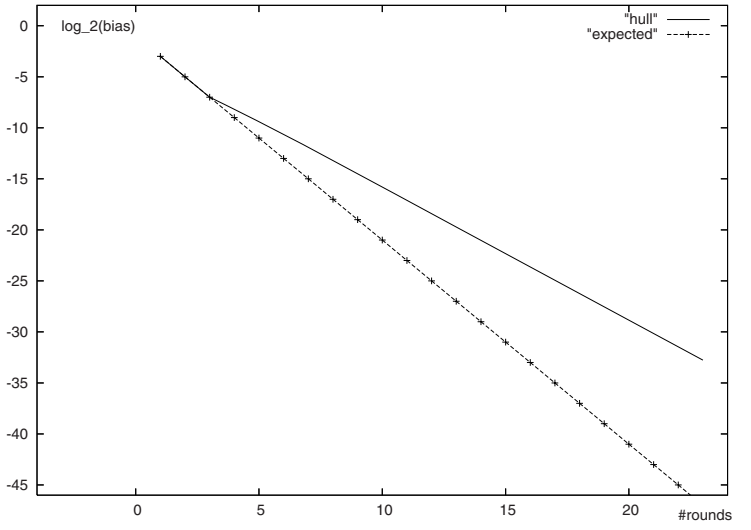

Fig. 2. Behaviour of linear hull bias (ALH) against expected bias (by Piling-up lemma) for increasing number of rounds of PRESENT (data from Table 2)

Table 2. Computed bias (cb) and expected bias (eb) of linear hulls in PRESENT for input/output mask 0000000000200000_x

# rounds	1	2	3	4	5	6	7	8
# trails	1	1	1	9	9	27	72	192
(cb)	2^{-3}	2^{-5}	2^{-7}	$2^{-8.20}$	$2^{-9.40}$	$2^{-10.61}$	$2^{-11.90}$	$2^{-13.19}$
(eb)	2^{-3}	2^{-5}	2^{-7}	2^{-9}	2^{-11}	2^{-13}	2^{-15}	2^{-17}
# rounds	9	10	11	12	13	14	15	16
# trails	512	1344	3528	9261	24255	63525	166375	435600
(cb)	$2^{-14.48}$	$2^{-15.78}$	$2^{-17.08}$	$2^{-18.38}$	$2^{-19.71}$	$2^{-21.02}$	$2^{-22.33}$	$2^{-23.63}$
(eb)	2^{-19}	2^{-21}	2^{-23}	2^{-25}	2^{-27}	2^{-29}	2^{-31}	2^{-33}
# rounds	17	18	19	20	21	22	23	
# trails	1140480	2985984	7817472	20466576	53582633	140281323	367261713	
(cb)	$2^{-24.94}$	$2^{-26.25}$	$2^{-27.55}$	$2^{-28.85}$	$2^{-30.16}$	$2^{-31.47}$	$2^{-32.77}$	
(eb)	2^{-35}	2^{-37}	2^{-39}	2^{-41}	2^{-43}	2^{-45}	2^{-47}	

Applying a key-recovery (1R attack) at the top end of (11) requires guessing only eight bits because there are only two active bits in $0000000000A00000_x$. The attack complexity becomes $2^{59.49+8} \cdot 2/(16 \cdot 18) = 2^{60.33}$ 18-round computations. The memory complexity is just an 8-bit counter and the success rate is about 0.997.

