Multi-Trail Statistical Saturation Attacks

B. Collard*, F.-X. Standaert**

UCL Crypto Group, Microelectronics Laboratory, Université catholique de Louvain.
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium.
baudoin.collard; fstandae@uclouvain.be

Abstract. Statistical Saturation Attacks have been introduced and applied to the block cipher PRESENT at CT-RSA 2009. In this paper, we consider their natural extensions. First, we investigate the existence of better trails than the one used in the former attack. For this purpose, we provide a theoretical evaluation of the trail distributions using probability transition matrices. Since the exhaustive evaluation of all possible distributions turned out to be computationally hard, we additionally provide a heuristic branch-and-bound algorithm that allows us to generate a large number of good trails. These tools confirm that the trail of CT-RSA 2009 was among the best possible ones, but also suggest that numerous other trails have similar properties. As a consequence, we investigate the use of multiple trails and show that it allows significant improvements of the previous cryptanalysis attempts against PRESENT. Estimated complexities indicate that PRESENT-80 is safe against key recovery, by a small security margin. We also discuss the impact of multiple trails for the security of the full PRESENT-128. We finally put forward a "statistical hull" effect that makes the precise theoretical analysis of our results difficult, when the number of block cipher rounds increases.

Introduction

PRESENT is a block cipher presented at CHES 2007, that was designed for small embedded applications [3]. It has a Substitution Permutation Network architecture, with a 64-bit block size and 31 rounds. The same 4-bit S-box is applied 16 times in parallel in each round. The designers have proposed two key sizes: 80 and 128 bits. Due to its simple and elegant structure, it has been the focus of different cryptanalysis attempts. In [23], the author presented a first attack against PRESENT, using differential cryptanalysis. It applies to 16 block cipher rounds and requires the whole codebook and a time complexity of 2^{65}. In 2009, Ozen et al. proposed a related key rectangle attack against up to 17 rounds of PRESENT, with a time complexity of 2^{104} and 2^{63} chosen plaintexts [20]. Two papers exploit the linear cryptanalysis and target 26 rounds, respectively by taking advantage of the linear hull effect [17] and multiple approximations [10]. These attacks require the whole codebook. Finally, [19] combines linear cryptanalysis and weak keys and targets up to 28 rounds in this context.

* Work supported by the project Nanotic-Cosmos of the Walloon Region.
** Associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

In this paper, we pay a particular attention to a Statistical Saturation Attack that was specifically devised for the cryptanalysis of PRESENT (although it could apply to other ciphers). It exploits the weak diffusion of certain bits (called the trail) during the encryption process, when some of the plaintext bits are fixed. This property did lead to an estimated attack against up to 24 rounds, using approximately 2^{60} chosen plaintexts. As detailed in [6], the main limitation when trying to extend this technique towards more rounds is the data complexity that exceeds the complete codebook. We consequently investigated the tracks that could be used to get rid of this limitation. Our contributions are threefold.

First, we provide tools allowing one to approximate the diffusion in a trail, using Markov chains. We exhibit that, besides the iterative trail proposed in [6], there exists many other trails with a similarly weak diffusion. We also propose a heuristic branch-and-bound algorithm in order to generate them efficiently. We finally confirm our theoretical analysis with experiments that can break up to 16 rounds PRESENT. Then, in a second part of the paper, we investigate the exploitation of these multiple trails. We show that they can be used to trade data complexity for time complexity. As a result, we discuss the possibility to mount a key-recovery attack against the full 31-round PRESENT-128, using the complete codebook, and a time complexity below 2^{128} memory accesses. We put forward that such an attack could be possible under certain (optimistic) conditions of independence for the trails - the exact evaluation of these conditions being an important scope for further research. We note that such an attack would anyway be of theoretical interest only. In particular, the authors in [3] clearly suggest PRESENT-80 for their target applications (rather than PRESENT-128). However, these results question the number of rounds for PRESENT-128 and highlights that they could have been increased over those for PRESENT-80. Finally, in a third part of the paper, we put forward a "statistical hull effect", *i.e.* a counterpart of the linear hull effect in linear cryptanalysis. We discuss its impact for the theoretical analysis of our estimated attack complexities.

1 The Statistical Saturation Attack

1.1 Principle of the attack

The Statistical Saturation Attack, originally described in [6], takes advantage of a weakness in the diffusion layer of PRESENT. For the S-boxes 5, 6, 9 and 10 (called the *active* S-boxes), only 8 out of 16 input bits are directed to other S-boxes. Figure 1 illustrates this observation (note that there exists many other examples of weak diffusion in the permutation). Consequently, if we fix the 16 bits at the input of the active S-boxes, then 8 bits will be known at the very same input for the next round. We can iteratively repeat this process round by round and observe a non-uniform behavior at the output of the active S-boxes.

