# Experiments on the Multiple Linear Cryptanalysis of Reduced Round Serpent

B. Collard, F.-X. Standaert[⋆], J.-J. Quisquater
UCL Crypto Group, Microelectronics Laboratory, Louvain-la-Neuve, Belgium

**Abstract.** In 2004, Biryukov *et al.* presented a new theoretical framework for the linear cryptanalysis of block ciphers using multiple approximations. Although they provided first experimental results to confirm the relevance of their approach, a scope for further research was to apply this framework to other ciphers. In this paper, we present various attacks against reduced-round versions of the AES candidate Serpent. Our results illustrate that the hypotheses of Crypto 2004 hold (at least) as long as the number of approximations exploited in the linear attack are computationally tractable. But they also underline the limits and specificities of Matsui's *algorithms* 1 and 2 for the exploitation of such approximations. In particular, they show that the optimal application of *algorithm 2* requires good theoretical estimations of the approximation biases, which may be a problem when the linear hull effect is non-negligible. These results finally confirm the significant reductions of the attacks data complexity that can be obtained from multiple linear approximations.

## 1 Introduction

The linear cryptanalysis [10] is one of the most powerful attacks against modern block ciphers in which an adversary exploits a linear approximation of the type:

$$P[\chi_P] \oplus C[\chi_C] = K[\chi_K] \tag{1}$$

In this expression, $P$, $C$ and $K$ respectively denote the plaintext, ciphertext and the expanded key while $A[\chi]$ stands for $A_{a_1} \oplus A_{a_2} \oplus \cdots \oplus A_{a_n}$, with $A_{a_1}, \cdots, A_{a_n}$ representing particular bits of $A$ in positions $a_1, \cdots, a_n$ ($\chi$ is usually denoted as a mask). In practice, linear approximations of block ciphers can be obtained by the concatenation of one-round approximations and such concatenations (also called characteristics) are mainly interesting if they maximize the deviation (or bias) $\epsilon = p - \frac{1}{2}$ (where $p$ is the probability of a given linear approximation).

In its original paper, Matsui described two methods for exploiting the linear approximations of a block cipher, respectively denoted as *algorithms* 1 and 2. In the first one, given an $r$-round linear approximation with sufficient bias, the algorithm simply counts the number of times $T$ the left side of Equation 1 is equal to zero for $N$ pairs (plaintext, ciphertext). If $T > N/2$, then it assumes either $K[\chi_K] = 0$ if $\epsilon > 0$ or $K[\chi_K] = 1$ if $\epsilon < 0$ so that the experimental value $(T - N/2)/N$ matches the theoretical bias. If $T < N/2$, an opposite reasoning holds. For the attack to be successful, it is shown in [10] that the number of available (plaintext, ciphertext)-pairs must be proportional to $\frac{1}{\epsilon^2}$.

[⋆] Postdoctoral researcher of the Belgian Fund for Scientific Research (FNRS).

In the second method, a $r$-1-round characteristic is used and a partial decryption of the last round is performed by guessing the key bits involved in the approximation. As a consequence, all the guessed key bits can be recovered rather than the parity $K[\chi_K]$ which yields much more efficient attacks in practice. In addition, since it exploits a $r$-1-round approximation rather than a $r$-round one, it also takes advantage of a better bias which reduces the data complexity.

Among the various proposals to improve the linear cryptanalysis of block ciphers, Kaliski and Robshaw proposed in 1994 an algorithm using several linear approximations [7]. However, their method imposed a strict constraint as it requires to use only approximations implying the same bits of subkeys $K[\chi_K]$. This restricted at the same time the number and the quality of the approximations available. As a consequence, an approach removing this constraint was proposed in 2004 [2] that can be explained as follows. Let us suppose that one has access to $m$ approximations on $r$ block cipher rounds of the form:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \ (1 \leq i \leq m), \tag{2}$$

and wishes to determine the value of the binary vector of parity:

$$\mathbf{Z} = (z_1, z_2, ..., z_m) = (K[\chi_K^1], K[\chi_K^2], ..., K[\chi_K^m]) \tag{3}$$

The improved algorithm associates a counter $T_i$ with each approximation, that is incremented each time the corresponding linear approximation is verified for a particular pair (plaintext-ciphertext). As for *algorithm* 1, the values of $K[\chi_K^i]$ are determined from the experimental bias $(T_i - N/2)/N$ and the theoretical bias $\epsilon_i$ by means of a maximum likelihood rule. The extension of *algorithm* 2 to multiple approximations is similarly described in [2].

An important consequence of this work is that the theoretical data complexity of the generalized multiple linear cryptanalysis is decreased compared to the original attack. According to the authors of [2], the attack requires a number of texts inversely proportional to the capacity of the system of equations used by the adversary that is defined as: $c = 4 \cdot \sum_{i=1}^{m} \epsilon_i^2$. Therefore, by increasing this quantity (*i.e.* using more approximations), one can decrease the number of plaintext/ciphertext pairs necessary to perform a successful key recovery.

