J. Cryptol. (2013) 26: 11–38
DOI: 10.1007/s00145-011-9111-4

Journal of
CRYPTOLOGY

Slender-Set Differential Cryptanalysis

Julia Borghoff, Lars R. Knudsen, Gregor Leander, and Søren S. Thomsen

Department of Mathematics, Technical University of Denmark, Lyngby, Denmark
S.Thomsen@mat.dtu.dk

Communicated by Willi Meier

Received 13 May 2011
Online publication 27 October 2011

Abstract. This paper considers PRESENT-like ciphers with key-dependent S-boxes. We focus on the setting where the same selection of S-boxes is used in every round. One particular variant with 16 rounds, proposed in 2009, is broken in practice in a chosen plaintext/chosen ciphertext scenario. Extrapolating these results suggests that up to 28 rounds of such ciphers can be broken. Furthermore, we outline how our attack strategy can be applied to an extreme case where the S-boxes are chosen uniformly at random for each round, and where the bit permutation is key-dependent as well.

Key words. Symmetric key, Block cipher, PRESENT, Differential cryptanalysis.

1. Introduction

Small computing devices are becoming more and more popular and establish a part of the pervasive communication infrastructure. One example of these tiny computing devices are RFID systems which are used, for instance, for identifying and tracking animals and on toll roads. A prediction for the future is that RFID tags will replace bar codes. But this extensive deployment of computing devices is not only useful and convenient, it also carries a wide range of security risks. At the same time, we are talking about extremely resource constrained environments. Therefore, the demand for lightweight encryption algorithms increases. The block cipher PRESENT [4] is an important example of a lightweight cipher. It consists of alternate layers of substitutions and permutations.

Important design principles of lightweight ciphers are efficient hardware implementation, good performance, and a moderate security level. Usually, there is a trade-off between the performance and the security level. In order to speed up the algorithm, we want as few rounds of encryption as possible, but there is a minimum number of rounds required to assure the security level.

PRESENT is a 64-bit iterated block cipher that comes in two variants, one with an 80-bit key and one with a 128-bit key. Both run in 31 rounds, each round has three layers, a substitution layer consisting of 16 parallel applications of the same 4-bit S-box,

a permutation layer consisting of a bit-wise permutation of 64 bits, and a key addition layer, where a subkey is exclusive-ored to the text. PRESENT was designed to allow fast and compact implementation in hardware. The best known cryptanalytic attack on PRESENT is a linear attack on 26 of the 31 rounds [6]. The attack requires all possible 2^{64} texts and has a running time of 2^{72}. Although this attack is hardly practical, it illustrates that the number of rounds used should not be dramatically reduced.

An idea of how to strengthen the cipher in a way that enables one to reduce the number of rounds has been presented by two researchers from Princeton University. The cipher Maya [10] is a 16-round SP-network similar to PRESENT. The main difference is that the substitution layer of Maya consists of 16 different S-boxes which are key dependent and therefore kept secret. The bit permutation between the S-box layers is fixed and public. In each round, a round key is xored to the text. It is argued that this cipher can be implemented efficiently in practice and also that "differential cryptanalysis is infeasible".

The Maya design is one particular way of designing a PRESENT-like cipher with secret components. In an extreme case, one could choose 16 S-boxes uniformly at random and independently for every round. Furthermore, one could also make the bit permutation part of the key, chosen uniformly at random from the set of all such permutations and used repeatedly, or as another extreme, a bit permutation could be chosen for each round uniformly at random and independently for every round.

The idea of having ciphers where the substitutions are not publicly known, but part of the secret key, is not new. Notable examples are Khufu [14], the Khufu variation Blowfish [15], GOST [11], as well as other proposals [2,16].

Our Results In this paper, we present a novel differential-style attack which enables us to find the secret S-boxes one by one. To recover an S-box, we identify sets of eight input pairs that yield the same output difference of Hamming weight one for this S-box. We call these *slender sets*.

The fundamental idea of the attack is to consider a (collection of) truncated differential(s) for the full cipher to identify slender sets of a single S-box in the first round, and thus infer information about this S-box. The main assumption is that the probability for the truncated differentials is higher when the input difference to the second round has Hamming weight one. This is the key for using a differential-style attack without actually having to specify differential characteristics of differentials.

We apply the attack to PRESENT-like ciphers, where we focus on the simple case where every round has the same selection of secret S-boxes and the same publicly known bit permutation.

The attack was implemented and successfully recovered the secret S-boxes in versions up to 16 rounds. The time complexity of the attack on the 16-round version is approximately 2^{38} using a similar number of chosen plaintexts/chosen ciphertexts. In particular, we implemented a successful attack on the proposed cipher Maya. In our experiments, the correct S-boxes were usually found in less than one week on a standard PC.

To better understand the running time of the attack, we establish a simplified, mathematical model for the complexity of this attack and verify by numerous experiments

that the model fits the real world. Extrapolations of the experimental data, backed up by our model, indicate that the attack has the potential to break up to 28 rounds with a chosen plaintext complexity less than 2^{64}.

Furthermore, we outline how even the extreme case of PRESENT-like ciphers with secret components, that is, the case where all components in all rounds are chosen independently and uniformly at random, can be attacked.

We also apply our attack to PRINTCIPHER [12], which can be seen as a PRESENT-like cipher with secret S-boxes, although there are some subtle, but rather important differences, which have the consequence that our attack is only moderately successful.

Moreover, we discuss a linear-style attack which follows a similar principle as the differential-style attack. Here, linear information is used to group the 16 possible inputs to an S-box into two sets, where the S-box output of each element within a set generates the same value for a certain unknown linear characteristic. This approach seems to be less efficient than the differential-style attack.

Related Work Biryukov and Shamir investigated the security of iterated ciphers where the substitutions and permutations are all key-dependent [3]. In particular, they analysed an AES-like cipher with 128-bit blocks using eight-bit S-boxes. An attack was presented on five layers (SASAS, where S stands for a substitution and A stands for an affine mapping) of this construction which finds all secret components (up to an equivalence) using 2^{16} chosen plaintexts and with a time complexity of 2^{28}. Using the terminology of "rounds" as in the AES, this version consists of two and a half rounds.

The extreme case of our cipher, where the S-boxes and the bit-permutation are chosen at random for each round, is a special instance of the SASAS cipher [3]. In fact, the attack of Biryukov and Shamir applies to three rounds of this variant and has a running time of 2^{16} using 2^8 chosen texts. However, the complexity of the attack for more than three rounds is unclear, but seems to grow very quickly [3]. The SASAS attack is a multiset attack, whereas we use a differential-style attack to recover the S-boxes. Also, the technique to recover the bit permutation is different.

There have been other attempts to cryptanalyse ciphers with secret S-boxes. Gilbert and Chauvaud presented a differential attack on the cipher Khufu [9]. Khufu is an unbalanced Feistel cipher, and the attack exploits the relatively slow diffusion in the cipher and bears some resemblance with our work. Also, Vaudenay provided cryptanalysis of reduced-round variants of Blowfish [17]. Moreover, the cipher C2, which has a secret S-box, was cryptanalysed by Borghoff et al. [5].

Organisation The paper is organised as follows. In Sect. 2, the target cipher and its generalisation are presented. Section 3 explains the approach for recovering the secret S-boxes. In Sect. 4, practical issues of the attack are discussed. In Sect. 5, we give experimental results for the attack applied to the Maya cipher [10]. Section 6 describes our model to back up the extrapolations of the experimental data. In Sect. 7, we show how to use a linear-style attack to break the cipher, and we outline an attack on the generalised cipher in Sect. 8. In Sect. 9, we apply our attack to PRINTCIPHER. We conclude in Sect. 10.

Require: X is a 64-bit plaintext
Ensure: $C = E_K(X)$ where E_K means the encryption function with key K
1: Derive 16 four-bit S-boxes S_i and N 64-bit round keys K_i from K
2: STATE $\leftarrow X$
3: **for** $i = 1$ to N **do**
4: Apply S-box S_j to jth four-bit nibble STATE_j of STATE, $0 \leq j \leq 15$
5: Apply bit permutation to STATE
6: Add round key K_i to STATE
7: **end for**
8: $C \leftarrow$ STATE

Algorithm 1: Pseudo-code of a PRESENT-like cipher with secret S-boxes. The number of rounds is N.

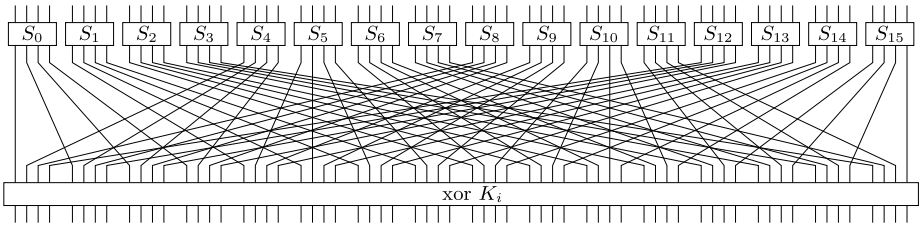


Fig. 1. One round of the cipher of Algorithm 1 with an example bit permutation.

2. The Cipher

We now describe our target cipher, which is a PRESENT-like cipher where the secret consists of one round key for each round and 16 S-boxes. We assume that the round keys and the S-boxes are randomly chosen. In practice, these secret components will be derived from a master key. The 16 S-boxes form the substitution layer which is used repeatedly throughout all the rounds. The permutation layer consists of a bit permutation which is fixed and publicly known. It is reasonable to assume that this bit permutation is such that it ensures a good diffusion across different S-boxes. In particular, it is reasonable to assume that the 4 output bits from one S-box are mapped to 4 different S-boxes in the next round.

One round of encryption works as described in Algorithm 1 (see also Fig. 1). The state is first initialised with the plaintext. In each of the N rounds, the state is divided into nibbles of four bits which are processed by the 16 S-boxes in parallel. Then the bit permutation is applied to the concatenation of the output of the S-boxes, and the output is xored with a round key. The last value of the state is the ciphertext.