For six rounds, and still using mask 0000000000200000_x, we have found 27 trails, each with bias 2^{-13} inside this linear hull. Concatenating the linear hull three times, we arrive at 27^3 18-round trails. The ALH (0000000000200000_x, 0000000000200000_x) for 18 rounds is $(2^{-37})^2 \cdot 27^3 = 2^{-59.73}$. Extending it to 20

rounds by choosing carefully an additional relation on top and at the bottom of it results in

$$\begin{aligned} &0000000000A00000_x \xrightarrow{1r} 0000000000200000_x \xrightarrow{18r} \\ &0000000000200000_x \xrightarrow{1r} 0020000000200020_x. \end{aligned} \tag{12}$$

The ALH of (12) is $(2^{-39})^2 \cdot 27^3 = 2^{-63.73}$. Thus, (12) can be used to distinguish 20-round PRESENT from a random permutation using the full codebook. A key-recovery (1R) attack at the top of (12) leads to a complexity of $2^{64+8}/(8 \cdot 21) \approx 2^{64.60}$ 21-round computations. The memory needed is an 8-bit counter and the success rate is about 0.246. For 80-bit keys, the remaining 80-12 = 68 key bits can be found by exhaustive search.

For nine rounds and mask 0000000000200000_x, we have found 512 trails, each with bias 2^{-19} inside this linear hull. Concatenating the linear hull twice, we arrive at 512^2 18-round trails. The ALH (0000000000200000_x, 0000000000200000_x) for 18 rounds is $(2^{-37})^2 \cdot 512^2 = 2^{-56}$. Extending it to 20 rounds, just like (12), leads to an ALH of $(2^{-39})^2 \cdot 512^2 = 2^{-60}$. A key-recovery (1R) attack at the top of this linear hull results in a complexity of $8 \cdot 2^{60+8}/(8 \cdot 21) \approx 2^{63.60}$ 21-round computations. The memory needed is an 8-bit counter and the success rate is about 0.997. For 80-bit keys, the remaining 72 key bits can be found by exhaustive search, leading to a complexity of 2^{72} encryptions.

For ten rounds and mask 0000000000200000_x, we have found 1344 trails, each with bias 2^{-21} inside this linear hull. Concatenating the linear hull twice, we arrive at 1344^2 20-round trails. The corresponding ALH (0000000000200000_x, 0000000000200000_x) for 20 rounds is $(2^{-41})^2 \cdot 1344^2 = 2^{-61.22}$. Extending it to 21 rounds leads to

$$\begin{aligned} &0000000000A00000_x \xrightarrow{1r} 0000000000200000_x \xrightarrow{20r} \\ &0000000000200000_x, \end{aligned} \tag{13}$$

with an ALH of $(2^{-42})^2 \cdot 1344^2 = 2^{-63.21}$. A key-recovery (2R) attack at both ends of this linear hull requires guessing 16 key bits. The effort becomes $2^{63.21+16}/(16 \cdot 23) \approx 2^{60.68}$ 23-round computations. The memory needed is a 16-bit counter. For 80-bit keys the remaining 64 key bits can be found by exhaustive search, leading to a final complexity of 2^{64} encryptions.

For the 21-round linear hull, with bitmask 0000000000200000_x, we have found 53582633 trails with bias 2^{-43} and the accumulated bias is $2^{-30.16}$. These trails always have one single active S-box per round. In order to improve the accumulated bias, we identify the second best trails across 21 rounds in which 23 active S-boxes are involved. Unlike the best trails, the second best ones have a '2-way branching' that is the trail splits from one to two S-boxes. This branching later merges back into a single S-box (Fig. 3) after three rounds. We developed another depth-first search program to find the 2nd-best trails for a variable number of rounds. The results are listed in Table 3. From the empirical results in Table 3, the number of 2nd best trails seems to be (# rounds-3) times more than the number of best trails. This means that the contribution of the second best trails to the overall bias of the 22-round hull is about $\sqrt{18 \cdot 53582633} \cdot 2^{-47}$ or

Table 3. Number of second best trails using bitmask 0000000000200000_x

# rounds	# best trails	# 2nd best trail	bias of 2nd best trails
5	9	18	$2^{-12.915}$
6	27	81	$2^{-13.830}$
7	72	288	$2^{-14.915}$
8	192	960	$2^{-16.046}$
9	512	3072	$2^{-17.207}$
10	1344	9536	$2^{-18.376}$
11	3528	28896	$2^{-19.565}$
12	9261	85995	$2^{-20.771}$
13	24255	252021	$2^{-21.990}$
14	63525	730235	$2^{-23.219}$

$2^{-32.077}$. Combining the biases of the 1st best trails and 2nd best trails results in $\sqrt{(2^{-30.16})^2 + (2^{-32.077})^2} \sim 2^{-30.11}$.