Thanks to this non-uniform behavior, 16 bits of the last subkey can be recovered as follows. We first generate a large number of plaintexts with 8 fixed bits. The plaintexts are encrypted using r-rounds PRESENT and the distribution of

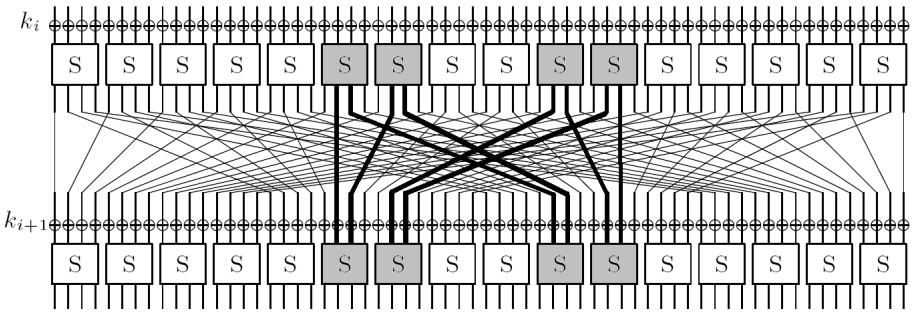

Fig. 1: Permutation layer of PRESENT: bold lines show the weak diffusion property.

the ciphertexts are recorded for the 16 bits at the output of the 4 active S-boxes in the last round. Given this experimental distribution, it is possible to compute the output distribution of the target 8-bit trail one round before, using a partial decryption process. For one key guess, the evaluation of such an $r-1$-round distribution requires 2^{16} computations. Hence the total time complexity for all the key guesses equals $2^{16} * 2^{16} = 2^{32}$. Additionally using an FFT-based trick similar to the technique presented in [4], this complexity can be decreased to $16 \cdot 2^{16} \cdot 2^8$. For the correct key guess, the experimental 8-bit distribution in the penultimate round is expected to be more non-uniform than for any other guess. This is because decrypting with a wrong guess is expected to have the same effect as encrypting one more round. We can thus hope to distinguish the correct key from the wrong ones by computing the distance between a partially decrypted distribution and the uniform distribution. If the attack works properly, the distribution with the highest distance should correspond to the correct key.

1.2 Extensions of the attack

In [6], the authors propose 3 extensions to improve the cryptanalysis:

(ext. 1) Increase the fixed part in the plaintexts. One can easily gain one round in the attack by simply fixing the 16 bits of plaintext corresponding to the 4 active input S-boxes of the trail. This way, the 8-bit trail in the second round is also fixed and the diffusion is postponed by one round. By fixing 32 bits out of 64 (corresponding to S-boxes 4-5-6-7-8-9-10-11), one can similarly extend the attack by 2 rounds. However, we are then limited in the generation of at most 2^{32} texts. This limitation may be mitigated with the following extension.

(ext. 2) Use multiple fixed plaintext values. The same analysis can be performed multiple times, using different values for the 8-bit (or 16- or 32-bit) fixed part of the plaintexts and then combining the results (*e.g.* taking the sum of the uniform *vs.* measured distances corresponding to the different fixed plaintexts). This allows exploiting more texts and moving to a known-plaintext context. The resulting attack is similar to multiple linear cryptanalysis: each fixed part of the plaintext can be seen as analogous to an additional approximation in [2, 11].

(ext. 3) Partial decryption of two rounds instead of one. In this case, 8 S-boxes are active in the last round and 4 S-boxes are active in the penultimate round. As detailed in [7] and illustrated in Figure 4, one can perform two independent partial decryptions in parallel, in order to decrease the time complexity of the attack down to $2 \cdot (16 \cdot 2^{16} \cdot 2^8) \cdot (8 \cdot 2^8 \cdot 2^4) = 2^{44}$ elementary operations.

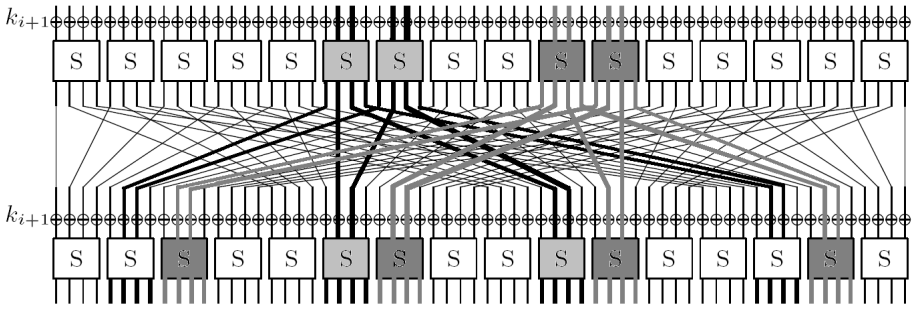

Fig. 2: Practical trails for 2-round partial decryption in PRESENT with reduced time complexity. The two independent trails are shown in different shades of gray.

2 Evaluating the trail distributions with Markov chains

As a matter of fact, the previous attack essentially exploits the property that it is possible to evaluate the distribution of a subset of output bits given the distribution of a subset of input bits for one round of PRESENT. Minier and Gilbert use a similar technique in their attack against Crypton [16]. In [6], the authors exploited an iterative trail with 8 active bits in 4 active S-boxes in each round. However, as already mentioned, this is not the only possible trail and an interesting problem is to determine if there are other trails leading to better attacks. In this section, we show how to evaluate the distribution of a trail going through several rounds of PRESENT. For this purpose, we characterize such a trail by a matrix containing the transition probabilities between the inputs and outputs. Additionally, we rely on the assumption that the bits that are not part of the trail are uniformly distributed. This assumption as well as the Markov chains that we exploit in the following were also used by Vaudenay in his paper on χ^2 cryptanalysis [21]. As will be discussed in Section 5, this is becoming incorrect as the number of rounds in the trail increases. But as the next section will show, this assumption is required in order to limit the computational cost of our estimations to tractable values, when comparing different trails.

