Improved and 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. This paper reports on the improved and multiple linear cryptanalysis of reduced round Serpent by mean of a branch-and-bound characteristic search within the algorithm. We first present a 9-round linear characteristic with probability $\frac{1}{2} + 2^{-50}$ that involves a reduction of the estimated data complexity of the best reported attack by a factor of 16. Then, we investigate the possibility to take advantage of multiple linear approximations for improving the linear cryptanalysis of Serpent. According to the framework of Biryukov *et al.* from Crypto 2004, we provide estimations of the improved data complexity of such attacks and derive practical cryptanalysis scenarios. For computational reasons, the branch-and-bound search is not guaranteed to be optimal. However, these are the best reported complexities of a linear attack against Serpent.

Keywords: linear cryptanalysis, multiple linear cryptanalysis, Advanced Encryption Standard, Serpent, linear approximations, branch-and-bound.

1 Introduction

The linear cryptanalysis [8] 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 secret key while $A[\chi]$ stands for $A_{a_1} \oplus A_{a_2} \oplus ... \oplus A_{a_n}$,with $A_{a_1}, ..., A_{a_n}$ representing particular bits of A in positions $a_1, ..., a_n$ (χ 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 *algorithm 1* and *algorithm 2*. In the first one, given an r-round linear approximation with sufficient bias, the algorithm simply counts the number of times the left side of Equation 1 is equal to zero for N pairs (plaintext, ciphertext). If $T > N/2$, then it assumes

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

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 [8] that the number of available (plaintext, ciphertext)-pairs must be proportional to $\frac{1}{\epsilon^2}$.

In the second method, an 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.

Among the various proposals to improve the linear cryptanalysis of block ciphers, Kaliski and Robshaw proposed in 1994 an algorithm using several linear approximations [6]. 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 [4] 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:

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

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 ϵ_i by means of a maximum likelihood rule. The extension of algorithm 2 to multiple approximations is similarly described in [4].

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 [4], 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: $\overline{c}^2 = 4 \cdot \sum_{i=1}^{n} \epsilon_i^2$. Therefore, by increasing this quantity by using more approximations, one can decrease the number of texts necessary to perform a successful key recovery.

In this paper, we aim to apply the previously described cryptanalytic tools to the AES candidate Serpent [1]. For this purpose, we first apply a branch-and-bound algorithm to derive an improved single linear characteristic for the cipher. It allows us to reduce the expected complexity of a linear cryptanalysis by a factor of 16. Due to the structure of the Serpent algorithm components (in particular its S-boxes and diffusion layer), the Matsui's branch-and-bound method could not be applied as such and we proposed a modified algorithm, based on the minimization of the number of active S-boxes in the linear transform. Then, in the second part of the paper, we take advantage of our modified algorithm in order to investigate multiple linear approximations. We show that a large number of linear approximations with significant biases can be derived and evaluate the resulting capacities of the obtained systems. As result of these experiments, the *theoretical* complexity against 10-rounds Serpent can be as low as 2^{80}. We mention that these conclusions have to be tempered by the possibility to perform practical attacks dealing with large number of equations and by the possibility that a significant part of these equations give rise to dependent information, as discussed at the end of this paper. Therefore, practical experiments of multiple linear cryptanalysis against actual ciphers appear to be a necessary step for the better understanding of these theoretical improvements.

2 The Serpent algorithm

The Serpent block cipher was designed by Ross Anderson, Elie 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:

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

3 Matsui's branch-and-bound approximation search

The first step in a linear cryptanalysis consists in finding linear approximations with biases as high as possible. In practice, the adversary usually starts by investigating the non-linear components in the cipher (*e.g.* the S-boxes) and tries to extrapolate partial approximations through the whole. A first problem is then to compute the probability of the concatenated linear approximations, that is usually estimated thanks to the following *piling-up lemma.* Let the bias of a linear approximation on the block cipher ith round be defined as:

$$(\chi_{I_i}, \chi_{O_i}) = \epsilon_i = \Pr\{I_i[\chi_{I_i}] \oplus O_i[\chi_{O_i}] = 0\} - 1/2 \tag{4}$$