In this paper, we aim to apply the previously described cryptanalytic tools to the Advanced Encryption Standard (AES) candidate Serpent [1] in order to confirm the analysis of Crypto 2004 and put forward a number of intuitive facts about its implementation. Our results confirm the significant reductions of the attacks data complexity that can be obtained from multiple linear approximations but also their computational limitations. From a more theoretical point of view, multiple linear cryptanalysis is also the best understood technique to take advantage of the linear hull effect [11]. Therefore it allows to fill the gap between the practical [8] and provable security approaches for block ciphers.

Finally, our results underline the specificities of Matsui's *algorithms* 1 and 2 for the exploitation of multiple approximations. In particular, they show that the optimal application of *algorithm 2* requires good theoretical estimations of the approximation biases, which may be a problem when the linear hull effect is non-negligible. As a consequence, sub-optimal strategies sometimes have to be applied. By contrast, our application of *algorithm* 1 nicely follows the predictions of [2], even if the approximation biases are underestimated.

The rest of the paper is structured as follows. Section 2 refers to our linear approximation search algorithm. Section 3 describes preliminary observations on the linear cryptanalysis of Serpent and highlights the existence of a linear hull effect. Sections 4 and 5 respectively provide the experimental results of our attacks against reduced-round Serpent, using *algorithms* 1 and 2. Finally, conclusions are in Section 6 and a description of the Serpent cipher is in Appendix A.

## 2 Linear approximations search

The first step in a linear cryptanalysis attack consists in finding linear approximations of the cipher with biases as high as possible. But the problem of searching such approximations is not trivial, because of the great cardinality of the set of candidates. In 1994, Matui proposed a branch-and-bound algorithm making possible to effectively find the best approximation of the DES [12]. However, for practical reasons that are out of the scope of this paper, this method hardly applies to block ciphers with good diffusion such as the AES candidates. As a consequence, we rely on approximations found with a modified heuristic that is described in [3]. Although it does not ensure to obtain the best approximations of Serpent, it provided the best-reported ones in the open literature. Note that, given a $r$-round approximation found with the branch-and-bound algorithm, the first round masks and last round masks can be replaced by any other mask provided that the biases are left unchanged. Due to the properties of the Serpent S-boxes, several similar approximations can easily be generated with this technique. This allowed us to obtain large sets of approximations involving the same key bits to guess at the cost of dependencies in the linear trails[1].

## 3 Preliminary observations

Prior to the investigation of multiple linear approximations, we performed experiments with single approximations. We started with *algorithm 1* of which the principle is as follows. For each plaintext-ciphertext pair, we evaluate the left part of equation 1 and increment or decrement a counter given the result. This way, the counter can be used to evaluate the experimental bias of the approximation. If the experimental and theoretical biases have the same sign, then we can presume that the parity of the subkey bits in the right part of Equation 1 is zero. Otherwise, we guess that this parity is one, so that empirical and theoretical results match. As it is suggested in [10], this heuristic relies on a maximum likelihood approach in which we choose the parity so that theoretical and practical results fit well. Thus, the unknown parity can be guessed with an arbitrary high

[1] A linear trail is a set of $r + 1$ masks describing a r-round approximation [11]

probability by computing the experimental mean of the bias and choosing the parity that minimizes the distance between theory and practice. As an illustration, we used a 4-round linear approximation of Serpent with a theoretical bias of $2^{-12}$ and observed its experimental value for a number of plaintext-ciphertext pairs proportional to $2^{24}$. Figure 1 illustrates that the bias value becomes stable after approximately $8/\epsilon^2$ encrypted pairs. It also shows that the theoretical bias (provided by the branch-and-bound algorithm in [3]) was underestimated, which suggests that the linear hull effect is not negligible in our experiments [11]: there are several approximations with the same input/output mask that contribute to the bias in a non negligible way. This effect can cause the complexity of a linear cryptanalysis attack to be overestimated [6].

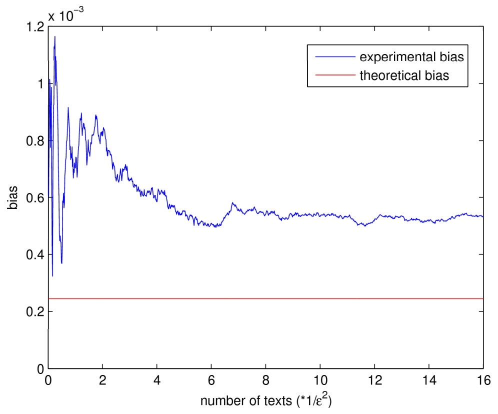

Fig. 1: Evolution of the experimental bias *w.r.t.* the number of known-plaintexts used.

Next to this first experiment, we observed the behavior of 64 linear approximations with various biases, as illustrated in Figure 2. It shows that, provided a sufficient number of encrypted plaintexts, the approximations separate in two classes: the ones with positive bias and the ones with negative bias. This experiment suggests the interest of exploiting multiple linear approximations: since any of these experimental biases provide the adversary with some information on the block cipher key, it is worth trying to exploit them in an efficient way.