2.1 Transition matrix for an S-box

Let us consider an active S-box with size $n*n$, and suppose that the trail includes i active bits among n in input and j active bits in output. Consequently, there are 2^i possible inputs and 2^j possible outputs, and the size of the transition matrix is $2^i * 2^j$. This matrix is constructed in the following way:

– Initialize a matrix of size $2^i * 2^j$ and fill it with zeros.
– For every possible S-box input value, extract a masked i-bit input and the j-bit output, and increment the matrix in the corresponding position.
– Multiply the matrix by $2^i/2^j$ for normalization.

By construction, any transition matrix has the properties that the sum over any row is equal to one. For example, the iterative trail represented in Figure 1 uses the same transition matrix for each active S-box:

$$A = \begin{pmatrix} 0 & 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0 & 0.5 \\ 0.5 & 0 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 & 0 \end{pmatrix}$$

In this case, the matrix is square, but it is not mandatory as it depends on the number of active input/output bits. The interpretation of the transition matrix is easy: each row represents a possible value for the input and the column represents the probability of transition to a particular output value given the input.

2.2 Transition matrix for the permutation layer

For a permutation layer like the one used in PRESENT, the number of active bits in output is equal to the number k of active bits in input. Consequently, the matrix is square with size $2^k * 2^k$. Moreover, at each input value in the trail corresponds one and only one output value and thus the transition matrix contains only zeros and ones (i.e. it is a particular instance of permutation matrix).

2.3 Transition matrix for the subkey addition

The effect of a XOR between the input bits in the trail and unknown key bits is similar to a permutation. To each value in the trail before the key addition corresponds only one output value after the key addition. Hence, the transition matrix for the subkey addition is also a permutation matrix. However, unlike the transition matrix for the permutation layer, this transition matrix does not increase the diffusion in the trail. Intuitively, this is because the key addition does not mix the active bits coming from different active S-boxes as with the permutation layer. Mathematically, this corresponds to the property that the transition matrix can be decomposed into a Kronecker product of small submatrices. Consequently, given the assumption of uniform distribution for the bits that are not part of the trail, different subkeys have different output distributions in the trail, but they present an identical non-uniform behavior. Hence, it is sound to compute the distribution of a trail independently of the keys.

2.4 Composition of transition matrices

If several S-boxes are active in parallel in a trail, the overall transition matrix is given by the Kronecker product of the transition matrix related to each S-box. The transition matrix of a round can then be computed as the matrix product

of the transition matrices for the S-box and permutation layers. Thereafter, given the transition matrix for a complete trail, the output distribution can be directly evaluated as the the vector-matrix product of the input distribution and the transition matrix. The main drawback of this method is that it requires to compute a matrix product with matrices of size $2^n * 2^n$ where n is the number of active bits involved at any point in the trail during the encryption process.

2.5 Practical example

We illustrate this technique using the iterative trail of Figure 1. This trail is composed of 4 active S-boxes at each round, with 2 bits out of 4 active in each S-box. As there are 8 active bits at each round, the full transition matrix has a size of $2^8 * 2^8$. It is computed in the following way: The matrix transition for the 4 parallel S-boxes is computed as: $A^4 = A \otimes A \otimes A \otimes A$, where $\otimes$ is the symbol for the Kronecker product. The matrix for one full round is then given by $R = A^4 \cdot P$ (where P is the transition matrix for the permution on 8 bits). The transition matrix after n rounds is then $R^n = \underbrace{R \cdot R.... \cdot R}_{n\ times}$. Given a vector d_{in} of size $(1 * 2^8)$ describing the distribution of the 8-bits active bits in input, the distribution of the output active bits is finally given by $d_{out} = d_{in} \cdot R^n$.

3 Heuristic Branch-and-Bound for trail search

As detailed in the previous section, the evaluation of the distribution for a single trail can be computationally intensive if this trail involves a lot of active bits. This was not an issue for the attack in [6], but may become the limiting factor for other trails (or other ciphers). In order to mitigate this limitation, we now present a heuristic algorithm based on the branch-and-bound proposed by Matsui for linear approximation search in [15]. The goal of this heuristic is to perform a pre-selection of "interesting trails" that minimizes the number of active S-boxes (so that the treatment of the previous section can still be applied) while trying to limit the diffusion based on a simpler criteria than a probabilistic distance.

3.1 Description of the algorithm

The basic principle of the heuristic is to maximize the ratio between the number of active bits and the number of active S-boxes in a trail. While this criteria does not ensure finding the best trail distributions, it is extremely fast to evaluate and to integrate into the execution of a branch-and-bound algorithm. As will be shown later in the section, it also provides reasonably good results. The justification is the following: even though all the trails with low diffusion are not necessarily good trails (because of the influence of the transition matrices for the S-boxes), all the good trails must have low diffusion in their permutations layers. Consequently, a good strategy is to first generate a large number of trail candidates with a branch-and-bound heuristic, then to compute the full transition matrices for each of these candidates and to select the best ones only.

In order to speed-up the execution of the algorithm, we used a similar implementation technique as presented in [4]. That is, we start by exhaustively counting the couples (input, output) of the permutation for which the number of active S-boxes is low. Such couples are called *permutation candidates* and are entirely defined by the number of active input and output S-boxes, the position of these S-boxes and their corresponding mask value. The permutation candidates are then stored in a database (a hash table) instead of being generated on-the-fly during each branching phase. All the candidates having the same active output S-boxes are stored in the same list. Once the database is created, we launch the actual trail search: a trail on r rounds can be obtained by the concatenation of r permutation candidates, if the positions of the active S-boxes at the exit of a permutation candidate correspond to those of the active S-boxes at the input of the next candidate. These constraints are easily checked, as the candidates are picked up in the database according to the position of their active output S-boxes. The objective function that we need to maximize is the average ratio between the number of active bits and the number of active S-boxes in the trail. Note finally that we pile up the candidates starting with the last round, then going down gradually until the first round, in order to benefit from the knowledge of the best ratio in each phase of the branch-and-bound.