We now make the key recovery attack on 25-round by guessing 20 bits at both ends of the 21-round linear hull. This means $2^{64+16+16+4+4} \cdot 10/16 = 2^{103.33}$ 1-round computations, or $2^{103.33}/25 \approx 2^{98.68}$ 25-round computations, which only applies to 128-bit keys. For the remaining 88-bit subkey, we can search it exhaustively. The success rate is 0.61.

For the 22-round linear hull, with bitmask 0000000000200000_x, we have found 140281323 trails, each with bias 2^{-45} and $\sqrt{19 \cdot 140281323} \cdot 2^{-49}$ trails each with bias 2^{-49}. The corresponding ALH is $(2^{-45})^2 \cdot 140281323 + (19 \cdot 140281323) \cdot 2^{-49} \approx 2^{-62.83}$, which means an accumulated bias of $2^{-31.415}$. We use this 22-round linear hull to make a key recovery attack on 26-round PRESENT. This means $2^{64+16+16+4+4} \cdot 10/16 = 2^{103.33}$ 1-round computations, or $2^{103.33}/26 \approx 2^{98.62}$ 26-round computations, which only applies to 128-bit keys. The success rate is only 0.00002.

It is reasonable that the linear trails in a linear hull could not be independent. Kaliski *et al.*, though, showed that the linear dependency of the linear approximations has no effect for the attack [20].

6 Conclusions

This paper described the first linear hull attacks and revisited algebraic attacks with a comparison between two distinct algorithms on reduced-round versions of the block cipher PRESENT. The analysis based on linear hulls were used to detect any significant variation in the bias, which would impact the linear attack complexities; and, to assess the linear hull effect in PRESENT and its resilience to LC. We have confirmed that the linear hull effect is significant even for a small number of rounds of PRESENT.

Table 4 lists the attack complexities for PRESENT for increasing number of rounds and in increasing order of time complexity.

Table 4. Attack complexities on reduced-round PRESENT block cipher

#Rounds	Time	Data	Memory	Key Size (bits)	Source	Comments
5	2.5 min	5 KP	—	80	Sect. 3	KR†, AC
5	2.5 min	5 KP	—	128	Sect. 3	KR†, AC
5	1.82 h	64 KP	—	80	[10]	KR, AC
6	2^{26}	2^{26} KP	—	all	eq. (6)	DR* + KR, LC
7	$2^{100.1}$	$2^{24.3}$ CP	2^{77}	128	[30]	IC
14	2^{61}	2^{61} CO	—	all	eq. (1)	DR* + KR, LC
15	$2^{35.6}$	$2^{35.6}$ CP	2^{32}	all	[5]	KR, SC
15	2^{64}	2^{64} KP	—	all	eq. (1)	DR*, LC
16	2^{62}	2^{62} CP	1Gb	all	[1]	KR, AC + DC
16	2^{64}	2^{64} CP	2^{32}	all	[29]	KR, DC
17	$2^{69.50}$	2^{64} KP	2^{12}	80	eq. (9)	KR, LC
17	$2^{73.91}$	2^{64} KP	2^{16}	80	eq. (9)	KR, LC
17	2^{104}	2^{63} CP	2^{53}	128	[25]	KR, RKR
17	2^{93}	2^{62} CP	1Gb	128	[1]	KR, AC + DC
18	2^{98}	2^{62} CP	1Gb	128	[1]	KR, AC + DC
19	2^{113}	2^{62} CP	1Gb	128	[1]	KR, AC + DC
24	2^{57}	2^{57} CP	2^{32}	all	[5]	KR, SSC
25	$2^{98.68}$	2^{64} KP	2^{40}	128	Table 2	KR, LH
26	$2^{98.62}$	2^{64} KP	2^{40}	128	Table 2	KR, LH