The cipher Maya, proposed by Gomathisankaran and Lee [10], is an instance of the cipher described in Algorithm 1 with $N = 16$.

We attack this cipher by recovering all 16 S-boxes. However, in the general case, we do not know the last round key, and therefore what we recover is, in fact, the S-boxes $S_i(x) \oplus K_{N,i}$, where $K_{N,i}$ is the ith nibble of the last round key K_N. Once this is done,

Require: X is a 64-bit plaintext
Ensure: $C = E_K(X)$ where E_K means the encryption function with key K
1: Derive $16 \cdot N$ four-bit S-boxes $S_{i,j}$, $1 \leq i \leq N$, $0 \leq j \leq 15$ and $N - 1$ 64-bit bit permutations P_i from K
2: STATE $\leftarrow X$
3: **for** $i = 1$ to $N - 1$ **do**
4: Apply S-box $S_{i,j}$ to jth four-bit nibble STATE_j of STATE, $0 \leq j \leq 15$
5: Apply bit permutation P_i to STATE
6: **end for**
7: Apply S-box $S_{N,j}$ to STATE_j, $0 \leq j \leq 15$
8: $C \leftarrow$ STATE

Algorithm 2: Pseudo-code of a PRESENT-like cipher with secret S-boxes and secret bit permutations, all unique for each of the N rounds.

we can peel off the first and last layers of encryption, and attack the cipher with two rounds less; this time, the S-boxes are known, and a standard differential or linear attack can be mounted to extract the round keys. What we obtain in the end is an equivalent description of the cipher, but not necessarily the key. Still, the equivalent description of the cipher will allow us to encrypt or decrypt any text of our choice.

A slight variation of Algorithm 1 also adds a round key K_0 at the beginning. This simply means that we recover S-boxes $S_i(x \oplus K_{0,i}) \oplus K_{N,i}$, and not $S_i(x) \oplus K_{N,i}$ as above. However, this makes no difference to the attack, which proceeds as above.

We also outline how our attack can be applied to a generalisation. Here, the S-boxes are chosen uniformly at random for each round. Additionally, the bit permutation can be chosen randomly for each round and kept secret as part of the key. In this case, the addition of the round keys is not necessary because it can be seen as part of the S-boxes. Furthermore, the permutation is omitted in the last round. This allows an easier adaptation of our attack ideas from Algorithm 1 to Algorithm 2.

This extreme variant can be compared with an instance of SASAS [3]. Note that in this variant, only the general structure of the cipher, the block size, and the number of rounds is known. This variant is described in Algorithm 2.

3. Principle of the Attack

In this section, we explain the idea of our approach to recover the S-boxes in the basic variant of a PRESENT-like cipher with secret S-boxes. It is a differential-style attack and the complexity is analysed in Sect. 6.

3.1. *Preliminaries*

Recall that in the basic variant of the cipher (cf. Algorithm 1), there are 16 secret S-boxes which are applied in all rounds. We denote by E_K the encryption function with key K, and by S_i, $0 \leq i < 16$, the 16 S-boxes; all S-boxes S_i are bijective mappings with the signature $\mathbb{F}_2^4 \to \mathbb{F}_2^4$. We shall often identify elements of $\mathbb{F}_2^4$ with hexadecimal digits $0, 1, \ldots, f$ in the natural manner.

For convenience, we introduce the following notation.

Definition 1. Given the S-box S and $e \in \mathbb{F}_2^4$, we denote the set of all pairs $\{x, y\}$ such that $S(x) \oplus S(y) = e$ by D_e. Here, we consider the pairs $\{x, y\}$ and $\{y, x\}$ to be identical. A pair $\{x, y\}$ belonging to a set D_e where e has Hamming weight 1 is called a *slender* pair. A set consisting of slender pairs is called a slender set.

Without loss of generality, we explain how to recover the leftmost S-box S_0. In order to obtain information about S_0, we encrypt a certain number t of structures P_{r_i} of plaintexts of the form

$$P_{r_i} = \left\{(x \| r_i) \mid x \in \mathbb{F}_2^4\right\}$$

where each $r_i \in \mathbb{F}_2^{60}$ for $0 \leq i < t$ is chosen uniformly at random. Two different plaintexts $(x\|r_i), (y\|r_i)$ in P_{r_i} have an input difference of the form

$$(x \| r_i) \oplus (y \| r_i) = \left(? \| 0^{60}\right),$$

where 0^n denotes the bit string consisting of n zeros.

We shall be looking at the corresponding ciphertexts in order to see if there is an input pair for which only one S-box is active in the ciphertext. For now, let $p(\{x, y\})$ denote the probability that only one S-box is active in the ciphertext difference when the plaintext pair is $\{x\|r, y\|r\}$, taken over all the different choices of $r \in \mathbb{F}_2^{60}$. The attack is based on some assumptions. The first assumption is a standard one in differential cryptanalysis:

Assumption 1. *The probability $p(\{x, y\})$ depends only on the value of $S(x) \oplus S(y)$, not specifically on the pair $\{x, y\}$. Hence, given $e = S(x) \oplus S(y)$, we can denote this probability p_e.*

We shall be particularly interested in identifying slender pairs. In order to do this, we need the following assumption.

Assumption 2. *The probability p_e is higher when e has Hamming weight* 1, *than when e has Hamming weight greater than* 1.

The differential $0 \xrightarrow{S} 0$ has probability 1 for any S-box S, and all other differentials are likely to have probability less than 1. Combined with the bit permutation, which is assumed to map output bits from the same S-box to four different S-boxes in the next round in order to maximise diffusion, this means that the weight of the output difference after the first S-box layer determines the number of active S-boxes in the second round. This argument can be repeated for the following rounds. Hence, the characteristics through the whole cipher ending with a single active S-box, that have the highest probability, are expected to have low-weight differences, possibly even weight-one differences, in every round. This motivates Assumption 2, which has also been experimentally verified to hold in most cases.

Learning all the probabilities p_e would require encryptions of all 2^{64} possible plaintexts, but we can estimate the probabilities by introducing counters

$$C\big(\{x, y\}\big) = \big|\big\{r_i \mid \exists j : E_K(x\|r_i) \oplus E_K(y\|r_i) = 0^{4j}\|?\|0^{60-4j}\big\}\big|$$

for all pairs $\{x, y\}$, $x, y \in \mathbb{F}_2^4$. Hence, the counter $C(\{x, y\})$ counts how often only one S-box is active in the ciphertext pair when the input pair to S-box S_0 is $\{x, y\}$.

Assumption 1 says that pairs belonging to the same set D_e should also have similar counter values when sufficiently many plaintexts have been encrypted. Assumption 2 says that the highest counter values will (usually) correspond to slender pairs. In the attack we are going to try to identify the slender sets, and this will be relatively easy if the probabilities p_e and $p_{e'}$, $e \neq e'$, are sufficiently different. Experiments show that this condition is often satisfied.

The counter C consists of 120 values since there are $\binom{16}{2} = 120$ different pairs $\{x, y\}$. After encrypting sufficiently many structures, we may sort C in descending order, and thereby hopefully obtain a partitioning of the 120 pairs into a number of sets corresponding to D_e for different values of e. For every $e \neq 0$ it holds that $|D_e| = 8$. We shall return to this partitioning method in a moment. Our final goal will be to learn all four slender sets D_e.

3.2. *Generalising to All S-Boxes and Their Inverses*

In a practical attack, we do not only want to eventually recover the S-box S_0, but all the S-boxes. The above observations can clearly be generalised to all S-boxes by introducing additional types of structures and additional counters.

Moreover, the symmetry between encryption and decryption in the cipher we are considering here means that one may obtain the same type of information about the inverse S-boxes as one obtains about the S-boxes themselves. This can even be done in a chosen-plaintext setting, although it may require more texts than in a chosen-ciphertext setting.

Assume now that we have identified u slender sets for some S-box S, and v slender sets for its inverse S^{-1}. The following table shows the average number of S-boxes that would give rise to the same $u + v$ sets; these averages are based on 100,000 randomly generated S-boxes.

$u\backslash v$	1	2	3	4
1	207	3.52	1.44	1.19
2	3.52	1.16	1.03	1.01
3	1.44	1.03	1.01	1.01
4	1.19	1.01	1.01	1.01

Evidently, if $u + v \geq 6$, then the S-box is usually uniquely determined from the $u + v$ sets, and in many cases, fewer sets are sufficient. However, there exist S-boxes S which are not uniquely determined even if all four slender sets are known for both S and S^{-1}.

On a side note: if D_e and $D_{e'}$ are known for some S-box S, then $D_{e\oplus e'}$ does not give any new information about S, since $D_{e\oplus e'}$ can be derived from D_e and $D_{e'}$. Clearly,

if $\{x, y\} \in D_e$ and $\{x, z\} \in D_{e'}$, then $\{y, z\} \in D_{e \oplus e'}$. This observation generalises to more than two sets. In general, given sets D_{e_i} one can construct all sets D_e where e can be written as a linear combination of the vectors e_i, see Lemma 4 in Appendix A. Therefore, we shall generally only be interested in the four slender sets, since other sets give no additional information about the S-box.

We now describe a number of ways to partition the pairs into sets and to check that this partitioning is correct.

3.3. *Partitioning Pairs into Sets*

Assume again that we are trying to recover S-box S_0. Our starting point for partitioning pairs (in particular, the slender pairs) into sets is the counter C.

The straightforward partitioning method simply sorts C in descending order, and takes the first eight pairs as the first set, the next eight pairs as a second set, etc. Using this method obviously means that we shall often make the wrong partitioning into sets, but the partitioning can be checked using the very strong *filtering methods* described in the following subsection.

3.4. *Filtering Methods*

Given u sets for some S-box S and v sets for its inverse S^{-1}, the most indicative method to check whether these sets may be correct is to see how many S-boxes would give rise to the same sets. If no S-box gives rise to these sets, then clearly the sets must be wrong. We call this filter the *existence filter*. However, counting the number of S-boxes that give rise to these sets is somewhat inefficient (see, however, Sect. 4), and as we have seen, if we only know a few sets, there are usually several S-boxes that give rise to the same sets, and so the probability of a false positive is high in this case.