3.2 Results

As an illustration, we generated 1000 trails with maximum 5 active S-boxes in each round and computed the theoretical data complexity as in [6], for the distinguishers based on each of these trails. That is, following the analysis of Baignère et al. [1], we estimate the data complexity as proportional to the inverse of the Euclidean distance between the distributions evaluated in Section 2 and a uniform distribution. The results in the figure show that after 15 rounds, the data complexity varies between 2^{50} and 2^{66}, according to the trail. The original

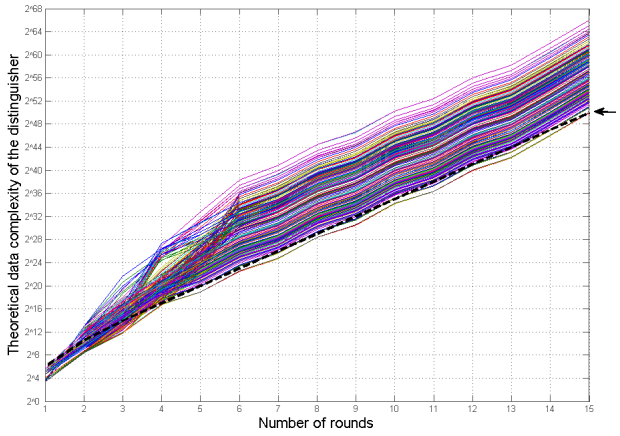

Fig. 3: Theoretical data complexity for distinguishers based on 1000 different trails.

trail of Figure 1 is marked with an arrow and is among the best ones (see Figure 3). Note that by increasing the number of trails generated by the branch-and-bound (beyond 1000), we can easily produce very large amounts of trails with good theoretical data complexities. As will be detailed in Section 5, the amount of such trails increases exponentially with the number of rounds.

3.3 Experimental validation of the estimated data complexity

The estimations in the previous section indicate that the data complexity increases by approximately 2^3 for every additional round. As these estimations rely on the assumptions needed to evaluate the distributions in Section 2, we confirmed these predictions experimentally, in order to verify that our assumptions hold to a sufficient extent. For this purpose, we complemented the experiments in [6] and attacked up to 15 rounds PRESENT with 2^{32} texts, using a 2-round partial decryption process. Figure 4 illustrates that we gain one round compared to the original attack of CT-RSA, and confirms the theoretical expectations. Note that the two-round decryption also allows a significantly increased gain (because there are more key bits guessed in the experiment).

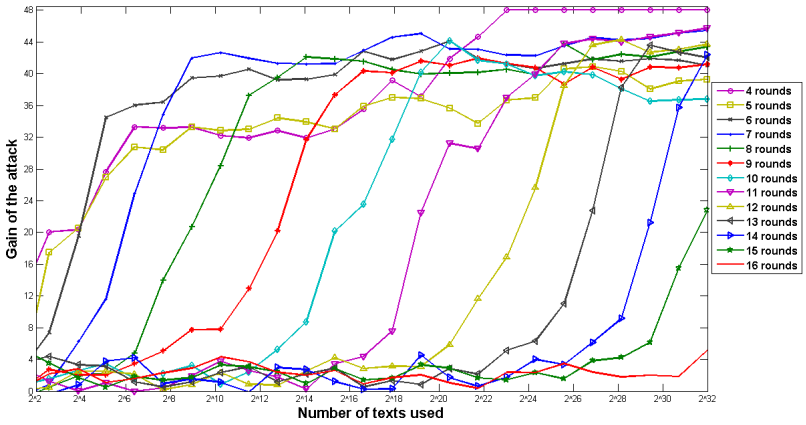

Fig. 4: Average gain of 6 attacks against 4 to 16 rounds PRESENT, using up to 2^{32} texts (these attacks exploit **ext. 1**, with 32 fixed bits and **ext. 3**).

4 Multiple trails

The previous section shows that the simple trail of Figure 1 is among the best ones to perform a Statistical Saturation Attack against PRESENT. On the other hand, we also observe that a large number of trails perform similarly good in theory. Hence, a natural idea is to investigate the use of multiple trails, as can be done in linear cryptanalysis with multiple approximations [2]. In the following, we consequently consider two questions. First, we study the possibility to exploit several trails with different input masks and the same output mask, in order to

increase the gain of the attack. Our experiments suggest that this technique yields good results and allows improving the best-reported cryptanalysis against PRESENT. Then, in Section 5, we show that there exists many different trails with the same input and output masks, the combination of which affects the distribution of the output in a hardly predictable way. We discuss the impact of such a "statistical hull" effect on the assumptions of Section 2.

4.1 (ext. 4) Multiple trails cryptanalysis

In the Statistical Saturation Attack of [6], the main limitation of the attack was the number of texts required to find the correct subkey. Above 24 rounds (and assuming that **ext. 2** yields the expected improvements), the data complexity of the attack reaches the codebook size of PRESENT. In this section, we consequently investigate the possibility to use several distinct trails in order to partially remove this limitation. Thanks to our branch-and-bound, we were able to generate many trails with different input masks and the same output mask. It allowed us to run several independent attacks in parallel, each one using a different trail and thus different partitioning of the plaintexts. As the output mask is the same, we can combine the results of the attacks together because the partial decryption involves the same subkey bits. In practice, each single-trail attack produces a vector containing the distances between the uniform and output distributions after partial decryption with the keyguess. A straightforward combination that was used in the context of linear cryptanalysis using multiple approximations (and for **ext. 2**) simply consists of taking the mean of these vectors. Such heuristic may not be optimal (*e.g.* compared to a maximum likelihood approach), but as detailed in [5], it is convenient when we lack the exact information about the expected distribution of the trail output after partial decryption. In particular, it is useful when linear hull (or related) effects imply errors when determining the approximated probability density functions of multidimensional approximations in linear cryptanalysis (see [9]). As will be exhibited in Section 5, this is exactly the type of situation that we face in this paper.