Following Kaliski and Robshaw [7], Biryukov *et al.* proposed a general approach to extend Matsui's linear cryptanalysis to multiple linear approximations [2]. As in the simple approximation case, an experimental bias is derived for each approximation in a distillation phase during which counters are extracted from the data. Then, in the analysis phase, an euclidian distance between the theoretical prediction and the experiment is evaluated for each possible parities of the key bits involved in the approximations. The parity minimizing this distance is guessed to be the correct one. The expectation is that the number of encrypted plaintexts required to achieve a given success rate can be reduced when the number of approximation is increased, according to the value of the capacity $c = 4 \cdot \sum_{i=1}^{m} \epsilon_i^2$. In the next sections, we experimentally evaluate the extent to which these expectations can be fulfilled, both for Matsui's *algorithm* 1 and 2.

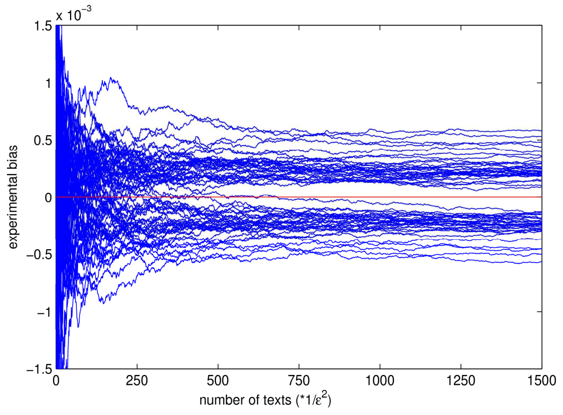

Fig. 2: Evolution of 64 experimental biases *w.r.t.* the number of known-plaintexts used.

## 4 Experimental attacks with *algorithm 1*

### 4.1 Selection of the approximations

A significant drawback of Matsui's *algorithm* 1 compared to the second one is that the adversary does not recover master key bits, but a linear equation involving key bits in all the cipher rounds. In the context of non-linear key-scheduling algorithms, this makes the practical exploitation of the attack results difficult since it does not straightforwardly reduce the complexity of an exhaustive key search. When using multiple linear approximations, this drawback can be partially relaxed in the following way[2]. First, the best approximation provided by the branch-and-bound algorithm is selected. Then, only the input/output masks are modified in order to generate large sets of equations. Finally, the adversary progressively increases the size of its system of equations: each time he adds an equation to the system, he also checks the rank $r$ of the corresponding matrix, indicating the number of linearly independent relations in his system. By choosing the system of equations such that the independencies between the equations only relate to meaningful key bits, the adversary ends up with an exploitable information on the cipher key. For example, if the adversary only modifies the input masks to generate a system of the form:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \ (1 \leq i \leq m), \tag{4}$$

he can recover first round key bits. Only one bit (corresponding to the non-variable part of the trail in the system) has to be guessed additionally. As an illustration, we performed attacks against 4-rounds of Serpent, using 64 approximations such that the resulting system of equations has rank $r = 10$.

[2] Of course, it remains that *algorithm* 1 uses a $r$-round approximation compared to a $r-1$-round approximation in *algorithm* 2 which increases its data complexity.

### 4.2 Attacks results

Figure 3 depicts the evolution of the distance between the theoretical and experimental biases, for various values of the parity guess and number of encrypted plaintexts. The correct key candidate is expected to minimize this distance which is verified in practice: $32/c$ encrypted plaintexts are sufficient to uniquely determine the correct parity guess. As expected, increasing the number of encrypted plaintexts improves the confidence (or reduces the noise) in the attack result. For example, Figure 4 illustrates the result of a similar attack when $4096/c$ encrypted plaintexts are provided to the adversary. Interestingly, the attack result has a very regular structure underlining the impact of the Hamming distance between different key candidates: close keys have close biases. However, such figure does not tell us (or quantify) how much the exploitation of multiple approximations allowed reducing the attack data complexity.