The total bias ϵ_{tot} on r rounds is then given by:

$$\epsilon_{tot} = [\epsilon_1, \epsilon_2, ..., \epsilon_r] = 2^{r-1} \prod_{i=1}^{r} \epsilon_i, \tag{5}$$

and the best r rounds linear approximation is defined as:

$$B_r = \max_{\substack{\chi_{I_i} = \chi_{O_{i-1}} \\ (2 \leq i \leq r)}} [(\chi_{I_1}, \chi_{O_1}), (\chi_{I_2}, \chi_{O_2}), ..., (\chi_{I_r}, \chi_{O_r})] \tag{6}$$

Next to the pilling up lemma, the problem of searching the best r-round approximation is not trivial. Il consists of finding $r+1$ binary masks (one for each output round plus one mask for the input of the cipher), which define a linear approximation of maximum bias for a particular encryption algorithm. The difficulty of the problem mainly comes from the great cardinality of the set of candidates. In 1994, Matsui proposed a branch-and-bound algorithm making possible to effectively find the best linear approximation of the DES [9]. The algorithm works by induction: knowing the value of maximum bias on the *r-1* first rounds, it manages to find the maximum bias on r rounds as well as the corresponding input and output masks. For this purpose, it constantly requires to know a lower bound $\overline{B_r}$. This bound must imperatively be lower or equal to the exact B_r and the closer it is from B_r, the faster the algorithm converges.

In practice, the set of all the possible characteristics through the cipher can be seen as a large tree. At the roots stand the input masks of the cipher approximations, at each branch stand the output masks of a round and at the leaves stand the output masks of the cipher. Each branch of the tree is divided in as many new branches as there are possible approximations for the next round. In this tree, a linear approximation on i rounds forms a path starting from a root up to the i-th level. The number of leaves of the tree growing exponentially with the number of rounds, it quickly becomes unthinkable to evaluate all the approximations by exhaustive search. The principle of the branch-and-bound therefore consists in cutting whole branches of the tree as soon as it becomes obvious that all the corresponding linear approximations will have a bias lower than the bound $\overline{B_r}$. Being given an approximation on i rounds and its bias ϵ_i, a linear approximation on r rounds generated by concatenating an approximation on $r-i$ rounds with this approximation cannot have a bias better than $[B_{r-i}, \epsilon_i]$. Consequently, if $[\epsilon_i, B_{r-i}]$ is strictly lower than $\overline{B_r}$, it is useless to prospect this part of the tree more in detail. Matsui's technique is obtained by the systematic application of this reasoning and is described in more details in [9], Appendix A.

The branch-and-bound strategy can be applied as such to many ciphers. However, its efficiency and time complexity varies, notably depending on the initial estimation $\overline{B_r}$. Small estimations increase the size of the search tree. If the estimation is too large, the algorithm may not return any characteristic at all [5]. For DES, good estimations could be found by first performing a restricted search over all characteristics which only cross a single S-box in each round. Unfortunately, the algorithm may perform poorly for other ciphers and in particular, could hardly be applied as such to Serpent, as the next section underlines.

4 Linear approximations search for Serpent

4.1 Observations on the S-boxes and single round approximations

The Serpent S-boxes have only four (non-trivial) different biases: $\pm 2^{-2}$ and $\pm 2^{-3}$. Consequently, by the piling-up lemma, the bias of any approximation is a power of 2. The S-boxes S_0, S_1, S_2 and S_6 have 36 biases $\epsilon = \pm 0.25$ and 96 biases $\epsilon = \pm 0.125$. The S-boxes S_3, S_4, S_5 and S_7 have 32 biases $\epsilon = \pm 0.25$ and 112 biases $\epsilon = \pm 0.125$. In Table 1, we summarized the distributions of the approximations biases for one round of Serpent and compared them with similar results obtained in [10] for the DES and FEAL.