If we use n such trails, the time complexity is multiplied by the same factor because we have to repeat the partial decryption for each trail. The overhead required to combine the results is negligible. The effect on the data complexity is more intricate because each input mask defines a different partition of the plaintexts, according to the bits that are fixed and those that can change. For example, if the whole codebook is used, each plaintext will be used exactly once for each trail. The plaintexts can either be stored, requiring 2^{67} bytes of memory, or they can be generated on-the-fly, which would require $n * 2^{64}$ encryptions.

In order to evaluate the feasibility of this technique, we applied the statistical saturation attack on 9-round PRESENT for 128 different input masks with 2^{28} texts each. We selected the trails according to two different rules:

1. Best trails: we generated masks using our branch-and-bound and we selected the 8-round trails leading to the lowest theoretical data complexity.
2. Random trails: we generated trails from random (compatible) masks.

The results of our experiments are in Figure 5, which compares the mean gain (as defined in [2]) of attacks against 9-round PRESENT, exploiting different amounts of trails (up to 128), in function of the number of plaintexts used in the attacks. They illustrate that the combination of the information coming from different trails can be done constructively (*i.e.* lead to increased gains). For example, reaching a gain of 10 bits with a single trail requires approximately 2^{24} texts in the left part of the figure. But a combination of 2^7 trails leads to a nearly equivalent gain after 2^{18} texts in this case. Interestingly, the positive impact of combining several trails appears to be reduced for the random trails case (in the left part of the figure). In addition, there are two phenomenons that are worth being mentioned. First, the practical gains of trails having similar theoretical data complexity turned out to be quite different. For example, in the context of the best selected trails, we observed that 12.5% of them led to relatively low bias (less than 4 bits) even after 2^{28} texts. Second, the difference between the best and random trails was not as strong in practice as expected from the theoretical data complexities computed in the previous section. In both cases, this observation relates to the "statistical hull" effect discussed in Section 5.

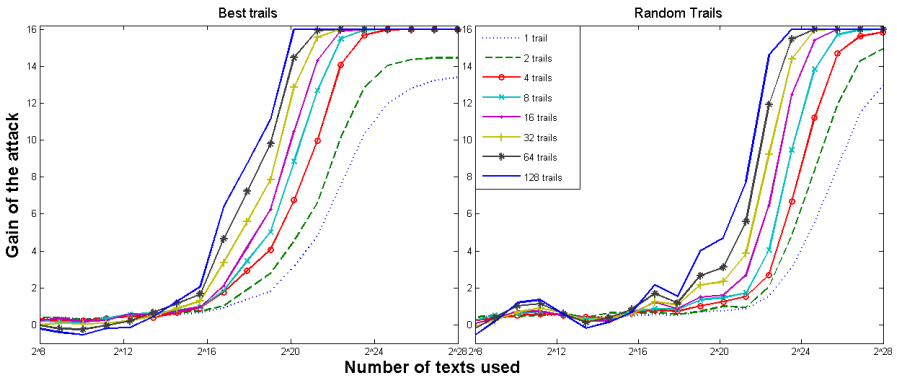

Fig. 5: Gains of attacks using multiple trails (left: best trails, right: random trails).

Note that, since the experiments in Figure 5 are far from using the full codebook of PRESENT, the attacks using different trails also use different plaintexts. By contrast, when estimating the effectiveness of attacks against more than 26-round PRESENT, the data complexity gets close to the full codebook. It means that exploiting multiple trails will require to rearrange (and hence, reuse) the codebook several times. Such a context raises the question to evaluate whether these multiple partitions of the codebook also improve the gain of the attack. Quite naturally, generating the full codebook is unfeasible for a 64-bit cipher. As a first step, we consequently considered a reduced-size version of PRESENT, with 16-bit blocks [12]. This allowed us to compute the average gain of one versus a combination of 128 trails, for different amounts of plaintexts. The results of these experiments are in Figure 6, for attacks against different number of rounds.

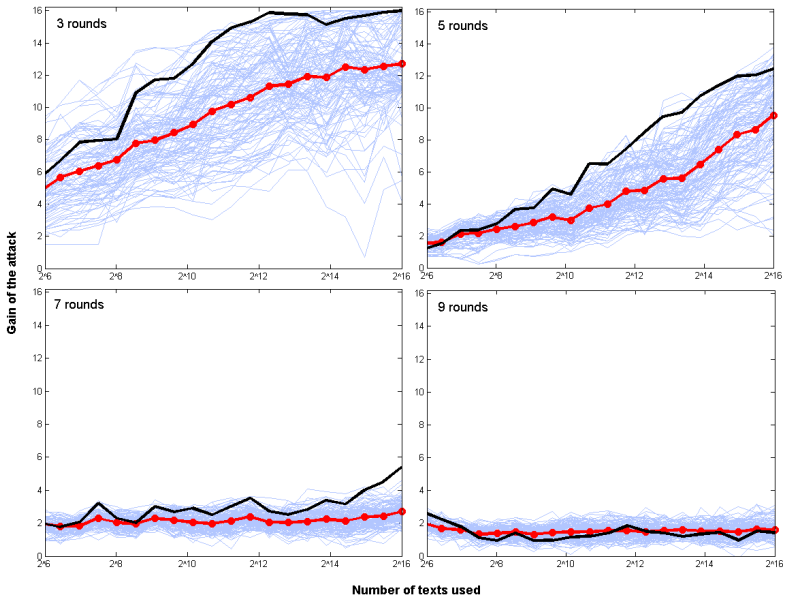

Fig. 6: Comparison between the gain of a single trail and the combined gain of 128 trails, using the full codebook against a simplified PRESENT with 16-bit block size.