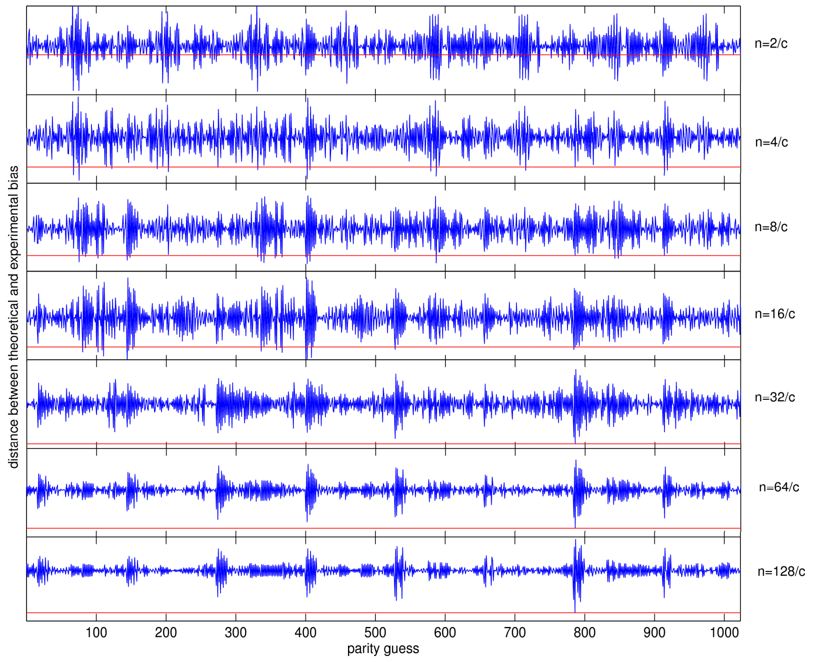

Fig. 3: Evolution of the distance between the theoretical and experimental biases *w.r.t.* the parity guess when the number of encrypted plaintexts increases (64 4-rounds approximations with a capacity of $5.25 \cdot 10^{-6}$). The horizontal line in each graph indicates this distance for the correct parity guess. The scale is the same in each figure.

For this purposes, we ran another set of experiments in which we computed the gain of the attack. As defined in [2], *if an attack is used to recover an $n$-bit key and is expected to return the correct key after having checked on the average $M$ candidates, then the gain of the attack, expressed in bits, is defined as:*

$$\gamma = -log_2 \frac{2 \cdot M - 1}{2^n} \tag{5}$$

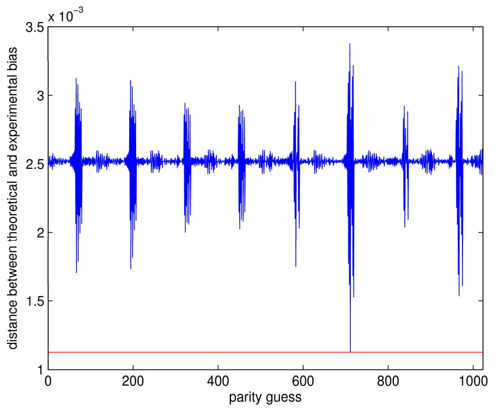

Fig. 4: Evolution of the distance between the theoretical and experimental biases (same as Figure 3) *w.r.t.* the parity guess when using $4096/c$ encrypted plaintexts.

Intuitively, it is a measure of the remaining workload (or number of key candidates to test) after a cryptanalysis has been performed. In the context of multiple linear cryptanalysis attacks, the gain is simply determined by the position of the correct key (or parity) candidate in the weighted list of candidates obtained from the analysis phase. For example, if an attack is used to recover 8 key bits and the correct key candidate is the most likely (*resp.* second most likely), it has a gain of 8 bits (*resp.* 6.42 bits). Importantly, when *algorithm* 1 is used, the maximum gain of an attack depends on the rank of its systems of equations.

In Figure 5, the gain of three attacks are given with respect to the data complexity. The first attack (in red) recovers one bit of parity using only one approximation (*i.e.* it is a simple linear cryptanalysis). The second attack (in green) uses 10 approximations and recovers up to 10 parity bits, while the third attack (in blue) recovers 10 parity bits using 64 linearly dependent approximations. As expected, the gain obtained using 64 approximations increases about 8 times faster than with 10 approximations. This example shows the flexibility of-

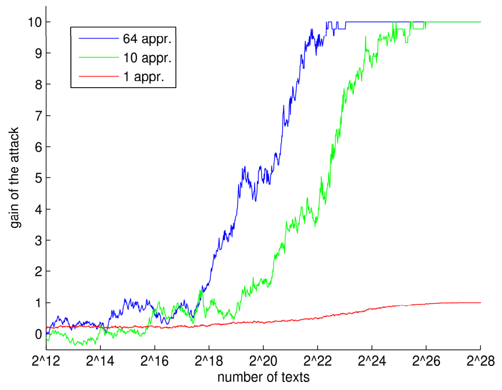

Fig. 5: Evolution of the gain with respect to the number of encrypted plaintexts.

fered by multiple approximations when *algorithm* 1 is applied: they can be used both to get a better gain for a fixed number of plaintext or to get a lower data complexity for a fixed gain. This observation is even better quantified in Figure 6 where the evolution of the gain is given according to the number of encrypted plaintexts normalized by the capacity (*i.e.* the number of plaintexs divided by the joint capacity of the approximations used in the attack). It clearly illustrates the tradeoff between attack complexity and gain. It also confirms that $N \propto 1/c$ is the number of plaintexts required for the attack to reach its maximum gain.

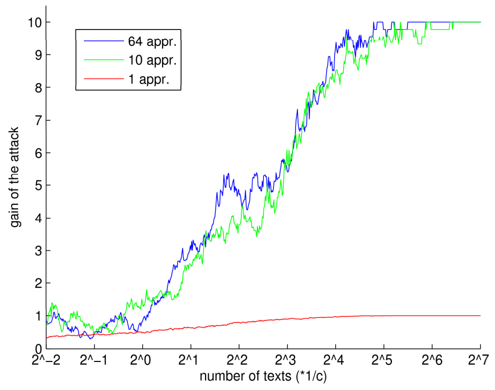

Fig. 6: Evolution of the gain *w.r.t.* the number of plaintexts normalized by the capacity.