ϵ	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
DES	1	13	195	3803	40035	371507	...
FEAL	16	1808	98576	$3.45 \cdot 10^6$	$7.74 \cdot 10^8$	$1.22 \cdot 10^9$	...
Serpent: $S_{0,1,2,6}$	1	1153	647041	$2.35 \cdot 10^8$	$6.25 \cdot 10^{10}$	$1.29 \cdot 10^{13}$	$2.15 \cdot 10^{15}$
Serpent: $S_{3,4,5,7}$	1	1025	512513	$1.67 \cdot 10^8$	$3.96 \cdot 10^{10}$	$7.33 \cdot 10^{12}$	$1.10 \cdot 10^{15}$

Table 1: Number of 1-round linear approximations with bias $\geq \epsilon$ for some ciphers

This table clearly illustrates that the number of approximations on a round of Serpent is several orders superior to those of DES and FEAL. We consequently expected practical problems in the linear approximations search due to an explosion of the number of candidates to be explored within the branch-and-bound. This was experimentally confirmed: a classical implementation of Matsui's algorithm appeared unable to determine the best approximation on as low as three rounds. One reason for this phenomenon is the good diffusion provided by the linear transform, causing a significant increase in the number of active S-boxes at each round, with an exponential increase in the number of approximation-candidates being tested. Even by arbitrarily limiting the number of candidates to be explored, the algorithm was stuck after 5 to 6 rounds.

4.2 Modified search algorithm

Matsui's branch-and-bound method is primarily concerned with the analysis of the S-boxes in order to obtain linear approximations with high biases. However, the previously described failure suggests that in Serpent, the search for optimal linear characteristics is severely slowed down by the avalanche effect. Therefore, limiting the number of active S-boxes in the linear approximations could help to improve the search efficiency. This observation led us to modify the branch-and-bound algorithm. In our modified method, we start by exhaustively counting all the couples (input, output) of the linear transform for which the number of active S-boxes is arbitrarily low. Such couples are called *transform candidates* and are entirely defined by the number of active input/output S-boxes, the position of these S-boxes and their corresponding mask value. The transform candidates are then stored in a database (*e.g.* a hash table) so that 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 approximation search: a linear approximation on r rounds can be obtained by the concatenation of r transform candidates, if the positions of the active S-boxes at the exit of a transform candidate correspond to those of the active S-boxes at the input of the next transform candidate (*i.e.* we have to fulfill boundary conditions, as in the original method). These constraints are easily checked, as the candidates are picked up in the database according to the position of their active output S-boxes. To calculate the bias of an approximation, we simply apply the piling up lemma to the S-boxes approximations generated between the transform candidates. We can then determine the best approximations by means of a traditional branch-and-bound, where the only difference is that we pile up the transform candidates instead of piling up round approximations. Note 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 best bias in the branch-and-bound.

In order to improve the flexibility of the approximation search, the input mask in the first round and the output mask in the last round are chosen in order to maximize the biases of these rounds. Therefore, in a r rounds approximation, transform candidates are only picked up for the $r-1$ inner rounds, which speeds up the research. Additionally, the first round input mask and last round output mask can be replaced by any other mask, provided that the biases are left unchanged. Due to the properties of the Serpent S-boxes, for any approximation found by the algorithm, many more similar approximations can be generated by the modification of the outer masks.

4.3 Performance and hardware constraints

The number of possible transform candidates that one can pile up in each round is limited by the boundary conditions and the size of the lists. Moreover, among these candidates, a majority leads to a zero bias and is thus directly rejected. Indeed, a significant proportion of the Serpent S-boxes linear approximations is null: the proportion varies between $93/225 = 0.413$[1] for the S-boxes $S_0, S_1, S_2, S6$ and $81/225 = 0.36$ for the others. If there are q active S-boxes at the exit of the transform candidates, then only a fraction of roughly $(1-0.413)^q$ or $(1-0.36)^q$ of these candidates will lead to a non-zero bias. As this proportion falls when q increases, there is no exponential increase of the number of candidates anymore.