*: time complexity is number of parity computations; †: recover half of the user key;
DR: Distinguish-from-Random attack; KR: Key Recovery attack
LC: Linear Cryptanalysis; DC: Differential Cryptanalysis; AC: Algebraic Crypt.;
SSC: Statistical Saturation analysis; IC: Integral Cryptanalysis;
RKR: Related-Key Rectangle; LH: Linear Hull; ML: Multiple Linear;
CP: Chosen Plaintext; KP: Known Plaintext; CO: Ciphertext Only.

A topic for further research is to look for the 3rd and 4th best trails inside a linear hull. The issue is to find out their contribution to the overall bias of the linear hulls, that is, if they can further improve the bias as the 2nd best trails did.

Acknowledgements

We would like to thank anonymous reviewers for their very important comments.

References

1. Albrecht, M., Cid, C.: Algebraic Techniques in Differential Cryptanalysis. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5565, pp. 193–208. Springer, Heidelberg (2009)
2. Brickenstein, M., Dreyer, A.: PolyBoRi: A framework for Gröbner basis computations with Boolean polynomials. Electronic Proceedings of MEGA (2007), http://www.ricam.oeaw.ac.at/mega2007/electronic/26.pdf

3. 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)
4. Buchberger, B.: An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal. Ph.D Dissertation (1965)
5. Collard, B., Standaert, F.X.: A Statistical Saturation Attack against the Block Cipher PRESENT. In: CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
6. Courtois, N., Shamir, A., Patarin, J., Klimov, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. Adv. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
7. Courtois, N., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
8. Courtois, N., Meier, W.: Algebraic Attacks on Stream Ciphers with Linear Feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
9. Courtois, N.T., Bard, G.V.: Algebraic cryptanalysis of the data encryption standard. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 152–169. Springer, Heidelberg (2007)
10. Courtois, N.T., Debraize, B.: Specific S-Box Criteria in Algebraic Attacks on Block Ciphers with Several Known Plaintexts. In: Lucks, S., Sadeghi, A.-R., Wolf, C. (eds.) WEWoRC 2007. LNCS, vol. 4945, pp. 100–113. Springer, Heidelberg (2008)
11. Courtois, N.T., Bard, G.V., Wagner, D.: Algebraic and slide attacks on keeLoq. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 97–115. Springer, Heidelberg (2008)
12. Courtois, N.T.: Tools for experimental algebraic cryptanalysis, `http://www.cryptosystem.net/aes/tools.html`
13. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
14. Eén, N., Sörensson, N.: MiniSat 2.0. An open-source SAT solver package, `http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/`
15. Faugére, J.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra, 61–69 (1999)
16. Faugére, J.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Symbolic and Algebraic Computation - ISSAC, pp. 75–83 (2002)
17. Ghasemzadeh, M.: A New Algorithm for the Quantified Satisfiability Problem, Based on Zero-suppressed Binary Decision Diagrams and Memorization. Ph.D. thesis, Potsdam, Germany, University of Potsdam (2005), `http://opus.kobv.de/ubp/volltexte/2006/637/`
18. Indesteege, S., Keller, N., Dunkelman, O., Biham, E., Preneel, B.: A practical attack on keeLoq. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 1–18. Springer, Heidelberg (2008)
19. Keliher, L., Meijer, H., Tavares, S.: High Probability Linear Hulls in Q. In: Second NESSIE Conference (2001)
20. Kaliski, B.S., Robshaw, M.J.B.: Linear Cryptanalysis Using Multiple Approximations and FEAL. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 249–264. Springer, Heidelberg (1995)