A first observation is that, even in this extreme context, the combination of the trails improves the overall gain of the attack significantly. That is, the bold plain curves (representing the combined gains) exceed the bold dotted curves (representing the average gains of single trails - the other curves representing all the 128 single-trail experiments). On the other hand, the improvements are not as large as in Figure 5, arguably because in such a small scale example, the input masks of the different trails are correlated. Also, it is noticeable that for the 9-round case, the gain of the multi-trail attack is similar to the one of a single-trail attack. This illustrates a context where the key-dependent signal provided by a single trail is so small that combining 128 multiple trails is not sufficient to reach a significant gain (since multiple trails can only be used to amplify an existing signal, here too small for the considered data complexities).

Summarizing, in the best case, different trails bring independent information, meaning that using two trails is equivalent to doubling the amount of texts with a single trail (this is the expectation in multiple linear cryptanalysis [2, 9])[1]. In practice, these (best) conditions of independence (*e.g.* for the masks) are not perfectly respected in our context. But as the previous experiments illustrate for reduced-round PRESENT, different trails yield useful information, even when recombining the same set of plaintext with correlated masks. We leave the exact evaluation of these dependencies as an important scope for further research.

[1] Just as it is expected when using multiple fixed values in **ext. 2**.

4.2 Consequence for the security of PRESENT-128

The previous section showed experimentally that combining multiple trails can lead to an improvement of the attack's gain with constant data complexity. In this section, we consider the impact of this observation for the security of PRESENT and quantify the overheads that it causes in terms of time and memory complexity. In particular, we analyze the possibility to perform a key-recovery attack exploiting the complete codebook of PRESENT-128.

According to [6], an attack against 24 rounds requires between 2^{57} and 2^{60} texts and the data complexity increases by a factor of 2^3 for every additional round. This was experimentally confirmed in Figure 4 for up to 16 rounds. If we extrapolate these estimations for 7 more rounds, it amounts to a total of approximately $2^{60+3\cdot 7} = 2^{81}$ texts for 31 rounds, which is more than the whole codebook. However, using multiple trails, we can decrease this complexity by extracting more information from a reduced number of texts. For example, using $2^{81}/2^{64} = 2^{17}$ trails with similar distributions as the one in [6] with the whole codebook - and assuming that they give rise to independent information ! - should be enough to recover 48 bits of the key with a significant gain.

The time complexity of such an hypothetical attack would be 2^{64} memory accesses for each trail, meaning 2^{81} memory accesses for all the 2^{17} trails. It would additionally require 2^{67} bytes to store the codebook. This complexity is slightly higher than an exhaustive search for 80-bit keys, but is a significant improvement for a 128-bit key. Also, there is a possible time-memory tradeoff since one can avoid storing the codebook by re-generating it for each trail.

Again, it is important to emphasize that these complexities are optimistic compared to what would be observed if experiments could be launched with the full codebook. This is because they assume that multiple trails bring independent information. As experimented in the previous section, this is only correct up to a certain (for now, hard to quantify) extent. Hence, it is necessary to multiply our estimated time complexities by a constant factor (as it was done with the data complexities in [6]). These corrective terms should mainly incorporate two effects: first, the possible correlation between different mask and trails as mentioned in this section; second, the possible deterioration and key dependencies of the statistical biases of single trails when the number of rounds increases, due to the statistical hull effect that we detail in the next section. Since we do not have a sound theory to analyze this statistical hull effect, and its experimental evaluation beyond 16 rounds is computationally intensive, we can only conjecture that the combination of multiple trails can be used to trade data complexity for time complexity up to a certain level. Yet, it remains that multiple trails improve the previous results from CT-RSA 2009. And the approximated time complexity of the hypothetical attack (*i.e.* 2^{81}) is small enough compared to 2^{128}, so that there is a reasonable chance that it will remain faster than exhaustive key search, even after the introduction of these corrections. At least, these estimations raise interesting questions about the number of rounds in PRESENT-128.

5 Statistical hull effect

As detailed in Section 4.1, the theoretical data complexities computed following the transition matrices in Section 2 do not always correspond to our practical experiments. In this section, we underline one possible reason explaining this divergence, in relation with the assumption of uniform distributions for the bits that are not part of the trail. That is, while this assumption is nicely respected for the input plaintexts, it becomes incorrect as the number of rounds increases. In fact, this behavior can be related to the statistical hull effect that has been put forward in the context of linear cryptanalysis. A linear hull describes a set of linear approximations that share the same input and output masks, but have different trails and different biases. Consequently, each of these approximations contributes to the bias of the overall approximation [18]. This phenomenon explains why linear cryptanalysis can perform better than expected by the theoretical bias evaluated for a particular approximation. Differential cryptanalysis has a similar concept of differential that is made of several differential characteristics with the same input and output differences, *e.g.* described in [14].