Nevertheless, the major drawback of the proposed method remains in the consequent size of the database used. Even if the proportion of useful transform candidates is weak, the number of candidates stored can easily reach several millions. In an optimized structure, a transform candidate with q active S-boxes requires $(10 + 9 * q)$ bits of memory[2]. If one considers a reasonable average of 16 active S-boxes per candidate, it yields 154 bits, *i.e.* 54400 candidates per megabyte, or approximately $55 \cdot 10^6$ candidates per gigabyte. Additionally, it is of course possible that the algorithm does not find any linear approximation beyond a certain number of rounds, either because the list of candidates checking the boundary conditions is empty, or because all these candidates lead to a zero bias. The higher the number of transform candidates, the lower the risk.

On the positive side, this database can be precomputed before the execution of the branch-and-bound. Once it is created, the execution of the algorithm is very fast since in each stage of the branch-and-bound, we only consult tables and calculate biases. In our experimentations, we generated the database as follows: we exhaustively generated all the possible transform candidates with maximum i $(1 \leq i \leq 5)$ active input S-boxes and stored only those with maximum $(15 - i)$ active output S-boxes. This required the analysis of approximately $3 \cdot 10^{11}$ linear transformations[3] among which only about $130 \cdot 10^6$ were stored/kept. With about 10^6 analyzes per second, this search took roughly 4 days on a 2GHz processor[4]. The exhaustive search of all the transform candidates with $i = 6$ would require the analysis of approximately $21 \cdot 10^{12}$ transformations, *i.e.* roughly 250 days of computation on the same processor. Better strategies could probably be investigated, taking advantage of the linearity of the diffusion layer, and are a scope for further research. Note that the database would not be the same if differential approximations were to be found [2, 9] although it would be strongly correlated.

[1] As $\exists$ 132 approximations with non-zero biases among the 15*15 non-trivial ones.
[2] Namely 2*5 bits to store the number of input/output active S-boxes (1...32) and 9 bits to store the 5-bit position (1...32) and 4-bit mask (0...15) of each active S-box.
[3] That is $2 \cdot \sum_{k=1}^{5} 15^k * \mathrm{C}_{32}^{k}$.
[4] Note that the task can be parallelized.

5 Practical results

5.1 Best approximation found

Since the best 9-round characteristic found by Biham *et al.* involves at most 15 active S-boxes at the extremity of the linear transform [3], our previously described database ensures us to find an approximation with a bias at least as high. Table 2 summarizes the results of our approximations search, in function of the starting S-box (and therefore round). The best biases are given in the penultimate column and the values of Biham *et al.* are in the last column. As a first result of our investigations, we found an improved approximation starting with S_3. Our improved characteristic is very similar to the one in [3], excepted for the three first rounds. It involves a reduction of the linear cryptanalysis data complexity by a factor of 16. Due to a lack of room, the description of the linear approximations used in our attacks are not given in this paper. It is available at: http://www.dice.ucl.ac.be/crypto/publications/2007/inscrypt_appendix.pdf.

r	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	Max	$\overline{B_r}$
3	–	2^{-8}	2^{-8}	2^{-8}	2^{-8}	2^{-7}	2^{-8}	2^{-7}	2^{-7}	2^{-7}
4	–	2^{-14}	2^{-13}	2^{-16}	2^{-12}	2^{-12}	2^{-12}	2^{-12}	2^{-12}	2^{-12}
5	–	2^{-23}	2^{-21}	2^{-23}	2^{-17}	2^{-18}	2^{-18}	2^{-18}	2^{-17}	2^{-18}
6	–	–	–	2^{-30}	2^{-23}	2^{-25}	2^{-24}	2^{-33}	2^{-23}	2^{-25}
7	–	–	–	2^{-36}	2^{-30}	2^{-32}	–	–	2^{-30}	2^{-32}
8	–	–	–	2^{-43}	2^{-37}	–	–	–	2^{-37}	2^{-39}
9	–	–	–	2^{-50}	–	–	–	–	2^{-50}	2^{-52}

Table 2: Biases of the best 9-round approximations for different starting S-boxes.