### 4.3 Gain versus success rate and further insights

Let us introduce the following definition:

**Definition 1 (success rate).** *The success rate of an attack using $n$ approximations (for a given number of plaintexts/ciphertexts) is the number of parity bits guessed correctly among the $n$ parities derived from the distance between the experimental and theoretical bias values.*

Figure 7 represents the evolution of the gain and of the success rate when the number of encrypted plaintexts increases. Interestingly, we can see that the gain increases much faster than the success rate. For example, after about $2^{23}$ encrypted plaintexts, the gain of the attack reaches its maximum, while the success rate only equals 0.8 at this point. This is a consequence of the linear dependancies between the approximations. Suppose we are given $m$ linear approximations:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \ (1 \le i \le m), \tag{6}$$

Such that the following relation holds (case of dependant text masks):

$$\chi_P^1 \oplus \chi_P^2 \oplus ... \oplus \chi_P^m = \chi_C^1 \oplus \chi_C^2 \oplus ... \oplus \chi_C^m \tag{7}$$

Approximation $m$ is linearly dependant in the sense that no additionnal information is given in the deterministic case (i.e. if the approximations hold with probability 1). However, in the probabilistic case, some information can still be extracted (as shown in [13]), as the bias of the $m-th$ approximation is not necessarily related to the bias of the $m-1$ first. When performing multiple linear approximations cryptanalysis, the left part of each approximation is evaluated for a large number of plaintext-ciphertexts pairs and then the parity of the right part (involving subkey bits) is choosen so as to minimize the distance between experimental and theoretical bias. However, some of the parity guess might be wrong in which case the system of equations (where $p^i$ is the parity guess for approximation $i$):

$$\begin{cases} K[\chi_K^1] = p^1 \\ K[\chi_K^2] = p^2 \\ \quad \ldots \\ K[\chi_K^m] = p^m \end{cases}$$

can be inconsistent given the linear dependancies, and one or more parity guess must be changed. This can be verified only if there are linear dependancies between the approximations. As the consistency check can be performed before the exhaustive search for the remaining unknown bits, this increases the gain of the attack. Intuitively, this shows that using more approximations than the rank of the system in an attack provides an effect similar to an error correcting code: some parity candidates can be rejected a-priori. By contrast, the success rate of an attack is expected to remain the same when the number of approximations increases (provided that the theoretical biases of each approximation are equal).

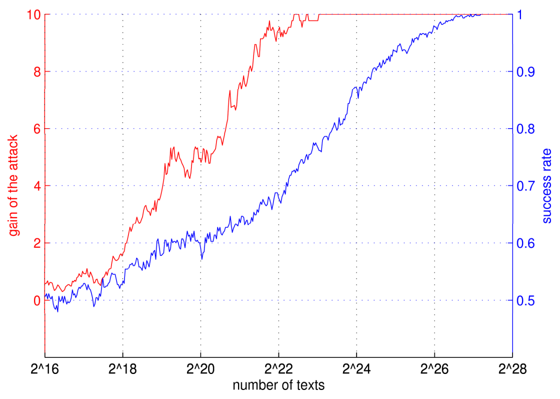

Fig. 7: Evolution of the gain and success rate *w.r.t.* the number of encrypted plaintexts.

As an illustration, we can study the relation between the success rate and the gain of an attack. Suppose that at least $n'$ out of the $n$ approximation parity are guessed correctly. Obviously, the succes rate is higher than $n'/n$. In order to recover the key (and evaluate the gain), we must generate a list of candidates from the value of the parity bits and then try each candidate until the correct one is found. This can be done using the following strategy:

- Choose the first candidate so as to minimize the euclidian distance between theoretical and experimental bias.
- Assume one guess is incorrect; choose one parity bit and take its complement; try the $\binom{n}{1}$ possible candidates;
- Assume two guesses are incorrect; choose two parity bits and take their complements; try the $\binom{n}{2}$ possible candidates;
- ...
- Assume $n - n'$ guesses are incorrect; choose $n - n'$ parity bits and take their complements; try the $\binom{n}{n-n'}$ possible candidates;

After $n - n'$ steps, we have necessarily found the correct candidate as there is maximum $n - n'$ wrong guesses, thus the gain of the attack equals:

$$\gamma = -log_2\left(\frac{\sum_{i=0}^{n-n'} \binom{n}{i}}{2^n}\right) \tag{8}$$

In this equation, we implicitly assume the independence between the approximations (*i.e.* we assume the maximum gain can be $n$). However, experiments using up to 416 approximations (including 15 linearly independent ones) show that this prediction fits reasonably well even when the approximation are not independent, as long as the gain does not saturate to its maximum value (see Figure 8). We observe that for a given success rate, the gain of the attack increases with the number of approximations. This example highlights (from a different point of view) the advantage of multiple linear approximations compared to single linear cryptanalysis in which the success rate is equivalent to the gain.

