Cryptanalysis of PRESENT-like Ciphers with Secret S-boxes

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

Department of Mathematics, Technical University of Denmark

Abstract. At Eurocrypt 2001, Biryukov and Shamir investigated the security of AES-like ciphers where the substitutions and affine transformations are all key-dependent and successfully cryptanalysed two and a half rounds. This paper considers PRESENT-like ciphers in a similar manner. We focus on the settings where the S-boxes are key dependent, and repeated for every round. We break one particular variant which was proposed in 2009 with practical complexity 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 secret as well.

Keywords. 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 e.g., for identifying and tracking animals or 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 [1] is an important example of a lightweight cipher. It consists of alternate layers of substitutions and permutations.

Important design principles of lightweight ciphers are an efficient hardware implementation, a 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

* Supported by a grant from the Danish research council, grant number 274-07-0246.
** Supported by a grant from the Villum Kann Rasmussen Foundation

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 [2]. 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 [3] 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". In this paper we will investigate the question if such a cipher would be stronger than the original, and if so, how much stronger.

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 and chosen uniformly at random from the set of all such permutations and used repeatedly or as another extreme, a bit permutation is chosen for each round uniformly at random and independently for every round.

The idea of having ciphers where the substitutions are not publicly known and part of the secret key is not new. Notable examples are Khufu [4],the Khufu variation Blowfish [5] and GOST [6] as well as other proposals [7, 8].

Our results. In this paper we focus on the Maya case. We present a novel differential-style attack which enables us to find the S-boxes in the first round one by one.

The attack was implemented and successfully recovered the secret key in versions up to 16 rounds. The complexity of the attack on the 16-round version is approximately 2^{38} using a similar number of chosen plaintexts/chosen ciphertexts. In particular, the proposed cipher Maya can be broken with practical complexity. In our experiments the correct key was 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. Extrapolation 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 uniformly and independent at random can be attacked.

Related work. Biryukov and Shamir investigated the security of iterated ciphers where the substitutions and permutations are all key-dependent [9]. 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 substitution and A stands for 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 [9]. 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 [9]. 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 [10]. 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 [11]. Moreover, the cipher C2, which has a secret S-box, was cryptanalysed by Borghoff et al. [12].

Organisation. The paper is organised as follows. In Section 2 the cipher is presented. Section 3 explains the approach for recovering the secret S-boxes. In Section 4, practical issues of the attack are discussed. In Section 5 we give experimental results for the attack when applied to the Maya cipher [3]. Section 6 describes our model to back up the extrapolations of the experimental data. We outline the more general case and further improvements in Section 7. Section 8 holds the conclusion.

2 The Cipher

We focus on a PRESENT-like cipher where the secret consists of one round key for each round and 16 secret S-boxes. We assume that the round keys and the S-boxes are randomly chosen. In practice these secret components might be derived from a master key using a key schedule which generates key dependent round keys and S-boxes. These 16 randomly chosen 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.

One round of encryption works as follows (cf. Algorithm 1). The current text is divided into nibbles of 4 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 the round-key.

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 S-boxes S_i and N round keys K_i from K
2: $\mathsf{STATE} \leftarrow X$
3: **for** $i = 1$ to N **do**
4: Parse STATE as $\mathsf{STATE}_0 \| \cdots \| \mathsf{STATE}_{15}$, where each STATE_j is a four-bit nibble
5: **for** $j = 0$ to 15 **do** {Substitution layer}
6: $\mathsf{STATE}_j \leftarrow S_j(\mathsf{STATE}_j)$
7: **end for**
8: Reassemble STATE
9: Apply bit permutation to STATE
10: Add round key K_i to STATE
11: **end for**
12: $C \leftarrow \mathsf{STATE}$

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

The cipher Maya, proposed by Gomathisankaran and Lee [3], is an instance of the cipher described in Algorithm 1 with $N = 16$. The authors claim that it is efficient in a hardware implementation.

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 16 S-boxes xored with the last round key. Once this is done, 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.

Furthermore, we shall outline how our attack can be applied to a generalization. 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 extreme variant can be compared with an instance of SASAS [9]. Note that in this variant nothing but the block size and the number of rounds is known. The pseudo-code of this variant is described as 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 Section 6.