Note that we also tried to find iterative approximations, *i.e.* approximations of which the input masks fulfill the boundary conditions of their own output. Such approximations are useful in practice because they can be straightforwardly extended to an arbitrary number of rounds [7]. However, our restricted database did not allow us to find any such approximation on 8 rounds. Searching over larger databases would therefore be necessary to further investigate the existence of good iterative characteristics.

5.2 Multiple linear approximations found

In addition to the previously reported characteristic, we ran our search algorithm in order to generate a list of useful approximations for a multiple linear cryptanalysis of Serpent. As for the previous section, this generation is very fast due to the properties of the branch-and-bound that allows an effective limitation of the candidates to explore. Table 3 reports the bias distribution of the 150 best 9-rounds linear approximations of Serpent found with our database (starting with S_3). Additionally, since each of these approximations can generate several others by simply replacing its input and output masks (as discussed in Section 4.2), we straightforwardly obtained the distribution in table 4.

bias	# of approx.	bias	# of approx.
2^{-50}	3	2^{-53}	42
2^{-51}	12	2^{-54}	66
2^{-52}	27	2^{-55}	

Table 3: Bias distribution of the 150 best approximations found.

bias	# of approx.	capacity	bias	# of approx.	capacity
2^{-50}	786432	$2.482 \cdot 10^{-24}$	2^{-55}	$1.208 \cdot 10^{13}$	$5.184 \cdot 10^{-20}$
2^{-51}	$6.134 \cdot 10^{7}$	$5.087 \cdot 10^{-23}$	2^{-56}	$1.250 \cdot 10^{14}$	$1.481 \cdot 10^{-19}$
2^{-52}	$2.290 \cdot 10^{9}$	$5.024 \cdot 10^{-22}$	2^{-57}	$1.059 \cdot 10^{15}$	$3.520 \cdot 10^{-19}$
2^{-53}	$5.447 \cdot 10^{10}$	$3.188 \cdot 10^{-21}$	2^{-58}	$7.513 \cdot 10^{15}$	$7.138 \cdot 10^{-19}$
2^{-54}	$9.281 \cdot 10^{11}$	$1.463 \cdot 10^{-20}$	2^{-59}	$4.553 \cdot 10^{16}$	$1.262 \cdot 10^{-18}$

Table 4: Bias distribution and cumulative capacities of the extended approximations.

5.3 Resulting capacity and discussion of the results

According to the framework in [4], the use of multiple approximations in linear cryptanalysis allows decreasing the number of plaintexts needed for a successful key recovery proportionally to the capacity of the obtained system (given in Table 4). It involves the following observations:

- The data complexity of the simple linear cryptanalysis of 10-rounds Serpent is approximately 2^{100} (using the best 9-round approximation with bias 2^{-50}).
- If the 786432 approximations with bias 2^{-50} are used, the resulting capacity equals $2.482 \cdot 10^{-24}$, which yields a theoretical data complexity of $2^{78.4}$.
- Cumulatively using all the $2.29 \cdot 10^9$ approximations of bias higher than 2^{-52}, we could theoretically reach a capacity of $5.024 \cdot 10^{-22}$, which would correspond to $2^{70.8}$ plaintext-ciphertext pairs.
- The more realistic use of 2048 (*resp.* $1.802 \cdot 10^6$) approximations with bias 2^{50} (*resp.* greater than 2^{52}) involving the same target subkey would result in a theoretical data complexity of 2^{87} (*resp.* 2^{81}).

As a matter of fact, these results do not take the size of the target subkey (and therefore the time complexity) into account but only consider the data complexity. In the next section, we propose more realistic attacks presenting a better trade-off between data and time complexities. Let us also mention that a possibly more powerful way to exploit multiple approximations would be to consider Matsui's *algorithm 1* and therefore avoid the time complexity problems related to key guesses, *e.g.* using the three 2^{-50} bias approximations and their derivatives. Since each approximation reveals up to one bit of information on the secret key and $m \gg 128$, the resulting linear system is strongly overdefined.