A much more efficient method is based on the trivial observation that for any valid set D_e, we have that $\{x, y : \{x, y\} \in D_e\} = \mathbb{F}_2^4$. In other words, a valid set "covers" all values in $\mathbb{F}_2^4$. Hence, if we have identified a candidate set D containing two pairs $\{x, y\}$ and $\{x, z\}$, then D cannot be a valid set. Although this method is very simple, it is, in fact, a very strong filter; the probability that eight randomly chosen pairs among the 120 pairs cover all values in $\mathbb{F}_2^4$ is only

$$\prod_{i=1}^{7} \frac{\binom{2i}{2}}{\binom{16}{2} - i} \approx 2^{-18.7},$$

and therefore in practice, many wrong candidate sets are discovered by this method. We call this filter the *cover filter*.

It should be noted that one can prove that the cover filter is not only necessary, but also sufficient; see Appendix A.

The final filtering method that we describe here is based on the observation that if two distinct pairs $\{x_1, y_1\}$ and $\{x_2, y_2\}$ belong to the same set D_e, then $\{x_1, y_2\}$ and $\{x_2, y_1\}$ will also belong to the same set $D_{e'}$ for some $e' \neq e$, and likewise, $\{x_1, x_2\}$ and $\{y_1, y_2\}$ will belong to the same set $D_{e''}$ for some $e'' \notin \{e, e'\}$. To see this, note that if $\{x_1, y_1\}$ and $\{x_2, y_2\}$ belong to the same set D_e, then (by definition) $S(x_1) \oplus S(y_1) = S(x_2) \oplus S(y_2) = e$, and therefore $S(x_1) \oplus S(y_2) = S(x_2) \oplus S(y_1) = e \oplus S(y_1) \oplus$

$S(y_2) \neq e$, etc. Hence, assume that we know two sets D' and D'' (both already known to cover $\mathbb{F}_2^4$), and that $\{a, b\} \in D'$ and $\{a, c\} \in D''$. Now, if $\{c, d\} \in D'$, then for these two sets to both be valid, it must hold that $\{b, d\} \in D''$. We call this filter the *bowtie filter*; if one follows the "partner" b of a in the set D' and jumps to the next set D'' to find the partner d of b there and so forth, then one should come back to the pair $\{a, b\}$ in D' after two jumps back and forth between the two sets, hence forming a bowtie-shaped cycle:

$$D' = \big\{\{a, b\},\ \{c, d\},\ \ldots\big\}$$

$$D' = \big\{\{a, c\},\ \{b, d\},\ \ldots\big\}$$

3.5. *Relaxed Truncated Differentials*

The method considered so far increments a counter only when there is a single active S-box in the ciphertext pair. The probability of this event is relatively low, so many plaintext pairs are needed before it is possible to partition pairs into sets.

It is much more likely that the weight one difference spreads moderately through the cipher resulting in a few active S-boxes in the ciphertext. Hence, we might find slender pair candidates more efficiently by looking at ciphertext pairs with more than one active S-box. The more active S-boxes we allow, the more noise (cases of a few active S-boxes that do not correspond to a weight one difference after the first S-box layer) we will get, and so there is a trade-off between the signal-to-noise ratio, and the strength of the signal.

It turns out that allowing even a relatively large number of active S-boxes does not introduce too much noise. This can be used to make the attack more efficient. For each input S-box S_i and for each pair $\{x, y\}$, we introduce counters $C_{i,j}(\{x, y\})$. We increment the counter $C_{i,j}(\{x, y\})$ every time the input pair $\{x, y\}$ to S-box S_i (with a random but fixed input to the other S-boxes) leads to exactly j S-boxes being active, where j ranges from 1 to 15. Having done a number of encryptions, we may sort the counters $C_{i,j}$ for some pair i, j. If the cover filter identifies sets based on this sorting, we assume that these are correct slender sets. When we have several sets, we use the bowtie filter to check the validity of the sets. We do this for increasing j from 1 to 15. Since the cover filter is a very strong filter, the risk of errors is low, both in the cases where the signal is weak (small values of j), and also in the cases where there is a lot of noise (large values of j).

4. The Attack in Practice

We now describe how the attack is carried out in practice. The attack consists of a data collection phase followed by an S-box recovery phase, and those two phases are repeated until all or almost all S-boxes have been recovered.

4.1. *Data Collection Phase*

In the data collecting phase, we simply encrypt structures and increment counters when applicable. Each structure consists of 16 plaintexts differing in only a single input S-box.

Which S-box is active is a random choice among the S-boxes that have not already been recovered.

After encryption, we check all 120 pairs of ciphertexts to see if any of them are active in less than 16 S-boxes. If so, we increment the corresponding counter for the input pair to the S-box that was active in the plaintext.

We also carry out decryptions in order to obtain information about the inverse S-boxes.

4.2. *S-Box Recovery Phase*

Every once in a while, we stop collecting data and try identifying sets for each S-box. This is done by first sorting the counters for each number of active output S-boxes. We start with the lowest number of active output S-boxes. We check if the top eight counter values in the sorted list pass the cover filter. If so, we consider these eight pairs a slender set and add it to a collection of identified sets, unless the set is already present in the collection. When there are multiple sets in the collection, we check if they pass the bowtie filter. We then look at the next eight pairs and so forth. We stop adding sets when we have identified four sets, or we run into an inconsistency such as a failing bowtie test or non-disjoint sets. In case of an inconsistency, we give up identifying sets for this S-box.

The bowtie filter can also be used to filter out candidate sets that can be derived from existing sets. Consider as an example a situation where the following two candidate sets D_e and $D_{e'}$ (passing the bowtie test) have been identified:

$$D_e = \big\{\{0,1\},\{2,3\},\{4,5\},\{6,7\},\{8,9\},\{a,b\},\{c,d\},\{e,f\}\big\},$$
$$D_{e'} = \big\{\{0,2\},\{1,3\},\{4,6\},\{5,7\},\{8,a\},\{9,b\},\{c,e\},\{d,f\}\big\}.$$

From these two sets we can derive the set $D_{e\oplus e'}$ directly as

$$D_{e\oplus e'} = \big\{\{0,3\},\{1,2\},\{4,7\},\{5,6\},\{8,b\},\{9,a\},\{c,f\},\{d,e\}\big\}.$$

As an example, $S(0)\oplus S(3) = (S(0)\oplus S(1))\oplus(S(1)\oplus S(3)) = e\oplus e'$. Hence, if we identify a set which can be derived from two sets already identified, then we should not add the third set to our collection (assuming that the first two sets are slender, which means the third is not).

We note that if one swaps two "bowtie pairs" in two valid sets (e.g. the pairs $\{0,1\}$ and $\{2,3\}$ could be swapped with $\{0,2\}$ and $\{1,3\}$ in D_e and $D_{e'}$ above), then the resulting sets will still pass both the cover and the bowtie test. This is a potential cause for errors; if two sets have roughly the same probability of causing a single active S-box in the ciphertext, and the distribution of the probabilities for each output S-box is similar for the two sets, then we are likely to generate wrong sets that pass both the cover and the bowtie test. This error may be caught by the existence filter (cf. the following), but if not, then we will be recovering the wrong S-box. This does happen in practice, although rather rarely.

We repeat the above method of identifying sets for the inverse S-boxes as well, maintaining separate counters for these.

Once we have identified as many sets as possible using this method (for both the S-box and its inverse), we can apply the existence filter to check if these sets can possibly be valid; if there is no S-box generating these sets, then the sets are obviously not valid. As mentioned in Sect. 3, applying the existence filter is not terribly efficient; on the other hand, it is not terribly slow either. A reasonably efficient way to implement it is by making guesses for values of $S(0)$ and the exact values e for the identified sets D_e until one runs into an inconsistency with the candidate sets. Note that once these guesses have been made, we may find the "partner" of 0 in all candidate sets. For instance, if the two sets D_e and $D_{e'}$ in the example above are our candidate sets, and we guess that $S(0) = 0$, then we would know that $S(1) = 2^i$ and $S(2) = 2^j$ for some (guessed) i, j, $i \neq j$ and $0 \leq i, j < 4$. We would obtain similar information about the inverse S-box from the candidate sets for the inverse S-box. This method is able to find all candidate S-boxes in a fraction of a second given at least one set for the S-box and one set for its inverse.

If an S-box has been recovered, we stop considering this S-box both in the data collection and the S-box recovery phase. If not all S-boxes have been recovered, we continue the data collection phase. In some cases, we have to give up recovering one or more S-boxes because we are unable to identify sufficiently many sets, or because we consistently get no candidates for the S-box based on the identified sets. In the latter case, there is obviously an error in the partitioning into sets. If we consistently obtain multiple candidates for an S-box, we may also accept this and consider the S-box recovered, keeping a record of all candidates.

4.3. *Example Data*

In order to better illustrate how the attack is carried out, we give some example data for an attack on a 10-round cipher of the type of Algorithm 1 with a randomly chosen key. After encrypting 2^{17} structures of 16 plaintexts, the plaintexts in each structure varying only in S-box 0, we obtain the following counter values with *one* active S-box in the ciphertext (the first 32 values after sorting by counter value, separated into four sets of eight pairs, are shown):

First set			Second set			Third set			Fourth set		
{0,e}	33	(8)	{0,8}	15	(4)	{6,d}	10	(4)	{1,2}	1	(9)
{2,a}	32	(8)	{3,f}	14	(4)	{4,f}	10	(1)	{7,e}	1	(9)
{5,6}	31	(8)	{2,4}	13	(4)	{0,7}	9	(1)	{0,d}	1	(9)
{b,f}	28	(8)	{9,e}	12	(4)	{1,a}	9	(1)	{7,9}	0	(d)
{4,c}	26	(8)	{2,3}	12	(1)	{1,b}	8	(4)	{7,8}	0	(5)
{8,9}	24	(8)	{5,7}	12	(4)	{d,e}	7	(1)	{6,f}	0	(f)
{1,3}	23	(8)	{b,c}	11	(1)	{6,9}	6	(1)	{7,a}	0	(e)
{7,d}	22	(8)	{a,c}	10	(4)	{5,8}	6	(1)	{6,e}	0	(5)

The pair in braces represents the input pair $\{x, y\}$ to S-box 0. The second number is the counter value in decimal, and the number in brackets is the output difference of S-box 0 (which is, of course, unknown to the attacker).