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 these 16 S-boxes S_i,

Require: X is a 64-bit plaintext
Ensure: $C = E_K(X)$ where E_K means the encryption function with key K
1: Derive $16 \cdot N$ S-boxes $S_{i,j}$, $1 \leq i \leq N$, $0 \leq j \leq 15$ and $N-1$ bit permutations P_i from K
2: $\mathsf{STATE} \leftarrow X$
3: **for** $i = 1$ to N **do**
4: Parse STATE as $\mathsf{STATE}_0 \| \cdots \| \mathsf{STATE}_{15}$, where each STATE_j is a four-bit nibble
5: **for** $j = 0$ to 15 **do** {Substitution layer}
6: $\mathsf{STATE}_j \leftarrow S_{i,j}(\mathsf{STATE}_j)$
7: **end for**
8: Reassemble STATE
9: **if** $i < N$ **then**
10: Apply bit permutation P_i to STATE
11: **end if**
12: **end for**
13: $C \leftarrow \mathsf{STATE}$

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.

$0 \leq i < 16$, and we note that all S_i are bijective mappings with the signature $\mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$. 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

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

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

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

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, which has been experimentally verified to hold in most cases.

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

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

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

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 $|D_e| = 8$. We shall return to this partitioning method in a moment. Our final goal will be to learn all four slender sets D_e.

Generalizing 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 S-boxes. The above observations can clearly be generalized 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 $u + v \geq 6$, 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 generalizes 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 2 in Appendix A. Therefore, we shall generally only be interested in the four slender sets, since all 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.

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.

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. However, counting the number of S-boxes that give rise to these sets is somewhat inefficient (see, however, Section 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. We call this filter the *existence filter*.

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 $\{x_1, y_1\}$ 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:

$$
\begin{aligned}
D' &= \{\ \{a, b\}\ ,\ \{c, d\}\ \ldots \\
D'' &= \{\ \{a, c\}\ ,\ \{b, d\}\ \ldots
\end{aligned}
$$

3.1 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 we will get, and so there is a tradeoff 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. When we have 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 passes 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 that 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:

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

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

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

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 (on the assumption 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'll be recovering the wrong S-box. This does happen in practice, although it is rather rare.

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

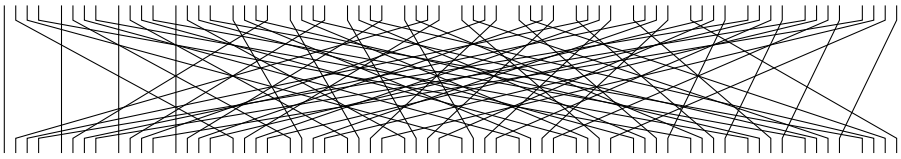
Once we have identified as many sets as possible using this method (for both the S-box and its inverse), we can apply the existence filter to check if these sets can possibly be valid; if there is no S-box generating these sets, then the sets are obviously not valid. As mentioned in Section 3, applying 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 (or a candidate for it) 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.

5 Case Study: the Block Cipher Maya

Maya is a block cipher proposed at WCC 2009 [3]. It is a PRESENT-like cipher with key dependent S-boxes (repeated in every round) and a fixed, known bit permutation (see Fig. 1). 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



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. 1. The Maya bit permutation.

S-box. In this way 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 round 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 and below the log of the complexity 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	*45.5*	36.0	35.9	36.9	35.7	*39.3*	37.4	37.1	*40.6*	*38.5*	*39.4*	*39.5*	36.0	36.7	*38.3*	37.4

Moreover, Table 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. 2 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 round 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 (or the given top 32 pairs). 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

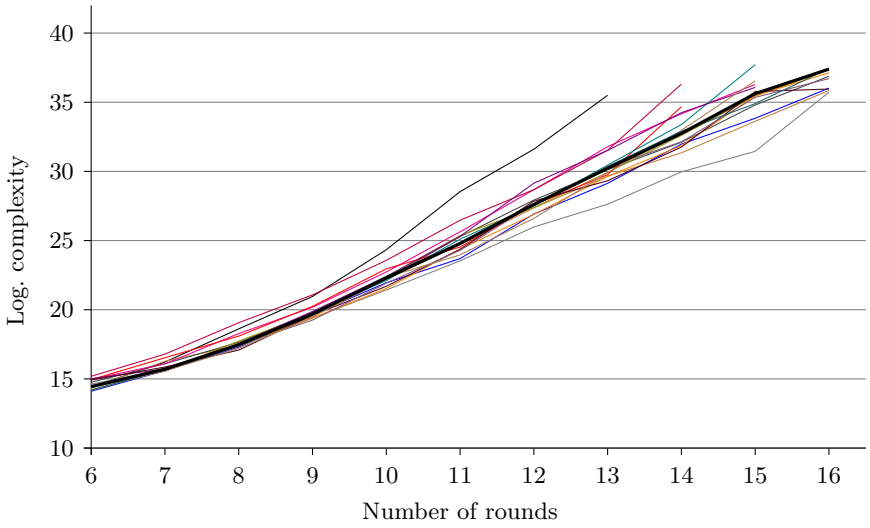


Fig. 2. A graphical representation of the data in Table 1. The thick line represents the median computed for each number of rounds.

no candidates for this partitioning, one might try swapping elements between sets in such a way that the bowtie condition still holds.