In general, the previous results have to be tempered by the possible influences of dependencies between the various linear approximations exploited in an attack. This is specially true in our context since our approximations are all generated from an initial set of 150 characteristics and the number of derivatives is much larger than $2 \cdot 128$. As discussed in [4], it is actually hard to determine the consequence of these dependencies on the capacity of the multiples approximations. Because $2 \cdot m \cdot \epsilon \ll 1$ (for any reasonable choice for the number m of approximations), the dependencies between the text masks (χ_P^i, χ_C^i) should have a negligible influence on the capacity. Therefore, the major question relates to the dependencies between the linear trails. As a matter of fact, the estimated data complexities in our analysis (as well as in Biryukov *et al.*'s) are fairly optimistic and an important next step in the understanding of linear cryptanalysis would be to experiment these predictions with a real life cipher.

6 Realistic attack scenarios against Serpent

In this section, we present realistic attack scenarios on reduced round Serpent using (multiple) linear cryptanalysis. The reduced version is just like Serpent, excepted for its reduced number of rounds. After the last S-box, the linear transformation is omitted as it has no cryptographic significance. The linear approximations used were generated with the algorithm presented before.

Using an approximation on $r-1$ rounds, one can recover bits of the subkey in round r. In Matsui's original method, a partial decryption of the last round is performed for every ciphertext by guessing the key bits involved in the approximation. The parity of the approximation for the plaintext and the partially decrypted ciphertext is then evaluated and a counter corresponding to each key guess is incremented if the relation holds or decremented otherwise. The key candidate with the highest counter in absolute value is then assumed to be the correct key. However, as we only consider a limited number of bits k (in the active S-boxes) during the partial decryption of the ciphertexts, the same pattern for these k bits possibly appear several times during the attack. In order to avoid doing the same partial decryption work several times, Biham *et al.* proposed in [3] an improvement which considerably reduces the time complexity of an attack:

– Initialize an array of 2^k counters (k is the size of the target subkey).
– *For each generated ciphertext:* extract the k-bit value corresponding to the active S-boxes and evaluate the parity of the plaintext subset defined by the approximation. Increment or decrement a counter corresponding to the extracted k-bit value according to the obtained parity.
– *Once all the ciphertexts have been generated:* for each k-bit ciphertext and for each k-bit subkey, partialy decrypt the k-bit ciphertext under the k-bit subkey and evaluate the parity of the output subset (as defined by the linear approximation). Keep this value in a table of size $2^k \cdot 2^k$.
– For each k-bit subkey, evaluate its experimental bias by checking, for each k-bit ciphertext, the parity of the approximation and the value of the corresponding counter. Then output the subkey with maximal bias.

This method reduces the attack time complexity from $O(N \cdot 2^k)$ to $O(2^k \cdot 2^k)$.

6.1 Attack on 7 rounds Serpent

The attack uses a 6-round approximation starting with S-box 4 and ending with S-box 1. This approximation is derived from the best 6-round approximation found in section 5 except a small change in the last round that reduces the number of active S-boxes from 13 to 5. The bias of the approximation falls from 2^{-23} to 2^{-25}. Using only one approximation, the data complexity is approximately 2^{52} known plaintexts. The attack requires 2^{20} counters and a time complexity of approximately 2^{40} decryptions of 1-round Serpent. Several similar approximations can be obtained by changing the input mask of the relation. We found up to 8 approximations with bias 2^{-25} and up to 96 approximations with bias 2^{-26}. This would lead to a capacity of respectively 2^{-45}, 2^{-43}, reducing the data complexity to 2^{47} or 2^{45} at the cost of a slight increase of the time/memory complexities.