21. Magma, software package, http://magma.maths.usyd.edu.au/magma/
22. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
23. Murphy, S., Robshaw, M.J.B.: Essential Algebraic Structure within AES. Adv. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 1–16. Springer, Heidelberg (2002)
24. Nyberg, K.: Linear approximation of block ciphers. Adv. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
25. Özen, O., Varici, K., Tezcan, C., Kocair, Ç.: Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT. In: ACISP 2009. LNCS, vol. 5594, pp. 90–107. Springer, Heidelberg (2009)
26. Raddum, H., Semaev, I.: New technique for solving sparse equation systems. Cryptology ePrint Archive, Report 2006/475 (2006), http://eprint.iacr.org/2006/475
27. Shannon, C.E.: Claude Elwood Shannon collected papers. Wiley-IEEE Press, Piscataway (1993)
28. Selçuk, A.A., Biçak, A.: On probability of success in linear and differential cryptanalysis. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 174–185. Springer, Heidelberg (2003)
29. Wang, M.: Differential Cryptanalysis of reduced-round PRESENT. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 40–49. Springer, Heidelberg (2008)
30. Z'aba, M.R., Raddum, H., Henricksen, M., Dawson, E.: Bit-Pattern Based Integral Attack. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 363–381. Springer, Heidelberg (2008)

A ElimLin Intermediate Results

Table 7 depicts the intermediate ElimLin results for 5-round PRESENT-80 where 36 bits of the key are fixed and we try to recover the remaining key bits. In the third column, T represents the total number of monomials and Ave is the average number of monomials per equation.

Table 5. The 4×4-bit S-box of PRESENT and the inverse S-box

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathsf{S}[x]$	12	5	6	11	9	0	10	13	3	14	15	8	4	7	1	2
$S^{-1}[x]$	5	14	15	8	12	1	2	13	11	4	6	3	0	7	9	10