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 to recover one or more sets D_e has small complexity and it is possible to get sufficient experimental data. However, 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 to group 120 counters $C(\{x, y\})$, each belonging to an input pair to an S-box of the first round, into 15 distinct groups. All pairs within a group 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 to 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 only 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 finally 8 counters distributed with parameters (n, p_8) (here we identified $e = (0, 0, 0, 1)$ with 1, $e = (0, 0, 1, 0)$ with 2 etc.). Without loss of generality we assume $p_1 \geq p_2 \geq p_4 \geq p_8$ and that holds $p_1 \neq p_2$. The attack works by looking at the 8 highest counters and is 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 $\{x_1, y_1\}$ with output difference '1' and a 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 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

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

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 following 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 ([13]). *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

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

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 $c = \text{LambertW}\left(\frac{1}{2\,\text{err}^2\,\pi}\right)$ [14] is a small constant depending on the error. As example we can assume that $q = 2p$ then in order for the attack to be successful we need a sample of size $\frac{3c}{p}$.

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 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. Therefore we determine the difference distribution matrix which contains only the probabilities for 1 to 1 bit differences, which as it turns out when comparing to experimental data, is a good approximation. This matrix is of size only 64×64. This enables us to simulate the propagation of 1 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 therefore 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 formula (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

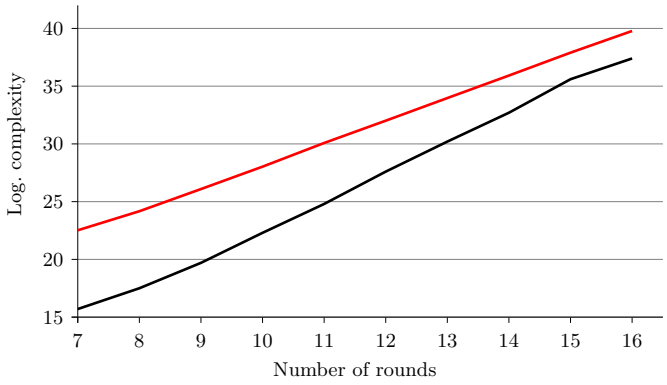


Fig. 3. Comparison between the medians of the experimental data and the model for recovering two sets D_e in each direction. The black line shows the experimental data while the red (gray) line shows the data from the model. The complexity unit is one plaintext.

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 a single active S-box will on average not lead to 16 active S-box 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 support this assumption.

To justify the introduced model we implemented the attack for a small number of rounds (see Section 5). For each number of rounds we sampled 1000 ciphers in our model to determine the sample size needed to distinguish between the two distributions. Fig. 3 gives a comparison of the experimental data with that of the model for the case that we want to recover at least four sets D_e for all 16 S-boxes. The black line shows the experimental data and the red 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 close to 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 modeled data suggest that we are able to break up to 28 rounds before we reach the bound of 2^{64} available plaintexts.

7 Extensions

In this section we outline some possible extensions of our attack. This includes some further improvements (cf. Section 7.1) as well as attacks on the more general variant of the cipher where all components in all rounds are chosen independently and uniformly at random (cf. Section 7.2).

7.1 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 to a single-bit difference at the input to the S-box layer in the last round or to 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 pairs of eight values related to a certain linear characteristic. It was confirmed in a small number of experiments on ciphers with a small number of rounds that this approach can be used to derive information about the S-boxes. One natural future direction of research is to combine the differential-style attack outline in this paper with a similar linear-style attack.

7.2 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.

For such a 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 for more details. One needs to recover all four slender sets D_e for each S-boxes. 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 4 slender sets where 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 remain. 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.

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 for the pair of positions differing in the plaintext. This is repeated a number of times for all pairs of bit positions. One may now assume that the highest counter values correspond to pairs of bit positions that are mapped to the same S-box input.

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 (if not lower than) the complexity of recovering S-boxes.