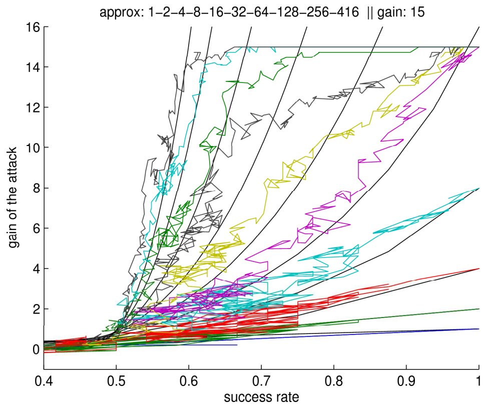

Fig. 8: Evolution of the gain *w.r.t.* the success rate for various number of approximations. This number ranges from 1 to 416 when moving from the bottom curves to the top curves. The black smooth curves are the theoretical predictions.

## 5 Experimental attacks with *algorithm 2*

In this section, we perform experimental attacks against 5-round Serpent using multiple linear approximations with *Algorithm 2*. It allows the adversary to recover subkey bits in the first/last round of the cipher as follows. An attack against $r$ cipher rounds deals with a linear approximation of the $r-1$ last/first rounds. For each plaintext-ciphertext pair and subkey candidate, the ciphertext is then partially en/decrypted with the subkey candidate, and the approximation is evaluated for the partial en/decryption. A counter indexed by the keyguess value is incremented/decremented according to the parity of the evaluation. For a wrong candidate, the partial en/decryption is expected to produce a random output, thus leading to an null experimental bias (meaning that its statistical evaluation is sufficiently close to zero). For the correct key guess, the experimental bias is expected to converge toward the theoretical bias. In order to speed-up the computations, we used the FFT trick proposed in [4].

### 5.1 Differences between *algorithms* 1 and 2

Compared with *algorithm 1*, the exploitation of multiple linear approximations with *algorithm* 2 faces an additional problem that we detail in this section. In multiple linear cryptanalysis, an adversary has a system of $m$ linear approximations. Each approximation has a theoretical value for its bias $\epsilon_i$ and the adversary additionally obtains an experimental value for this bias $\epsilon_i^*$. When *algorithm* 1 is used, this experimental value is used to minimize the Euclidean distance:

$$\min_g \sum_{i=1}^{m} (\epsilon_i - (-1)^{g(i)} \cdot \epsilon_i^*)^2, \tag{9}$$

where $g$ is the parity guess of the linear approximations. But when *algorithm* 2 is used, this experimental bias also depends on the round subkey that is used to perform the first (or last) round partial decryption: $\epsilon_{i,k}^*$. Therefore, the following Euclidean distance has to be computed:

$$\min_k \big( \min_g \sum_{i=1}^{m} (\epsilon_i - (-1)^{g(i)} \cdot \epsilon_{i,k}^*)^2 \big) \tag{10}$$

The practical consequence of these different conditions can be explained as follows. While the condition in Equation 9 leads to a correct value for the guess, even if the theoretical bias values are underestimated, the condition in Equation 10 only leads to a correct key candidate if a good theoretical estimation of these biases is available. This is due to the key dependencies of the experimental biases in *algorithm* 2. Unfortunately, in the context of the Serpent algorithm investigated in this paper, it has been shown in Section 3 that these theoretical values are not accurate, due to the linear hull effect. As a consequence, the framework of [2] cannot be straightforwardly applied in our context. This is in contrast, *e.g.* with the DES, where such a linear hull effect is negligible [6].

### 5.2 Attack results

The usual solution to overcome this problem is to look at the maximum experimental bias values. For example, when multiple approximations are considered, a vector of experimental biases can be derived for each approximation. Then the average is taken over all the approximations and its maximum value is expected to indicate the correct key candidate. But the theoretical framework of Crypto 2004 is not applied anymore and such an approach is more closely related to the experiments of Kaliski and Robshaw [7]. As an illustration, Figure 9 illustrates the evolution of the experimental biases averaged over 32 4-rounds approximations, for different numbers of plaintext/ciphertext pairs and 12 bits of keyguess. Figure 10 shows a comparison between simple and multiple linear cryptanalysis. This latter picture illustrates that multiple linear approximations still allow reducing the noise level in the modified attacks which improves their efficiency.