6.2 Attack on 8 rounds Serpent

Similarly, we can attack 8-round Serpent using a 7-round approximation starting with S-box 4 and ending with S-box 2. The approximation is the one found in section 5. It has a bias of 2^{-30} and 7 active S-boxes. Consequently, an attack using only one approximation requires 2^{56} 1-round decryptions, 2^{28} counters and $\simeq 2^{62}$ known plaintexts. We can again reduce the data complexity by taking advantage of multiple approximations. We found 8 approximations with the same bias and the same active S-boxes giving a capacity of 2^{-55}, thus a data complexity of approximately 2^{57} plaintexts. Adding 96 approximations with bias 2^{-31}, we obtain a capacity of 2^{-53} and therefore a data complexity of 2^{55} plaintexts (see Table 6). Again, this effect can be increased at the cost of more memory and computation. For example, there are 384 approximations with bias 2^{-32} and 512 approximations with bias 2^{-33}, that gives rise to data complexities of respectively $2^{54.1}$ or 2^{54}, but requires 2^{28} counters and 2^{56} memory access for each counter.

6.3 Attack on 9 rounds Serpent

The best approximation found on 8 rounds has a bias of 2^{-37} but it has 23 actives input S-boxes. We can slightly modify its input in order to lower this number to 11 active S-boxes. This way, the bias of the approximation is 2^{-39} instead of 2^{-37}. This approximation starts with S-box 4 and ends with S-box 3. The complexity of an attack based on this approximation is 2^{88} 1-round decryptions, 2^{44} counters and about 2^{80} known plaintexts. Multiples approximations allow us to decrease the number of texts needed. We found 128 approximations with bias 2^{-39}, 3584 approximations with bias 2^{-40}, 43008 approximations with bias 2^{-41} and 286720 approximations with bias 2^{-42}. The corresponding capacities are 2^{-69}, 2^{-66}, $2^{-64.1}$, 2^{-63}, thus requiring 2^{71}, 2^{68}, $2^{66.1}$, 2^{65} generated plaintexts.

6.4 Attack on 10 rounds Serpent

We finally ran our search algorithm to generate a list of 150 9-round approximations with high bias and a reasonable number of active S-boxes. Among the the huge number of candidates (see table 4), we found the three following approximations starting and finishing with S-box 3:

- Approximation 1 with bias 2^{-55} and 11 active S-boxes,
- Approximation 2 with bias 2^{-58} and 10 active S-boxes,
- Approximation 3 with bias 2^{-59} and 8 active S-boxes.

Using the first approximation, we obtain an attack requiring 2^{112} texts, 2^{44} counters and 2^{88} decryptions. Using the second approximation, we obtain an attack requiring 2^{118} texts, 2^{40} counters and 2^{80} decryptions. Using the third approximation, we obtain an attack requiring 2^{120} texts, 2^{32} counters and 2^{64} decryptions. Multiple linear cryptanalysis based on the first approximation leads to capacities equal to 2^{-97}, $2^{-93.42}$ or $2^{-90.93}$ according to bias of the approximations. Using the second approximation, the capacities decrease to 2^{-103}, $2^{-99.42}$ or $2^{-96.93}$. With the third approximation the capacities then become 2^{-105}, $2^{-101.42}$ or $2^{-98.93}$. All the presented attack results are summarized in Table 6 and Table 5 remembers the previously known attacks against Serpent.

6.5 Attack on 11 rounds Serpent

We can attack 11 round Serpent with a 9-rounds linear approximation. Such an attack requires a partial encryption before the first round of the approximation and a partial decryption after the last round. In this context, it becomes essential to minimize the total number of active S-boxes, both in input and in output.

Our algorithm provided a 9-rounds approximation with a bias of 2^{-58} and only 27 active S-boxes (15 in input and 12 in output). Using the trick proposed in [3], section6, we obtain a time complexity of $2^{120} \cdot 15/352 + 2^{48} \cdot (2^{118} \cdot 12/352 + 2^{60}) = 2^{166}$ 11-round Serpent encryptions and a memory complexity of 2^{121}. Incidentally, if we perform a partial encryption of the first round in the external loop (instead of a partial decryption of the last round), the time complexity becomes $2^{96} \cdot 12/352 + 2^{60} \cdot (2^{118} \cdot 15/352 + 2^{48}) = 2^{173.5}$ 11-round Serpent encryptions and a memory complexity of 2^{97}. The data complexity is left unchanged, that is 2^{118} known plaintext.