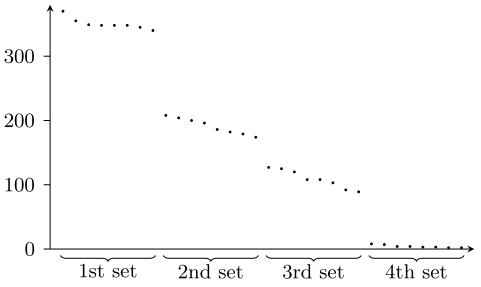
We see that the first set of eight pairs is a valid slender set. The second and third sets have very similar counter values, and correspond to an incorrect separation of the

union of two valid slender sets. This would be discovered by the cover filter since, e.g. the value 2 occurs twice in the second set. The fourth set contains more or less random pairs since the counter values are 0 or 1.

The same cipher with the same key and the same number of encryptions yields the following values of the counter being incremented when up to *five* S-boxes are active in the ciphertext:

First set			Second set			Third set			Fourth set		
{b,f}	370	(8)	{9,e}	208	(4)	{2,3}	127	(1)	{3,a}	8	(9)
{5,6}	355	(8)	{6,d}	204	(4)	{4,f}	125	(1)	{1,2}	7	(9)
{8,9}	349	(8)	{1,b}	200	(4)	{b,c}	120	(1)	{5,9}	4	(9)
{1,3}	348	(8)	{2,4}	196	(4)	{6,9}	108	(1)	{4,b}	4	(9)
{4,c}	348	(8)	{3,f}	186	(4)	{1,a}	108	(1)	{0,d}	3	(9)
{2,a}	348	(8)	{a,c}	182	(4)	{0,7}	103	(1)	{7,e}	3	(9)
{0,e}	345	(8)	{0,8}	179	(4)	{5,8}	92	(1)	{3,4}	2	(5)
{7,d}	340	(8)	{5,7}	174	(4)	{d,e}	89	(1)	{6,a}	2	(2)

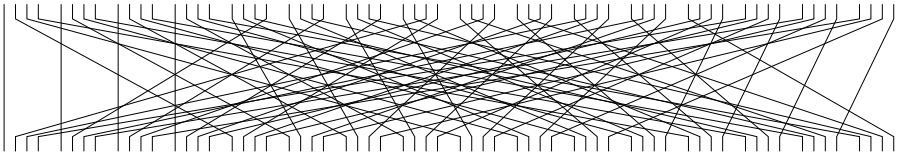
We see that the first three sets are valid slender sets. The fact that the fourth set is not would be discovered by the cover filter. This shows that accepting more noise (by allowing multiple active output S-boxes) can, in fact, lead to more information. We also see that the probabilities of up to five active S-boxes in the ciphertext are apparently very distinct for the three slender sets, but similar *within* the sets. The following plot shows how the counter values make a relatively big jump from one set to the next.



5. Case Study: the Block Cipher Maya

Maya is a block cipher proposed at WCC 2009 [10]. It is a PRESENT-like cipher with key dependent S-boxes (repeated in every round) and a fixed, known bit permutation (see Fig. 2). Each round also contains an addition of a round key. The round keys and the S-boxes are derived from the 1024-bit master key.

Since the S-boxes are the same in every round, using the differential-style attack described above, we are able to get information on the S-boxes and their inverses. We get information on both directions for every encrypted pair and can choose to also do decryptions to obtain information about the inverse of a specific S-box. In this way,



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	16	32	48	4	20	36	52	8	24	40	56	12	28	44	60	21	37	53	5	25	41	57	9	29	45	61	13	33	49	1	17

32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
42	58	10	26	46	62	14	30	50	2	18	34	54	6	22	38	63	15	31	47	3	19	35	51	7	23	39	55	11	27	43	59

Fig. 2. The Maya bit permutation.

we often recover at least two sets in each direction, which usually means all the S-boxes can be determined uniquely. The key addition, however, means that we only obtain the correct S-boxes up to an xor by the last round key, which is unknown. However, this still enables us to peel off the first and the last rounds of encryption, after which the attack can be repeated on this reduced cipher. Moreover, we expect that once the S-boxes are known, a dedicated differential or linear attack is more efficient than our general attack. In the end, we obtain a description of an equivalent cipher.

The standard number of rounds in Maya is 16. Below, the (base 2) logarithm of the complexities to recover the secret S-boxes for a number of different randomly chosen example keys is given. Complexities in *italics* are extrapolated values from running the attack on fewer rounds.

Case	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Complexity	*46.8*	36.0	35.9	36.9	35.7	*39.3*	37.4	37.1	*41.2*	38.5	*39.4*	*39.5*	36.0	36.7	40.5	37.4

Moreover, Table B.1 (Appendix B) shows the log of the complexity (number of texts) as a function of the number of rounds for the same example keys. See Fig. 3 for a graphical representation. The complexities refer to obtaining all 16 S-boxes (whenever possible, see discussion below), so that the first and the last round can be peeled off, and the cipher with two rounds less can then be attacked.

In this implementation of the attack, an S-box was considered correctly recovered if only one S-box gave rise to the given partitioning into sets. However, if a substantial amount of time had been spent on an S-box, the conditions were relaxed such that even if there were more than one candidate S-box, work on this S-box was still discontinued and all candidates were printed. In extreme cases, where there were no candidate S-boxes after a lot of time had been spent trying to recover the S-box, that S-box was given up. The choice of when to accept multiple candidates, or when to give up an S-box, obviously affects the complexity of the attack. A more sophisticated implementation might adapt better to these situations. As an example, if the program consistently gives rise to the same partitioning into sets, and there are no candidates for this partitioning, one might try swapping elements between sets in such a way that the bowtie condition still holds.

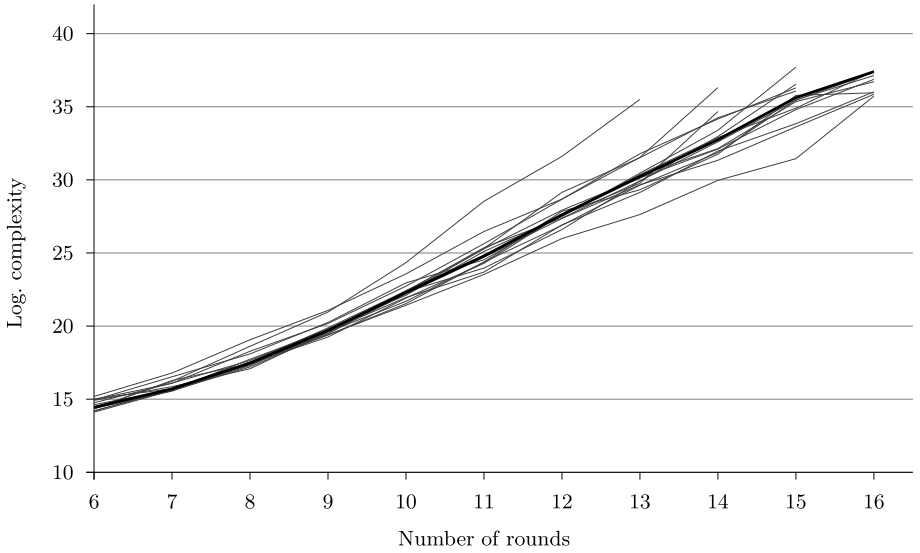


Fig. 3. A graphical representation of the data in Table B.1. The *thick line* represents the median computed for each number of rounds.

The error rate of the attack is very low. If we consider the highest number of rounds broken in each of the 16 test cases, then the total number of S-boxes that had to be recovered was $16 \cdot 16 = 256$. Of these, 245 were correctly recovered with only a single S-box candidate. For seven S-boxes, there were multiple candidates, and the correct S-box was always one of these. The number of candidates ranged from two to four. Three out of 256 S-boxes were incorrectly recovered with only a single S-box candidate. One S-box was given up due to too much time spent trying to recover it.

In a real attack, the fact that some S-boxes were incorrectly recovered would be discovered after attempting to break the cipher reduced by the first and the last rounds. By making sure that a large amount of information about the identified sets and the counter values is recorded, it is likely that one would be able to locate the S-box causing the problem. For instance, there may be 16 counters that are all similar, meaning that it is likely that two sets have been mixed up.

6. Model for the Complexity of Recovering Sets D_e

For a small number of rounds, the attack has a low complexity, which makes it possible to obtain a large amount of experimental data. However, in order to be able to extrapolate the attack complexity, we describe a theoretical model below.

We again focus on recovering a single S-box, e.g. S_0. In the attack, we are faced with the problem of separating 120 counters $C(\{x, y\})$, each belonging to an input pair to an S-box of the first round, into 15 distinct sets. All pairs within a set should yield the same output difference, i.e. belong to a set D_e for some e.

Interpreting the counters $C(\{x, y\})$ as random variables, a counter $C(\{x, y\})$, with $S(x) \oplus S(y) = e$ is binomially distributed with parameters n and p_e. Here, p_e is the probability that the difference $(e\|0^{60})$ after the first layer of S-boxes yields only one active S-box in the output, and n is the number of text pairs.