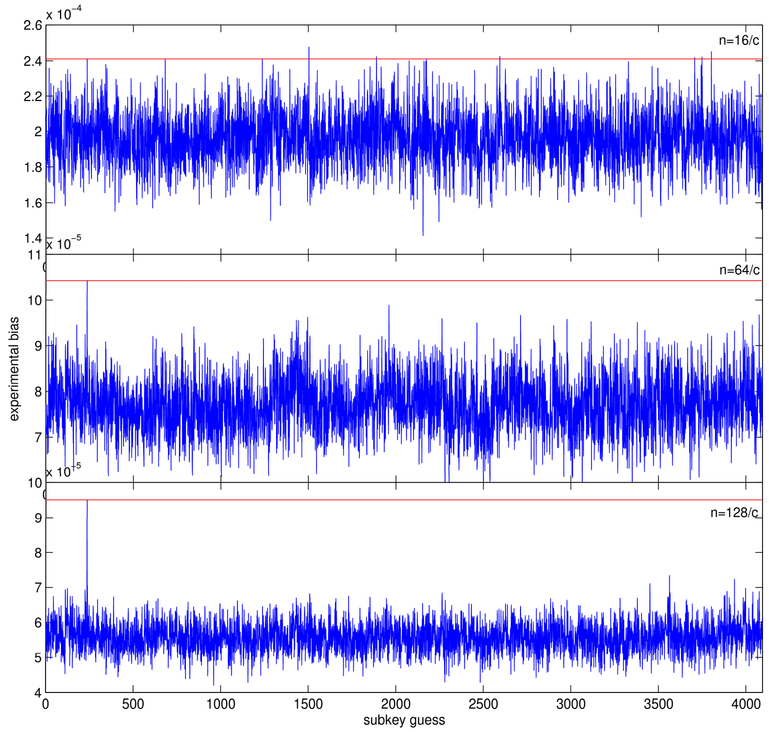

Fig. 9: Evolution of the experimental bias averaged over 32 4-rounds approximations for each guess of the $2^{12}$ key guesses, when the number of pairs increases ($c = 2.08 \cdot 10^{-7}$).

On the other hand, increasing the number of linear approximations does not involve reductions of the data complexity according to the capacity values as in the previous section. This is caused by the modified analysis phase, in which

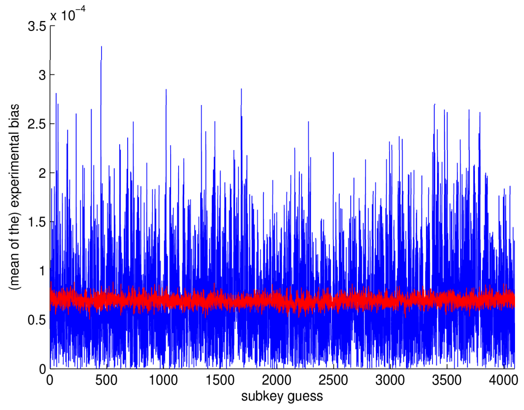

Fig. 10: Comparison of the noise levels between simple and multiple linear cryptanalysis (104 approximations vs. 1 approximation, n=64/c).

the exploitation of the information provided by the different approximations is not optimal anymore. This fact can be emphasized by investigating the gain of the attacks, for different number of approximations. For example, Figure 11 illustrates how the gain of three attacks with respectively 2, 8 and 32 approximations (having the same individual biases) increases. But the benefits are not as spectacular is in the context of *algorithm 1*. Namely, the 32-approximations gain is not increasing 16 times faster than when 2 approximations are used. Compared to the attack results with *algorithm* 1 (*e.g.* in Figure 4), it is finally interesting to notice that Figures 9 and 10 show no particular structure in the biases distribution. This is due to the partial en/decryption of one round that cancels the effects caused by close keys in the Hamming distance sense.

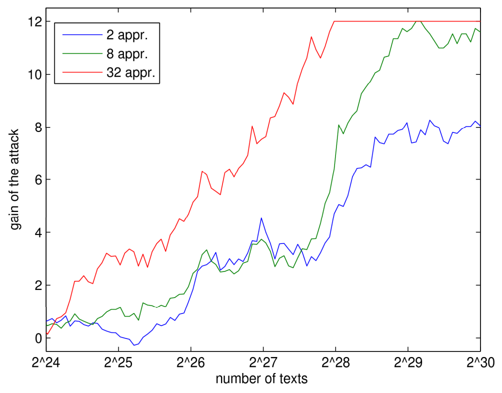

Fig. 11: Evolution of the gain of *algorithm 2* w.r.t. the number of encrypted plaintexts.

## 6 Conclusion and further works

This paper presented experimental results of multiple linear cryptanalysis attacks against reduced-round versions of the block cipher Serpent. It allowed us to highlight the following observations:

1. The hypotheses stated in [2] about the possible influence of dependencies (between the masks or linear trails) generally appear to be reasonably fulfilled, even for approximations of which a large part of the trail is identical.
2. Our experiments only considered a limited number of approximations. Dependencies effects could appear with more approximations. Note that the number of exploitable approximations is limited anyway, for computational reasons: because of the approximation matrix rank with *algorithm* 1 and because of the amount of partial en/decryptions to perform in *algorithm* 2.
3. By contrast with previous experiments against the DES, we observed a significant linear hull effect, with the following consequences:
   (a) Optimal attacks using Matsui's *algorithm* 1 closely followed the data complexities predicted with the capacity value (defined in [2]), even if the theoretical values of the approximation biases were underestimated.
   (b) Optimal attacks using Matsui's *algorithm* 2 did not lead to successful key recoveries because of the lack of good theoretical estimations of the bias values. Modified heuristics allowed us to take advantage of multiple approximations. But the improvement of the modified attack complexity is not following the predictions of the capacity values.
4. More generally, the analysis of Crypto 2004 leads to meaningful results as long as the branch-and-bound algorithm used to derive the linear approximations provides the adversary with the best possible biases.