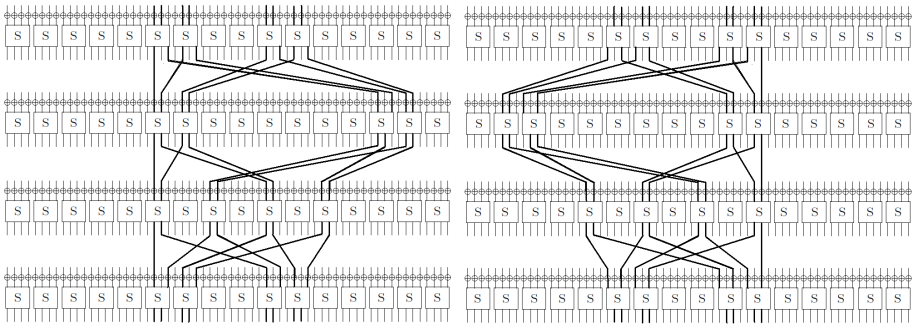

Fig. 7: 4-round trails with the same input and output masks.

In Statistical Saturation Attacks, an analogous phenomenon can also be observed. Namely, several trails with the same active input and output bits can be found, each of them having its own specific transition matrix. For example, two trails with the same input and output masks as the one of Figure 1 are given in Figure 7. By running our branch-and-bound algorithm, we could find numerous other trails corresponding to this input and output masks, as detailed in Table 1.

$\#rounds$	$\#trails$
2	1
3	5
4	54
5	1044

Table 1: Number of trails with the same input and output masks as in Figure 7.

The table directly suggests that the number of such trails increases exponentially with the number of rounds. For a large enough number of trails, it consequently becomes difficult to estimate their global effect on the distribution of the output bits. That is, as the trails may be correlated, the combined output distribution is not a simple combination of the theoretical output distributions of each trail. This observation can in fact be related to the work of Keliher *et al.* [13], in which the estimation of an upper bound for the linear hull effect was shown to be computationally hard in the number of rounds.

In the context of linear cryptanalysis, such experiments explain why, as the number of rounds increases, random masks can be almost as effective in recovering a key than a carefully selected trail (as witnessed, *e.g.* by Vaudenay's χ^2 cryptanalysis [21]). Strong hull effects may also imply key dependencies in the sense that the behavior of a trail for different keys may not be identical anymore (hence illustrating that the key equivalence hypothesis first discussed in [8] would not hold for PRESENT). In our (mainly experimental) setting, we conjecture that similar effects explain the deviations between the practical gain of trails having similar deviations from uniform under the assumptions of Section 2. However, we note that for a number of trails (*e.g.* the iterative one in Figure 1), these assumptions holds nicely. Analyzing the possible differences between these experimental observations and the ones made in the context of a linear cryptanalysis in another interesting scope for further research.

6 Conclusion and further works

A summary of the published cryptanalysis results against PRESENT is given in Table 2. It shows that Statistical Saturation Attacks outperform other types of cryptanalyses (in particular linear and differential) against this cipher. This is due to the design of its permutation layer. The main outcome of this paper is to show that the use of multiple trails allows improving the previous result of CT-RSA 2009. Also, if the assumptions in this paper are verified for larger number of rounds, the use of multiple-trails could lead to attacks with smaller time complexity than exhaustive key search against the full PRESENT-128.

As discussed in the previous sections, these estimations have to be considered with care, which is made explicit with the constant multiplicative factor c that we give for the time complexities in the table. This situation is similar to the one in linear cryptanalysis, where the precise estimation of the complexities is made difficult by the large cardinality of the trails to investigate.

In fact, the situation in the present paper is even more difficult, since we have to deal with complete distributions rather than scalar bias values. Positively, the experimental attacks that we performed against reduced number of rounds confirm our theoretical estimations to a reasonable extent. They at least show a significant improvement of the attacks when using multiple trails.

#rounds	Attack	Data compl.	Time compl.	Memory compl.	Ref.
16	SSA	$c * 2^{36}$CP	2^{28} MA	2^{16} counters	[6]
16	DC	2^{64}CP	2^{65} MA	$6 * 2^{32}$ bits	[23]
17	RKR	2^{63}CP	2^{104} MA	2^{53} counters	[20]
24	*SSA*	$c * 2^{60} CP$	$2^{28} MA$	2^{16} *counters*	[6]
26	LH	2^{64}KP	$2^{98.7}$ MA	2^{40} counters	[17]
26	MLC	2^{64}KP	2^{72} MA	2^{34} bytes	[10]
27	*MT-SSA*	$2^{64} CP$	$c \cdot 2^{69} MA$	2^{67} *bytes*	This paper
29	*MT-SSA*	$2^{64} CP$	$c \cdot 2^{75} MA$	2^{67} *bytes*	This paper
31	*MT-SSA*	$2^{64} CP$	$c \cdot 2^{81} MA$	2^{67} *bytes*	This paper

CP-Chosen Plaintext, KP-Known Plaintext, MA-Memory Access
DC-Differential Cryptanalysis, SSA-Statistical Saturation Attack, RKR-Related Key Rectangle, MLC-Multidimensional Linear Cryptanalysis, LH-Linear Hull

Table 2: Summary of attacks (italic are not experimented and use **ext. 2**, **ext. 4**.).