Assumption 2 states that counters $C(\{x, y\})$ such that $S(x) \oplus S(y)$ has a weight greater than one are significantly smaller than others, and we therefore focus on the 32 counters corresponding to slender pairs. Thus, we consider 8 counters distributed with parameters (n, p_1), 8 distributed with parameters (n, p_2), 8 distributed with parameters (n, p_4), and 8 distributed with parameters (n, p_8). Without loss of generality, we assume $p_1 \geq p_2 \geq p_4 \geq p_8$, and furthermore that $p_1 \neq p_2$. The attack works by looking at the 8 highest counters, and it will be successful if those counters correspond to the same output difference, e.g. $e = 1$, of the S-box. The attack fails whenever there exists a pair $\{x_1, y_1\}$ such that $S(x_1) \oplus S(y_1) = 1$, and another pair $\{x_2, y_2\}$ with $S(x_2) \oplus S(y_2) \neq 1$, such that $C(\{x_1, y_1\}) \leq C(\{x_2, y_2\})$. In the following, we estimate this failure probability (denoted err) depending on the number of samples n.

To simplify the problem for now, we consider only two pairs $\{x_1, y_1\}$ and $\{x_2, y_2\}$ and their corresponding counters, where $C(\{x_1, y_1\})$ is distributed with parameters (n, q), and $C(\{x_2, y_2\})$ is distributed with parameters (n, p) for $q > p$. The attack fails if $C(\{x_1, y_1\}) \leq C(\{x_2, y_2\})$, and thus we denote $Z = C(\{x_2, y_2\}) - C(\{x_1, y_1\})$ and

$$\text{err} = \Pr\big(C\big(\{x_1, y_1\}\big) \leq C\big(\{x_2, y_2\}\big)\big) = \Pr(Z \geq 0).$$

To investigate this error further, consider the usual approximation of the binomial distribution by the normal distribution, $C(\{x_1, y_1\}) \sim N(nq, nq(1-q))$ and $C(\{x_2, y_2\}) \sim N(np, np(1-p))$. With this approximation, the distribution of Z can be approximated by $Z \sim N(\mu, \sigma^2)$, where $\mu = n(p-q)$ and $\sigma = n(p(1-p) + q(1-q))$.

The density function for the normal distribution with mean μ and variance σ^2 is given by the formula $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. The integral of the normal density function is the normal distribution function

$$N(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{1}{2}x^2}\, dx.$$

The error we make is thus described by

$$\text{err} \approx 1 - \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{0} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{-\mu}{\sigma}} e^{-\frac{x^2}{2}} = 1 - N\left(\frac{-\mu}{\sigma}\right).$$

The following lemma gives an estimate of the 'tail' $1 - N(x)$ which is useful to approximate the error.

Lemma 1 (Feller [8]). *Let $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ be the normal distribution. As $x \to \infty$*

$$1 - N(x) \approx x^{-1}\phi(x).$$

Using the approximation of Lemma 1 yields

$$\text{err} \approx 1 - N\left(-\frac{\mu}{\sigma}\right) \approx -\frac{\sigma}{\mu}\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\mu}{\sigma})^2}. \tag{1}$$

From (1) it follows that for a given failure probability err, the sample must be of size

$$n > \frac{-c(p^2 - p + q^2 - q)}{(p-q)^2}, \tag{2}$$

where $c = \text{LambertW}(\frac{1}{2\,\text{err}^2\,\pi})$ [7] is a small constant depending on the error. As an example, we can assume that $q = 2p$. Then, in order for the attack to be successful, we need a sample of size roughly $\frac{3c}{p}$.

After having estimated the failure probability for 2 counters, assuming independence, the total error probability err_t, that is, the probability of the event that one of the 8 counters with parameter (n, p_1) being smaller than one of the 24 counters with parameters (n, p_2), (n, p_4), (n, p_8) can be bounded as

$$\text{err}_t \leq 1 - (1 - \text{err})^{8 \cdot 24}.$$

If we allow an error probability of $\text{err}_t \leq 0.5$, which in light of the strong cover filter is clearly sufficient, we need $\text{err} \leq 1 - 0.5^{1/(8 \cdot 24)} \approx 0.0036$. For this, $c = 8$ is sufficient.

The next step is to find a way to estimate the probabilities p_e. Assuming that the cipher is a Markov cipher, we can model the propagation of differences through the cipher as a matrix multiplication of the difference distribution matrices and the permutation matrices. Considering the difference distribution table for the whole layer of S-boxes would yield a $2^{64} \times 2^{64}$ matrix. As this matrix is far too big to be of any use, we determine the difference distribution matrix which contains the probabilities for 1-bit to 1-bit differences only. A comparison with experimental data shows that this is a good approximation. This matrix is of size only 64×64. This enables us to simulate the propagation of 1-bit to 1-bit differences through a number of rounds using matrix multiplications. For the resulting matrix, an entry (i, j) contains the probability that given the single, active input bit i after the first layer of S-boxes, a single output bit j in the second last round will be active. This matrix can thus be used to get an estimate for the parameters of the counters. We determine the probability that given a fixed 1-bit difference after the first round, exactly one S-box is active in the last round (analogously for the inverse). This can be done by summing over the corresponding matrix entries. Then we use the inequality (2) to calculate the number of plaintexts needed to recover at least two sets D_e in each direction. Note that in the original attack, we do not restrict ourselves to having a single active S-box in the last round, but a limited number of active S-boxes. Furthermore, we can expect that on average a single active S-box will not lead to 16 active S-boxes after two rounds of encryption. Thus, we believe that in practice we can break at least two more rounds of encryption with the sample size determined by the model, meaning the model yields an upper bound for the complexity. The comparison between the experimental data and the modeled data supports this assumption.

To justify the introduced model, we implemented the attack for a small number of rounds (see Sect. 5). For each number of rounds, we sampled 1000 ciphers in our

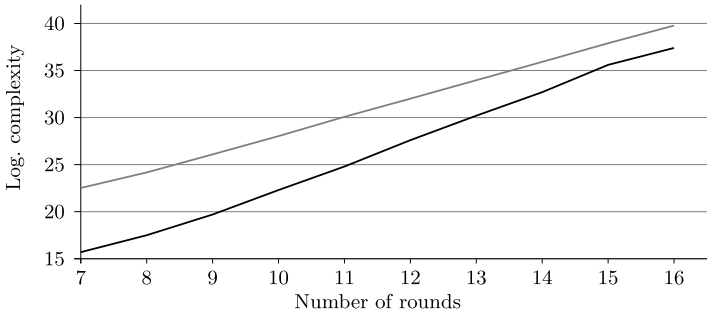


Fig. 4. Comparison between the medians of the experimental data and the model for recovering two sets D_e in each direction. The *black line* shows the experimental data while the *gray line* shows the data from the model. The complexity unit is one plaintext.

model to determine the sample size needed to distinguish between the two distributions. Figure 4 gives a comparison of the experimental data with that of the model for the case that we want to recover at least two sets D_e for all 16 S-boxes. The black line shows the experimental data, and the gray line shows the model for an error of around 0.3%, which corresponds to $c = 8$. The complexity denotes the logarithm of the number of plaintexts used. As seen, the model seems to give an upper bound on the complexity of the attack. In some rare cases, the difference between p and q is almost zero, which leads to a very high attack complexity. These rare cases have a strong influence on the average complexity, hence we considered the median instead of the mean to estimate the complexity of the attack.

The model suggests that we are able to break up to 28 rounds before we reach the bound of 2^{64} available plaintexts.

7. Linear Cryptanalysis

In the differential-style attack, one hypothesis is that the probability of a characteristic with a single-bit difference at the output of the S-box layer in the first round is correlated with the probability of a single-bit difference at the input to the S-box layer in the last round, or with the number of active S-boxes in the last round. Using a similar hypothesis for linear characteristics, one can mount a linear attack to extract information about the secret S-boxes. In the differential-style attack, one tries to identify sets of eight pairs of values related to a certain differential. In a linear-style attack, one tries to identify sets of eight values related to a certain linear characteristic. In the following, we outline this approach.

Before we describe the attack itself, we need to fix some notation. Given a function $H : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the bias ϵ of the linear approximation

$$\langle \beta, H(x) \rangle + \langle \alpha, x \rangle$$

for an m-bit output mask β and an n-bit input mask α is defined by

$$\Pr\big(\langle \beta, H(x) \rangle + \langle \alpha, x \rangle = 0\big) = \frac{1}{2} + \epsilon,$$

where the probability is taken over all choices of inputs x. A related measure for this bias is the Walsh- or Fourier-transform, defined as

$$\widehat{H}(\alpha, \beta) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \beta, H(x) \rangle + \langle \alpha, x \rangle}.$$

The fundamental relation between the Fourier transform of H and the bias of the linear approximation is given by

$$\epsilon = \frac{\widehat{H}(\alpha, \beta)}{2^{n+1}}, \tag{3}$$

and thus studying the bias and studying the Fourier transform are, up to scaling, equivalent. However, in our case, studying the Fourier transform has the advantage of allowing simpler proofs of relationships between the bias of several equations.

Without loss of generality, we focus on the leftmost S-box, denoted simply by S. All other S-boxes can be processed similarly. Denote by

$$\begin{aligned} F\colon \quad & \mathbb{F}_2^4 \times \mathbb{F}_2^{60} \to \mathbb{F}_2^{64} \\ & F(x, y) = c \end{aligned}$$

the encryption function that starts after the first layer of S-boxes. Here, we split the input into two parts; x is the output of the leftmost S-box S of the first S-box layer, and y is the concatenation of the outputs of the remaining S-boxes of the first S-box layer.

We assume that F is vulnerable to a linear attack. That is, there exists an input mask $\alpha = (\alpha_1, \alpha_2)$ and an output mask β such that the value of

$$\big\langle \beta, F(x, y) \big\rangle + \langle \alpha_1, x \rangle + \langle \alpha_2, y \rangle$$

is biased. In terms of the Fourier transform of F, this means that the value

$$\widehat{F}(\alpha, \beta) = \widehat{F}\big((\alpha_1, \alpha_2), \beta\big) = \sum_{x \in \mathbb{F}_2^4} \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, F(x,y) \rangle + \langle \alpha_1, x \rangle + \langle \alpha_2, y \rangle}$$

is non-zero.