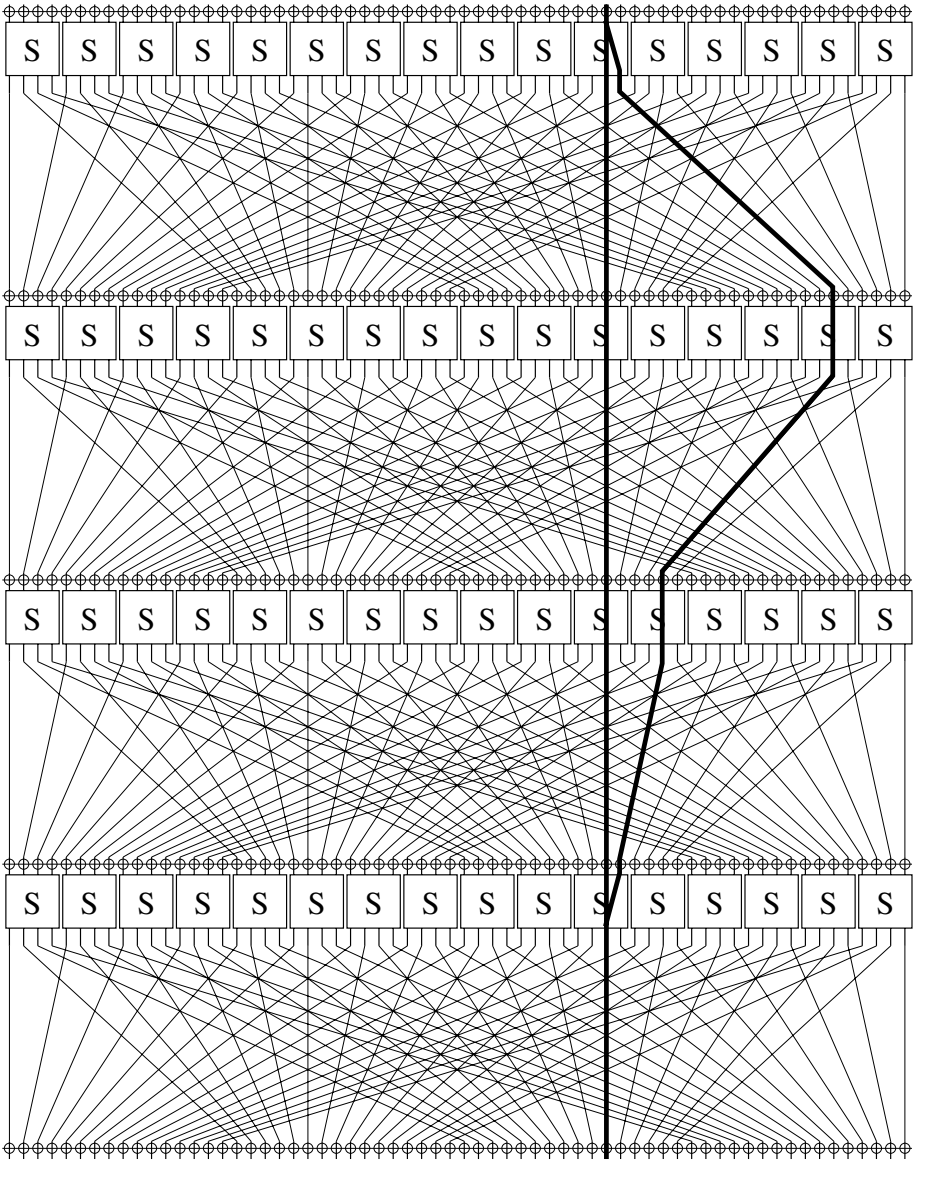

Fig. 3. Example of branching inside a trail, from single S-box to two S-boxes, and merging back to one S-box

Table 6. Linear Approximation Table (LAT) of the S-box of PRESENT

	OM															
IM	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	-4	0	-4	0	0	0	0	0	-4	0	4
2	0	0	2	2	-2	-2	0	0	2	-2	0	4	0	4	-2	2
3	0	0	2	2	2	-2	-4	0	-2	2	-4	0	0	0	-2	-2
4	0	0	-2	2	-2	-2	0	4	-2	-2	0	-4	0	0	-2	2
5	0	0	-2	2	-2	2	0	0	2	2	-4	0	4	0	2	2
6	0	0	0	-4	0	0	-4	0	0	-4	0	0	4	0	0	0
7	0	0	0	4	4	0	0	0	0	-4	0	0	0	0	4	0
8	0	0	2	-2	0	0	-2	2	-2	2	0	0	-2	2	4	4
9	0	4	-2	-2	0	0	2	-2	-2	-2	-4	0	-2	2	0	0
10	0	0	4	0	2	2	2	-2	0	0	0	-4	2	2	-2	2
11	0	-4	0	0	-2	-2	2	-2	-4	0	0	0	2	2	2	-2
12	0	0	0	0	-2	-2	-2	-2	4	0	0	-4	-2	2	2	-2
13	0	4	4	0	-2	-2	2	2	0	0	0	0	2	-2	2	-2
14	0	0	2	2	-4	4	-2	-2	-2	-2	0	0	-2	-2	0	0
15	0	4	-2	2	0	0	-2	-2	-2	2	4	0	2	2	0	0

Table 7. ElimLin result for 5-round PRESENT-80 when 36 bits of key are fixed

# Variables	# Equations	(Ave/ # Monomials)	# Linear Equations
10340	46980	7/ T= 46321	6180
4160	46980	8/ T= 48744	1623
2537	46980	9/ T= 40763	1069
1468	46980	14/ T= 43155	405
1063	46980	76/ T= 73969	165
898	46980	158/ T= 145404	201
697	46980	77/ T= 85470	584
113	46980	0/ T= 413	113