While these results do not threaten the practical applications of PRESENT (especially since it is mainly its 80-bit version that was advertised in [3]), they raise interesting open questions. For example, they make a case for designing efficient ciphers in which all the statistical effects that can be exploited in cryptanalysis are taken into account. The decorrelation theory appears as an interesting alternative in this respect [22]. But most importantly, the present experimental work implies the need of a better understanding of the statistical saturation attack and its extensions (in particular, **ext. 2**, *i.e.* using multiple fixed values in the trails, and **ext. 4**, *i.e.* using multiple trails). This implies providing sound explanations for the statistical hull and correlation effects between masks and trails, informally described in this paper. The similarities of our results with recent works in multidimensional cryptanalysis [9] also need to be investigated.

References

1. T. Baignères, P. Junod, S. Vaudenay, *How Far Can We Go Beyond Linear Cryptanalysis?*, in the proceedings of ASIACRYPT 2004, Lecture Notes in Computer Science, vol 3329, pp 432-450, Jeju Island, Korea, December 2004.
2. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of CRYPTO 2004, Lecture Notes in Computer Science, vol 3152, pp 1-22, Santa Barbara, California, USA, August 2004.
3. A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, C. Vikkelsoe, *PRESENT: An Ultra-Lightweight Block Cipher*, in the proceedings of CHES 2007, Lecture Notes in Computer Science, vol 4727, pp 450-466, Vienna, Austria, September 2007.
4. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improving the Time Complexity of Matsui's Linear Cryptanalysis*, in the proceedings of The International Conference on Information Security and Cryptology - ICISC 2007, Lecture Notes in Computer Science, vol 4817, pp 77-88, Seoul, Korea, November 2007.
5. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent*, Fast Software Encryption 2008, Lecture Notes in Computer Science, vol 5086, pages 382-397, Springer, February 2008.

6. B. Collard, and F.-X. Standaert, *A Statistical Saturation Attack on the Block Cipher PRESENT*, in the proceedings of CT-RSA 2009, Lecture Notes in Computer Science, vol 5473, pages 195-210, San Francisco, California, USA, April 2009.
7. B. Collard, and F.-X. Standaert, *A Statistical Saturation Attack on the Block Cipher PRESENT, Errata and Improvement*, available for download from: http://www.dice.ucl.ac.be/ fstandae/PUBLIS/62b.pdf
8. C. Harpes, G. Kramer, J. Massey, *A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-Up Lemma*, in the proceedings of EUROCRYPT 1995, LNCS, vol 921 , pp 24-38, Saint-Malo, France, May 1995.
9. M. Hermelin, J.Y. Cho, K. Nyberg, *Multidimensional Extension of Matsui's Algorithm 2*, in the proceedings of FSE 2009, Lecture Notes in Computer Science, vol 5665, pp 209-227, Leuven, Belgium, February 2009.
10. Joo Yeon Cho, *Linear Cryptanalysis of Reduced-Round PRESENT*, Cryptology ePrint Archive: Report 2009/397, available on: http://eprint.iacr.org/2009/397.
11. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Sciences, vol 839, pp 26-39, Santa Barbara, California, USA, August 1994.
12. G. Leander, *Small Scale Variants of the Block Cipher PRESENT*, IACR ePrint Archive, http://eprint.iacr.org/2010/143.
13. L. Keliher, H. Meijer, S.E. Tavares, *New Method for Upper Bounding the Maximum Average Linear Hull Probability for SPNs*, proceedings of EUROCRYPT 2001, Lecture Notes in Computer Science, vol 2045, pp 420-436, Innsbruck, Austria, May 2001.
14. X. Lai, J.L. Massey, and S. Murphy, *Markov Ciphers and Differential Cryptanalysis*, Advances in Cryptology - EUROCRYPT 1991, Lecture Notes in Computer Science, vol 547, pp. 17-38, Brighton, United Kingdom, April 1991.
15. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of EUROCRYPT 1993, LNCS, vol 765, pp 386-397, Lofthus, Norway, May 1993.
16. M. Minier, H. Gilbert, *Stochastic Cryptanalysis of Crypton*, in the proceedings of Fast Software Encryption 2000, Lecture Notes in Computer Science, vol 1978, pp 121-133, New York, USA, April 2000.
17. J. Nakahara Jr, P. Seperhdad, B. Zhang, M. Wang, *Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT*, to appear in the proceedings of CANS 2009, Kanazawa, Japan, December 2009.
18. K. Nyberg, *Linear Approximation of Block Ciphers*, proceedings of EUROCRYPT 1994, LNCS, vol 950, pp 439-444, Perugia, Italy, May 1994.
19. K. Ohkuma, *Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis*, in the proceedings of SAC 2009, Lecture Notes in Computer Science, vol 5867, pp 249-265, Calgary, Alberta, Canada, August 2009.
20. O. Özen, K. Varici, C. Tezcan, *Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT*, proceedings of ACISP 2009, Lecture Notes in Computer Science, vol 5594, pp 90-107, Brisbane, Australia, July 2009.
21. S. Vaudenay, *An Experiment on DES Statistical Cryptanalysis*, In the proceedings of the third ACM Conference on Computer and Communications Security (CCS 1996), ACM, pp. 139-147, New Delhi, India, March 1996.
22. S. Vaudenay, *Decorrelation: A Theory for Block Cipher Security*, Journal of Cryptology, vol 16, num 4, pp 249-286, Springer, 2003.
23. M. Wang, *Differential Cryptanalysis of Reduced-Round PRESENT*, in the proceedings of AFRICACRYPT 2008, Lecture Notes in Computer Science, vol 5023, pp 40-49, Casablanca, Morocco, June 2008.