As we explain below, we will make use of the bias of the value $\langle \beta, F(x, y) \rangle$ for fixed values of x. We denote the corresponding function by T_x. That is,

$$\begin{aligned} T_x\colon \quad & \mathbb{F}_2^{60} \to \mathbb{F}_2^{64} \\ & T_x(y) = F(x, y), \end{aligned}$$

and we look at

$$\widehat{T}_x(0, \beta) = \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, T_x(y) \rangle} = \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, F(x,y) \rangle}.$$

The first observation, linking the bias of T_x to the bias of F, is the following lemma:

Lemma 2. *With the notation from above, it holds that*

$$2^4 \widehat{T}_\lambda(0,\beta) = \sum_{\alpha_1 \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, \lambda \rangle} \widehat{F}\big((\alpha_1, 0), \beta\big).$$

Proof. We compute

$$\begin{aligned}
\sum_{\alpha_1 \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, \lambda \rangle} \widehat{F}\big((\alpha_1, 0), \beta\big) &= \sum_{\alpha_1 \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, \lambda \rangle} \sum_{x \in \mathbb{F}_2^4} \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, F(x,y) \rangle + \langle \alpha_1, x \rangle} \\
&= \sum_{\alpha_1 \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, \lambda \rangle} \sum_{x \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, x \rangle} \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, F(x,y) \rangle} \\
&= \sum_{\alpha_1 \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, \lambda \rangle} \sum_{x \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, x \rangle} \widehat{T}_x(0,\beta) \\
&= \sum_{x \in \mathbb{F}_2^4} \widehat{T}_x(0,\beta) \sum_{\alpha_1 \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, \lambda + x \rangle} \\
&= 2^4 \widehat{T}_\lambda(0,\beta),
\end{aligned}$$

where the last equality comes from the fact that

$$\sum_{\alpha_1 \in \mathbb{F}_2^4} (-1)^{\langle \alpha_1, \lambda + x \rangle} = \begin{cases} 0 & \text{if } \lambda \neq x, \\ 2^4 & \text{if } \lambda = x. \end{cases}$$

□

The attack now proceeds as follows. Denote the (whole) encryption function by E. We split its input as before and consider

$$\begin{aligned} E\colon \quad & \mathbb{F}_2^4 \times \mathbb{F}_2^{60} \to \mathbb{F}_2^{64} \\ & E(x, y) = c, \end{aligned}$$

where x is now the input to the first S-box, and y is the input to all other S-boxes. We define the function corresponding to fixing x as T'_x, that is,

$$\begin{aligned} T'_x\colon \quad & \mathbb{F}_2^{60} \to \mathbb{F}_2^{64} \\ & T'_x(y) = E(x, y). \end{aligned}$$

For a selection of masks β and each possible value of x, we estimate the value of $\widehat{T}'_x(0,\beta)$. This is simply done by encrypting a number of known plaintexts with the first four bits fixed to x and counting how often the value of

$$\big\langle \beta, T'_x(y) \big\rangle = \big\langle \beta, E(x, y) \big\rangle$$

is zero (resp., one). The next lemma is the key for deducing information about the secret S-box S.

Lemma 3. *With the notation from above, the bias of* $\langle \beta, T'_x(y) \rangle$ *is equal to the bias of* $\langle \beta, T_{S(x)}(y) \rangle$. *That is,*

$$\widehat{T}'_x(0, \beta) = \widehat{T}_{S(x)}(0, \beta).$$

Proof. It holds that $E(x, y) = F(S(x), \bar{S}(y))$ where $\bar{S}$ denotes the application of the remaining 15 S-boxes to y. Therefore,

$$\begin{aligned}
\widehat{T}'_x(0, \beta) &= \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, T'_x(y) \rangle} \\
&= \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, E(x,y) \rangle} \\
&= \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, F(S(x), \bar{S}(y)) \rangle} \\
&= \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, F(S(x), y) \rangle} \\
&= \sum_{y \in \mathbb{F}_2^{60}} (-1)^{\langle \beta, T_{S(x)}(y) \rangle} \\
&= \widehat{T}_{S(x)}(0, \beta)
\end{aligned}$$

as claimed. □

Note that we do not know the leftmost S-box S, but assume that, for a given mask β, there is exactly one mask a such that $\widehat{F}((a, 0), \beta)$ is high (in absolute terms) while for any $b \neq a$ the value $\widehat{F}((b, 0), \beta)$ is (relatively) close to zero. In this case, we get

$$\widehat{T}'_x(0, \beta) = \widehat{T}_{S(x)}(0, \beta) \approx 2^{-4} (-1)^{\langle a, S(x) \rangle} \widehat{F}\big((a, 0), \beta\big).$$

In this way, we can partition the values of x into two equally-sized sets, depending on the sign of $\widehat{T}'_x(0, \beta)$. This will finally give us two sets V_0 and V_1 with

$$V_\gamma = \big\{ x \mid \langle a, S(x) \rangle = \gamma \big\}.$$

While we still do not know a or γ, this is substantial information about the secret S-box. In particular, analogously to the differential-style attack, it is reasonable to assume that a is of weight one, since the (non-trivial) linear approximations of the whole cipher that have the largest bias most likely contain only one active S-box in most of the rounds. For the same reason, it also seems most promising to consider output masks, that is, β values, with only one active S-box. In other words, we consider only output masks β of the form $0^{4j} \| ? \| 0^{60-4j}$ for values j between 0 and 15.

To be more specific about how the attack may be carried out, assume that we have a set of counters $\widehat{T}'_x(0, \beta)$ for some output S-box. We may consider the 15 vectors $W_\beta = (\widehat{T}'_0(0, \beta), \ldots, \widehat{T}'_f(0, \beta))$, for some output masks $\beta \in \mathbb{F}_2^{64} \setminus \{0\}$ of the above described

form. We identify the (say) three longest such vectors using the Euclidean norm as a metric. The motivation for focusing on these vectors is that we assume these contain the most reliable information. We transform each of these vectors into a binary vector such that the eight highest counter values correspond to '1'-bits and the remaining correspond to '0'-bits. We then take a majority vote among the three vectors, and if this results in exactly eight '1'-bits and eight '0'-bits, and if the vectors agree in most positions, then we partition the different values of x according to this majority vote such that values of x corresponding to '1'-bits are in one set, and the remaining values are in the other set.

As an example, we carried out the attack using 2^{25} known plaintexts on a cipher with 10 rounds. For the first output S-box, we obtained counter values $\widehat{T}'_x(0, \beta)$ such that the three longest vectors were these:

$(-2033, -1359, -4834, 4915, -1041, 3063, 117, -2280, 4325, 3265, 769, 769, 3097, -3677, 199, -633, 3445),$

$(-1652, -935, -2953, 3464, -873, 1322, -194, -2732, 2047, 2492, 2945, 2264, -2970, -601, -550, 1467),$

$(-1681, -838, -1152, 2647, -151, 3376, 23, -853, 1122, 1851, 362, 2473, -2084, 166, -797, 2712).$

After transforming these vectors into binary vectors as described, one gets

$$(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1),$$
$$(0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1),$$
$$(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1).$$

Majority voting says that the "true" vector is $(0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1)$, so the values of x should be partitioned into the sets $\{0, 1, 2, 4, 6, 7, \mathrm{c}, \mathrm{e}\}$ and $\{3, 5, 8, 9, \mathrm{a}, \mathrm{b}, \mathrm{d}, \mathrm{f}\}$. Knowing the S-box, which can be compactly written c8b096ae2351d4f7, one can confirm that the first set is $\{x \mid \langle 8, S(x)\rangle = 0\}$ and the second is $\{x \mid \langle 8, S(x)\rangle = 1\}$.

These counters were specific to the first output S-box; the other 15 S-boxes give rise to tables of counters that may further confirm this partitioning, or they may reveal one or more of the sets $\{x \mid \langle 2^i, S(x)\rangle = 0\}$, $0 \le i < 3$. Clearly, intersections of two correct such sets (with different masks) contain exactly four values, and so this is a way to check correctness.

Compared to the differential-style attack, this attack has some drawbacks. First of all, in the differential style-attack, making use of relaxed truncated differentials resulted in a considerable speed-up, especially for a small number of rounds. Secondly, the different possibilities to filter wrong sets in the differential-style attack allowed us to define very useful criteria for when to stop collecting data, and reduced the probability of choosing a wrong S-box substantially.

We leave it as a possible direction of future research to overcome the drawbacks in the linear attack.

8. Fully Random PRESENT-like Ciphers

In this section, we consider PRESENT-like ciphers where the S-boxes and the bit permutations of all rounds are chosen independently and uniformly at random, that is, ciphers given by Algorithm 2.

8.1. *Recovering the S-Boxes*

For a fully random cipher, one would not get information about the inverse S-boxes like in the case of Maya. Moreover, the S-boxes are not uniquely determined, cf. Appendix A. One needs to recover all four slender sets D_e for each S-box. We implemented a series of attacks on such ciphers and the results show that recovering four sets is indeed possible, but not for all S-boxes. The following table shows the results of our tests to fully recover one S-box in the first round. The complexity is the number of chosen plaintexts needed and is given as the median of 500 tests.

Rounds	Complexity	Probability
4	$2^{12.5}$	73%
5	$2^{15.5}$	82%
8	$2^{24.5}$	81%

In each test, the computation was stopped if not all four slender sets were obtained with 2^{30} structures. The tests are very time-consuming, which is why results for 6 and 7 rounds were not implemented.

Summing up, the attack does not seem to be able to fully recover all S-boxes of the first (or last) round, merely about 80%. However, in the remaining cases, the attack identifies one, two or three sets S_e, which means that only a limited number of choices for these S-boxes remains. Depending on exactly how many choices of the S-boxes are left, one possible way to proceed is to simply make a guess, and repeat the attack on a reduced number of rounds. If S-boxes in other rounds cannot be successfully recovered, the guess might have been wrong. This is a topic for further research.