In practice, our experiments finally confirmed the significant improvement of multiple linear cryptanalysis attacks compared to Matsui's original attack. Open questions include the optimal exploitation of multiple approximations using *algorithm* 2 when good estimations of the bias values are not available or the extension of these experiments towards more cipher rounds, *e.g.* using [9].

## Description of the approximations

A detailed description of the linear approximations used in our experiments is available at the following address:
http://www.dice.ucl.ac.be/ fstandae/PUBLIS/50b.zip

## Acknowledgements

This work is partially supported by the Walloon Region under the project Nanotic-Cosmos and by the Belgian Interuniversity Attraction Pole P6/26 BCRYPT.

## References

1. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, in the proceedings of the First Advanced Encryption Standard (AES) Conference, Ventura, CA, 1998.
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. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent*, in the proceedings of InsCrypt 2007, LNCS, pp. 47-61, Xining, China, September 2007.
4. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improving the Time Complexity of Matsui's Linear Cryptanalysis*, In K.-H. Nam and G. Rhee, editor(s), The International Conference on Information Security and Cryptology - ICISC 2007, Volume 4817 of Lecture Notes in Computer Science, pages 77-88, Springer, November 2007.
5. J. Daemen, V. Rijmen, *The Wide-Trail Strategy*, in the proceedings of IMA 2001, LNCS, vol. 2260, pp. 222-238, Cirencester, UK, December 2001.
6. P. Junod, *On the Complexity of Matsui's Attack*, in the proceedings of SAC 2001, LNCS, vol. 2259, pp. 199-211, Toronto, Ontario, Canada, August 2001.
7. 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.
8. L.R. Knudsen, *Practically Secure Feistel Ciphers*, in the proceedings of FSE 1993, LNCS, vol. 809, pp. 211-221, Cambridge, UK, December 1993.
9. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, M. Schimmler, *Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker*, in the proceedings of Cryptographic Hardware and Embedded Systems - CHES 2006, Lecture Notes in Computer Science, vol. 4249, Springer, 2006.
10. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of Eurocrypt 1993, LNCS, vol. 765, pp. 386–397, Lofthus, Norway, May 1993.
11. K. Nyberg, *Linear Approximations of Block Ciphers*, in the proceedings of Eurocrypt 1994, LNCS, vol. 950, pp. 439-444, Perugia, Italy, May 1994.
12. M. Matsui, *On Correlation Between the Order of S-boxes and the Strength of DES*, in the proceedings of Eurocrypt 1994, Lecture Notes in Computer Science, vol. 950, pp. 366-375, Perugia, Italy, May 1994.
13. S. Murphy, *The Independence of Linear Approximations in Symmetric Cryptology*, IEEE Transactions on Information Theory, Vol. 52, pp. 5510-5518, 2006.

## A The Serpent algorithm

The Serpent block cipher was designed by Ross Anderson, Eli Biham and Lars Knudsen [1]. It was an Advanced Encryption Standard candidate, finally rated just behind the AES Rijndael. Serpent has a classical SPN structure with 32 rounds and a block width of 128 bits. It accepts keys of 128, 192 or 256 bits and is composed of the following operations:

- an initial permutation $IP$,
- 32 rounds, each of them built upon a subkey addition, a passage through 32 S-boxes and a linear transformation $L$ (excepted the last round, where the linear transformation is not applied),
- a final permutation $FP$.

In each round $R_i$, only one S-box is used 32 times in parallel. The cipher uses 8 distinct S-boxes $S_i$ $(0 \leq i \leq 7)$ successively along the rounds and consequently, each S-box is used in exactly four different rounds. Finally, the linear diffusion transform is entirely defined by *XOR*s ($\oplus$), *rotations* ($\lll$) and left *shifts* ($\ll$). Its main purpose is to maximize the avalanche effect within the cipher. If one indicates by $X_0, X_1, X_2, X_3$ the $4 \cdot 32$ bits at the input of the linear transformation, it can be defined by the following operations:

$$\begin{array}{c}
\text{input} = X_0, X_1, X_2, X_3 \\
\hline
X_0 = X_0 \lll 13 \\
X_2 = X_2 \lll 3 \\
X_1 = X_1 \oplus X_0 \oplus X_2 \\
X_3 = X_3 \oplus X_2 \oplus (X_0 \ll 3) \\
X_1 = X_1 \lll 1 \\
X_3 = X_3 \lll 7 \\
X_0 = X_0 \oplus X_1 \oplus X_3 \\
X_2 = X_2 \oplus X_3 \oplus (X_1 \ll 7) \\
X_0 = X_0 \lll 5 \\
X_2 = X_2 \lll 22 \\
\hline
\text{output} = X_0, X_1, X_2, X_3
\end{array}$$