8 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}. It is interesting to note that the best known attack on PRESENT, a linear attack, can be used to cryptanalyse up to 26 rounds of PRESENT (which has publicly known but carefully chosen S-boxes and bit permutation).

Also, the variant where the S-boxes and bit permutations are chosen at random for every round can also be attacked with a data complexity of less than 2^{64} for up to 16 rounds.

It is clear that our attacks exploit that there are weak differential properties for some randomly chosen four-bit S-boxes, and they do not apply to ciphers where the S-boxes are chosen as in PRESENT. However, a restriction to strong S-boxes (w.r.t to differential cryptanalysis) would also limit the size of the key.

References

1. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsø, C.: PRESENT: An Ultra-Lightweight Block Cipher. In Paillier, P., Verbauwhede, I., eds.: Cryptographic Hardware and Embedded Systems – CHES 2007, Proceedings. Volume 4727 of Lecture Notes in Computer Science., Springer (2007) 450–466

2. Cho, J.: 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 (2010)
3. Gomathisankaran, M., Lee, R.B.: Maya: A Novel Block Encryption Function. International Workshop on Coding and Cryptography 2009, Proceedings. Available: http://palms.princeton.edu/system/files/maya.pdf (2010/02/14).
4. Merkle, R.C.: Fast Software Encryption Functions. In Menezes, A., Vanstone, S.A., eds.: Advances in Cryptology – CRYPTO '90, Proceedings. Volume 537 of Lecture Notes in Computer Science., Springer (1991) 476–501
5. Schneier, B.: Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In Anderson, R.J., ed.: Fast Software Encryption 1993, Proceedings. Volume 809 of Lecture Notes in Computer Science., Springer (1994) 191–204
6. GOST: Gosudarstvennyi standard 28147-89, *cryptographic protection for data processing systems*. Government Committee of the USSR for Standards, 1989 (in Russian).
7. Biham, E., Biryukov, A.: How to Strengthen DES Using Existing Hardware. In Pieprzyk, J., Safavi-Naini, R., eds.: Advances in Cryptology – ASIACRYPT '94, Proceedings. Volume 917 of Lecture Notes in Computer Science., Springer (1995) 398–412
8. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: A 128-Bit Block Cipher Submitted as candidate for AES. Available: http://www.schneier.com/paper-twofish-paper.pdf (2010/02/05).
9. Biryukov, A., Shamir, A.: Structural Cryptanalysis of SASAS. In Pfitzmann, B., ed.: Advances in Cryptology – EUROCRYPT 2001, Proceedings. Volume 2045 of Lecture Notes in Computer Science., Springer (2001) 394–405
10. Gilbert, H., Chauvaud, P.: A Chosen Plaintext Attack of the 16-round Khufu Cryptosystem. In Desmedt, Y., ed.: Advances in Cryptology – CRYPTO '94, Proceedings. Volume 839 of Lecture Notes in Computer Science., Springer (1994) 359–368
11. Vaudenay, S.: On the Weak Keys of Blowfish. In Gollmann, D., ed.: Fast Software Encryption 1996, Proceedings. Volume 1039 of Lecture Notes in Computer Science., Springer (1996) 27–32
12. Borghoff, J., Knudsen, L.R., Leander, G., Matusiewicz, K.: Cryptanalysis of C2. In Halevi, S., ed.: Advances in Cryptology – CRYPTO 2009, Proceedings. Volume 5677 of Lecture Notes in Computer Science., Springer (2009) 250–266
13. Feller, W.: An Introduction to Probability Theory and Its Applications. 3rd edn. John Wiley and Sons (1968)
14. Corless, R.M., Gonnet, G.H., Hare, D., Jeffery, D.J.: Lambert's W function in Maple. Maple Technical Newsletter 9 (1993)

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

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

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

Lemma 2. *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 $y \in \mathrm{span}(e_1, \ldots, e_r)$.*

Proof. If $y \in \mathrm{span}(e_1, \ldots, e_r)$ then there exists a (not 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

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

□

Having this technical lemma in place, we can prove the following theorem.

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 $\mathrm{wt}(e) = 1$,*

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

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,

1. *given 4 sets the S-box is determined uniquely,*
2. *given 3 sets there are 8 possible S-boxes,*
3. *given 2 sets there are 384 possible S-boxes, and*
4. *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

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

For this, let $y \in \mathrm{span}(e_1, \ldots, e_r)$ be given. From Lemma 2 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 that 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.*

B Example Complexities for Maya

Table 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 on the assumption that non-existent complexities are infinite.

	Rounds										
Case	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