8.2. *Recovering the Bit Permutations*

Once the first S-box layer has been recovered, one can start recovering the first bit permutation layer. Here we outline the technique.

The idea is similar to the method of recovering S-boxes; one encrypts plaintext pairs differing in (e.g.) two bit positions. Whenever the output difference is small (e.g. one active S-box), one increments a counter specific to the pair of active bit positions in the plaintext. The two active bit positions are then changed, and the process is repeated. After many such encryptions, one may assume that the highest counter values correspond to pairs of bit positions that are mapped to the same S-box input in the second round.

This leads to information about which bit positions are mapped to the same S-box input in the next round. One can also vary three or four bit positions in order to obtain more information. The complexity of this method has not been thoroughly investigated, but preliminary results indicate that it is similar to the complexity of recovering S-boxes.

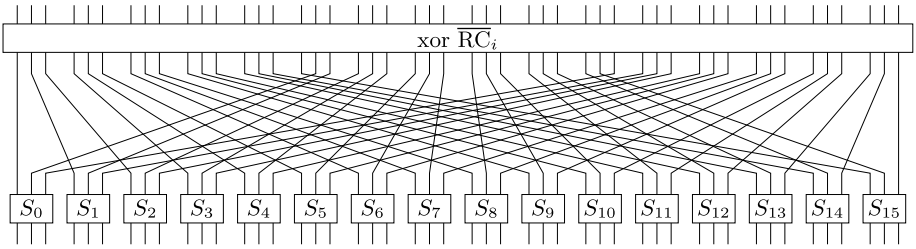


Fig. 5. Alternative representation of one round of PRINTCIPHER-48.

9. Application to PRINTCIPHER

PRINTCIPHER [12] is a lightweight block cipher similar to PRESENT, where the S-box size is 3 bits, and each S-box is preceded by a key-dependent bit permutation of the three input bits. There are two variants; PRINTCIPHER-48 has a 48-bit block size and applies 16 S-boxes in each round (there are 48 rounds in total), and PRINTCIPHER-96 has a 96-bit block size and applies 32 S-boxes in each round (there are 96 rounds in total). The key sizes are 80 bits and 160 bits, respectively; the first $\frac{3}{5}$ of the bits are used as round keys (which are the same in every round), and the remaining bits are used to generate the key-dependent bit permutations, where two bits are used for each S-box. Additionally, there is a round counter RC_i, which has size six bits for PRINTCIPHER-48, and seven bits for PRINTCIPHER-96, and which is xored to the least significant bits of the state after the fixed bit permutation. Except for this round counter, every round is the same.

The same S-box, which can be compactly written as 01367452, is used everywhere. This S-box is derived from the inversion operation over $\mathbb{F}_{2^3}$ defined by the polynomial $\omega^3 + \omega + 1$, and being almost perfect non-linear, the probability of every non-trivial differential is either 0 or 1/4. There are three 1-bit to 1-bit differentials with positive probability, namely $2^i \to 2^i$ for $i \in \{0, 1, 2\}$.

One can view the S-box as being key-dependent by combining it with the round key addition and the key-dependent bit permutation. In this case, the 16 or 32 S-boxes are chosen from a set of 32 S-boxes. These all have the same differential properties as the S-box 01367452, except that actual values can differ. One can also move the round constant addition up so that it takes place before the bit permutation (by applying the inverse bit permutation to RC_i, yielding $\overline{\mathrm{RC}_i}$). This results in a cipher that can be visualised as in Fig. 5 (for PRINTCIPHER-48), where each S-box S_i is chosen from the set of 32 S-boxes.

The following features of PRINTCIPHER distinguish it from a pure instance of Algorithm 1:

- the S-box size is three bits;
- the S-boxes are not chosen from the set of all 3-bit S-boxes, but from a set of only 32 (closely related) S-boxes;
- the "round keys" $\overline{\mathrm{RC}_i}$ are public;
- the order of the transformations is reversed.

These differences have some implications. First of all, the smaller S-box size means that the cover filter, which is an extremely useful and efficient filter in the case of ciphers with four-bit S-boxes, is somewhat less effective when the S-box size is three bits; there are 28 pairs of distinct values from $\mathbb{F}_2^3$, and if one randomly chooses four pairs among these 28, then the probability that every value appears exactly once is about $2^{-7.6}$, which is a lot higher than the probability of $2^{-18.7}$ in the case of four-bit S-boxes. However, the probability is small enough that it rarely happens by chance.

The second difference in the list above has more severe implications. For the particular S-box used in PRINTCIPHER, the 32 derived S-boxes can be separated into eight equivalence classes, each of four S-boxes, so that all four S-boxes in an equivalence class have exactly the same slender sets. Moreover, the equivalence class is uniquely determined by a single slender set. Hence, given any positive number of slender sets, one can only determine the S-box up to this equivalence, meaning that three (out of five) bits of key material is revealed. The separation of the 32 S-boxes into equivalence classes is given below.

01367452	10634725	36015274	63102547	74520136	47251063	52743601	25476310
10476325	01743652	47102563	74015236	63251047	36520174	25634710	52367401
75420316	57243061	42751603	24576130	03167542	30615724	16034275	61302457
26415370	62143507	41267053	14620735	53702641	35076214	70534126	07351462

It can be added that if the original S-box had instead been, e.g. 01345672 (cubing over $\mathbb{F}_{2^3}$), which has properties very similar to the actual original S-box, then knowing three slender sets for some S-box would reveal all five bits of key material for that S-box.

Another implication of the second difference in the list above is that the slender sets for the inverse S-boxes are the same for all 32 S-boxes. Hence, sets obtained through decryptions seem to give no additional information.

The fact that all "round keys" $\overline{\mathrm{RC}}_i$ are known means that the attacker has recovered the key once all S-boxes have been determined. The reversing of the order of the transformations within each round has no significance to the attack.

To summarise, given at least one slender set for each S-box, one is left with the task of finding 32 bits of key material for PRINTCIPHER-48, and 64 bits of key material for PRINTCIPHER-96. The remaining question now is: What is the complexity of finding a slender set for each S-box for a variant of PRINTCIPHER with r rounds? A reasonable attack on PRINTCIPHER-48 uses significantly fewer than 2^{48} encryptions, since with this many encryptions, the attacker will learn the whole codebook.

A few experiments on 8 to 16 rounds of PRINTCIPHER-48, where at least one slender set for a single S-box is identified, give the following medians of complexities (log number of plaintexts encrypted):

Rounds	8	9	10	11	12	13	14	15	16
Complexity	15.2	18.3	19.3	21.8	23.7	25.5	27.1	29.7	31.2

This is preliminary work, and it is not a straightforward application of the attack implemented on Maya due to the differences mentioned above, so some improvements are

likely to be possible. Still, extrapolating these results suggests that at most 24 rounds can be broken with 2^{48} plaintexts. This would be (almost) equivalent to a more standard differential attack and resembles some of the attacks presented in Abdelraheem et al. [1]. We expect results to be similar to a differential attack also for PRINTCIPHER-96. It should be added that the *invariant subspace attack* [13] breaks all rounds of PRINTCIPHER for a relatively large fraction of the keys.

10. Conclusion

In this paper, a novel differential-style attack was presented and applied to 64-bit PRESENT-like ciphers with secret components. A variant with 16 secret S-boxes can be attacked for up to 28 rounds with a data complexity of less than 2^{64}. Our attacks exploit the fact that a randomly chosen four-bit S-box with high probability has weak differential properties, and the attacks do not apply to ciphers such as PRESENT, where the S-box is publicly known, but chosen carefully. The best known attack on PRESENT, a linear attack, can be used to cryptanalyse up to 26 rounds of PRESENT. Hence, for PRESENT-like designs, keeping S-boxes secret but choosing them at random seems to be less secure than choosing a single, publicly known, good S-box.

One might protect against our attacks by choosing S-boxes from a smaller set of strong S-boxes (w.r.t. differential cryptanalysis). One cipher that is designed along those lines is PRINTCIPHER, a PRESENT-like cipher where each key-dependent S-box is chosen from a set of 32 strong S-boxes. We showed that our attack is apparently no more effective against PRINTCIPHER, than a more standard differential attack.

We also described how the variant where the S-boxes and bit permutations are chosen at random for every round can be attacked. Attacks were implemented for such ciphers with up to eight rounds. More work is needed to estimate the complexity of such attacks for more than eight rounds.

Furthermore, we sketched a linear-style attack that can be applied to deduce linear information about the S-boxes. However, compared to the differential-style attack, this attack has some drawbacks. It is left as a direction for future research to further investigate linear attacks in this context and to overcome the drawbacks.

Acknowledgements

Julia Borghoff was supported by a grant from the Danish research council for Technology and Production Sciences, grant number 10-093667. Søren S. Thomsen was supported by a grant from the Villum Kann Rasmussen Foundation.

Appendix A. What We Learn About the S-Boxes from the Sets

In this section, we discuss in detail how much we actually learned about an S-box after recovering one or more sets D_e. Here we focus on sets for the S-box itself and not on sets for its inverse. Before doing so, we remark that it is not possible to recover the S-boxes uniquely when no set for the inverse S-box is given. In particular, when two

S-boxes S and S' differ by a permutation of the output bits and by adding a constant after the S-box, in other words, there exists a bit permutation P and a constant c such that

$$S'(x) = P\big(S(x)\big) + c,$$

then those S-boxes cannot be distinguished. We therefore call two S-boxes fulfilling the above relation *equivalent*.

Lemma 4. *Given r sets $D_{e_1}, \ldots, D_{e_r}$ for $1 \leq r \leq 4$, and $e_i \in \mathbb{F}_2^4$, we can construct all sets D_y where $y \in \operatorname{span}(e_1, \ldots, e_r)$.*

Proof. If $y \in \operatorname{span}(e_1, \ldots, e_r)$ then there exists a (non-unique) chain of values

$$y_0 = e_{j_0}, y_1, \ldots, y_s = y$$

such that $y_i \oplus y_{i+1} = e_{j_i}$ for $j_i \in \{1, \ldots, r\}$. We can inductively construct the sets D_{y_i}. First, note that we already know the set $D_{y_0} = D_{e_{j_0}}$, and we can construct $D_{y_{i+1}}$ using the set D_{y_i} and $D_{e_{j_i}}$ given that

$$\{a, b\} \in D_{y_i \oplus e_{j_i}} \quad \Longleftrightarrow \quad \exists c \in \mathbb{F}_2^4 \quad \text{such that} \quad \{a, c\} \in D_{y_i} \text{ and } \{c, b\} \in D_{e_{j_i}}. \quad \square$$

Having this technical lemma in place, we can prove the following theorem (where $\operatorname{wt}(e)$ denotes the Hamming weight of e).

Theorem 3. *Let $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ be a (bijective) S-box and for $e \in \mathbb{F}_2^4$ with $\operatorname{wt}(e) = 1$,*

$$D_e = \big\{\{x, y\} \mid S(x) \oplus S(y) = e\big\}.$$

Given r sets $D_{e_1}, \ldots, D_{e_r}$ for $1 \leq r \leq 4$, up to equivalence, there are

$$\prod_{i=1}^{2^{4-r}-1} 2^4 - i2^r$$

possibilities for S. More concretely,

(i) *given* 4 *sets the S-box is determined uniquely,*
(ii) *given* 3 *sets there are* 8 *possible S-boxes,*
(iii) *given* 2 *sets there are* 384 *possible S-boxes, and*
(iv) *given* 1 *set there are* 645120 *possible S-boxes.*

Proof. Assume we are given r sets $D_{e_1}, \ldots, D_{e_r}$. First, up to equivalence, we can assume that $S(0) = 0$ and furthermore $e_1 = (1, 0, 0, 0)$, $e_2 = (0, 1, 0, 0)$, and so on. We claim that given this information, S is fixed on the set

$$\big\{x \mid S(x) \in \operatorname{span}(e_1, \ldots, e_r)\big\}.$$

For this, let $y \in \mathrm{span}(e_1, \ldots, e_r)$ be given. From Lemma 4 we know that we can construct the set D_y. As D_y passes the cover filter, there exists a pair $\{0, x\} \in D_y$ for some $x \in \mathbb{F}_2^4$. It follows that we found an $x \in \mathbb{F}_2^4$ such that

$$S(0) + S(x) = S(x) = y.$$

More generally, the same argument shows that, given $D_{e_1}, \ldots, D_{e_r}$, fixing $S(x') = y'$ the values of S are fixed for all x such that $S(x)$ is in the coset $y' \oplus \mathrm{span}(e_1, \ldots, e_r)$. Noting there are 2^{4-r} cosets of $\mathrm{span}(e_1, \ldots, e_r)$ and taking into account the bijectivity of the S-box, the theorem follows. □

In particular, the proof of Theorem 3 implies the following.

Corollary 1. *The cover filter is necessary and sufficient. That is to say, given a number of sets D_e where e runs through a subspace of $\mathbb{F}_2^4$, there exists an S-box corresponding to these sets if and only if each of the sets D_e passes the cover filter.*

Appendix B. Example Complexities for Maya

Table B.1. The log of the complexity (number of texts encrypted or decrypted) of 16 test runs of the attack on Maya as a function of the number of rounds. The complexities in italics are extrapolations based on the assumption of a linear relationship between the number of rounds and the log complexity. The median was computed under the assumption that non-existent complexities are infinite.

Case	Rounds										
	6	7	8	9	10	11	12	13	14	15	16
1	14.4	16.2	18.6	21.0	24.3	28.5	31.6	35.5	40.5		*46.8*
2	14.1	15.6	17.3	19.7	22.0	23.7	26.9	29.1	32.0	33.8	36.0
3	14.3	16.3	17.4	19.5	22.2	24.7	27.4	29.7	31.3	33.6	35.9
4	14.8	16.1	17.6	19.8	22.3	25.3	27.9	30.1	32.1	34.8	36.9
5	14.6	15.7	17.4	19.4	21.4	23.5	26.0	27.6	30.0	31.4	35.7
6	15.0	16.1	18.3	20.2	22.7	25.6	28.7	31.8	34.2	36.3	*39.3*
7	14.2	15.6	17.7	19.7	22.4	25.4	27.4	29.9	32.6	35.4	37.4
8	14.5	15.7	17.5	19.4	21.5	24.4	26.9	29.6	31.9	35.5	37.1
9	15.2	16.8	19.1	21.1	23.6	26.5	28.7	31.5	36.3	39.0	*41.2*
10	14.9	16.5	18.1	20.2	23.0	24.5	27.6	29.8	34.7	38.6	38.5
11	14.4	15.6	17.5	19.8	22.1	25.1	27.5	30.5	33.4	37.7	*39.4*
12	15.0	15.7	17.5	19.9	22.4	25.3	29.1	31.5	34.2	36.1	*39.5*
13	14.9	15.9	17.1	19.6	21.7	24.4	27.9	29.3	31.8	35.8	36.0
14	14.4	15.6	17.5	19.3	21.9	24.3	27.7	30.3	32.1	35.4	36.7
15	14.4	15.6	17.2	19.5	22.3	24.0	26.6	29.9	33.0	36.5	40.5
16	14.2	15.7	17.4	19.7	22.4	24.9	27.6	30.4	32.9	34.9	37.4
Median	14.4	15.7	17.5	19.7	22.3	24.8	27.6	30.2	32.5	35.6	37.4

References

[1] M.A. Abdelraheem, G. Leander, E. Zenner, Differential cryptanalysis of round-reduced PRINTcipher: computing roots of permutations, in *Fast Software Encryption 2011, Proceedings*, ed. by A. Joux. Lecture Notes in Computer Science, vol. 6733 (Springer, Berlin, 2011), pp. 1–17

[2] E. Biham, A. Biryukov, How to strengthen DES using existing hardware, in *Advances in Cryptology—ASIACRYPT'94, Proceedings*, ed. by J. Pieprzyk, R. Safavi-Naini. Lecture Notes in Computer Science, vol. 917 (Springer, Berlin, 1995), pp. 398–412

[3] A. Biryukov, A. Shamir, Structural cryptanalysis of SASAS, in *Advances in Cryptology—EUROCRYPT 2001, Proceedings*, ed. by B. Pfitzmann. Lecture Notes in Computer Science, vol. 2045 (Springer, Berlin, 2001), pp. 394–405

[4] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, C. Vikkelsø, PRESENT: an ultra-lightweight block cipher, in *Cryptographic Hardware and Embedded Systems—CHES 2007, Proceedings*, ed. by P. Paillier, I. Verbauwhede. Lecture Notes in Computer Science, vol. 4727 (Springer, Berlin, 2007), pp. 450–466

[5] J. Borghoff, L.R. Knudsen, G. Leander, K. Matusiewicz, Cryptanalysis of C2, in *Advances in Cryptology—CRYPTO 2009, Proceedings*, ed. by S. Halevi. Lecture Notes in Computer Science, vol. 5677 (Springer, Berlin, 2009), pp. 250–266

[6] J. Cho, Linear cryptanalysis of reduced-round PRESENT, in *Topics in Cryptology-CT-RSA 2010, the Cryptographers' Track at the RSA Conference 2010*, San Francisco, CA, USA, March 1–5, 2010 (Springer, Berlin, 2010)

[7] R.M. Corless, G.H. Gonnet, D. Hare, D.J. Jeffery, Lambert's W function in Maple. *Maple Tech. Newsl.* **9**, 12–22 (1993)

[8] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd edn. (Wiley, New York, 1968)

[9] H. Gilbert, P. Chauvaud, A chosen plaintext attack of the 16-round Khufu cryptosystem, in *Advances in Cryptology—CRYPTO'94, Proceedings*, ed. by Y. Desmedt. Lecture Notes in Computer Science, vol. 839 (Springer, Berlin, 1994), pp. 359–368

[10] M. Gomathisankaran, R.B. Lee, Maya: a novel block encryption function, in *International Workshop on Coding and Cryptography 2009, Proceedings* (2010). Available: http://palms.princeton.edu/system/files/maya.pdf

[11] GOST: Gosudarstvennyi Standard 28147-89, *Cryptographic Protection for Data Processing Systems*, government Committee of the USSR for Standards (1989) (in Russian)

[12] L.R. Knudsen, G. Leander, A. Poschmann, M.J.B. Robshaw, PRINTcipher: a block cipher for IC-Printing, in *Cryptographic Hardware and Embedded Systems 2010, Proceedings*, ed. by S. Mangard, F.X. Standaert. Lecture Notes in Computer Science, vol. 6225 (Springer, Berlin, 2010), pp. 16–32

[13] G. Leander, M.A. Abdelraheem, H. AlKhzaimi, E. Zenner, A cryptanalysis of PRINTCIPHER: the invariant subspace attack, in *Advances in Cryptology—CRYPTO 2011, Proceedings*, ed. by P. Rogaway. Lecture Notes in Computer Science, vol. 6841 (Springer, Berlin, 2011), pp. 206–221

[14] R.C. Merkle, Fast software encryption functions, in *Advances in Cryptology—CRYPTO'90, Proceedings*, ed. by A. Menezes, S.A. Vanstone. Lecture Notes in Computer Science, vol. 537 (Springer, Berlin, 1991), pp. 476–501

[15] B. Schneier, Description of a new variable-length key, 64-bit block cipher (Blowfish), in *Fast Software Encryption 1993, Proceedings*, ed. by R.J. Anderson. Lecture Notes in Computer Science, vol. 809 (Springer, Berlin, 1994), pp. 191–204

[16] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, *The Twofish Encryption Algorithm: A 128-Bit Block Cipher* (1999). ISBN 978-0471353812

[17] S. Vaudenay, On the weak keys of blowfish, in *Fast Software Encryption 1996, Proceedings*, ed. by D. Gollmann. Lecture Notes in Computer Science, vol. 1039 (Springer, Berlin, 1996), pp. 27–32