In this case, it is practically not possible to use multiples approximations, as they should have at least the same active S-boxes, both in their input and output.

6.6 Summary

Rounds	Type of attack	complexity		
		data	time	memory
6	differential [13]	2^{83}CP	2^{90}	2^{44}
	differential [13]	2^{71}CP	2^{103}	2^{79}
	differential [13]	2^{41}CP	2^{163}	2^{49}
7	differential [11]	2^{84}CP	$2^{78.9}$	2^{56}
8	Amp.Boomerang [13]	2^{128}CP	2^{163}	2^{137}
	Amp.Boomerang [13]	2^{110}CP	2^{175}	2^{119}
	differential [11]	2^{84}CP	$2^{206.7}$	2^{89}
9	Amp.Boomerang [13]	2^{110}CP	2^{252}	2^{212}
10	Rectangle [14]	$2^{126.3}$CP	2^{165}	$2^{131.8}$
	Boomerang [14]	$2^{126.3}$ACPC	2^{165}	2^{89}
	Lin.Cryptanalysis [3]	2^{116}KP	2^{92}	2^{45}
	Diff.Lin.Cryptanalysis [15]	$2^{105.2}$CP	$2^{123.2}$	2^{40}
11	Lin.Cryptanalysis [3]	2^{118}CP	2^{187}	2^{193}
	Diff.Lin.Cryptanalysis [15]	$2^{125.3}$CP	$2^{172.4}$	2^{30}
	Diff.Lin.Cryptanalysis [15]	$2^{125.3}$CP	$2^{139.2}$	2^{60}
Complexity is measured in encryption units. Memory is mesured in Bytes. CP - Chosen Plaintexts, KP - Known Plaintexts, ACPC - Adaptive Chosen Plaintexts and Ciphertext.				

Table 5: Summary of previous attacks on Reduced-rounds Serpent (see [15]).

Rounds	Type of attack	complexity		
		data	time	memory
7	Lin.cryptanalysis	2^{52}KP	2^{40}	2^{20}
	Mult.Lin.Cryptanalysis (8 appr.)	2^{47}KP	2^{43}	2^{23}
8	Lin.cryptanalysis	2^{62}KP	2^{56}	2^{28}
	Mult.Lin.Cryptanalysis (8 appr.)	2^{57}KP	2^{59}	2^{31}
	Mult.Lin.Cryptanalysis (104 appr.)	2^{55}KP	$2^{62.7}$	$2^{34.7}$
9	Lin.cryptanalysis	2^{80}KP	2^{88}	2^{44}
	Mult.Lin.Cryptanalysis (128 appr.)	2^{71}KP	2^{95}	2^{51}
	Mult.Lin.Cryptanalysis (3712 appr.)	2^{68}KP	$2^{99.9}$	$2^{55.9}$
10	Lin.cryptanalysis ($\epsilon = 2^{-55}$)	2^{112}KP	2^{88}	2^{44}
	Mult.Lin.Cryptanalysis (2048 appr.)	2^{99}KP	2^{99}	2^{55}
	Lin.cryptanalysis ($\epsilon = 2^{-59}$)	2^{120}KP	2^{64}	2^{32}
	Mult.Lin.Cryptanalysis (2048 appr.)	2^{107}KP	2^{75}	2^{43}
11	Lin.cryptanalysis ($\epsilon = 2^{-58}$)	2^{118}KP	2^{166}	2^{121}
	Lin.cryptanalysis ($\epsilon = 2^{-58}$)	2^{118}KP	$2^{173.5}$	2^{97}

Table 6: Summary of attacks on Serpent presented in this paper.

7 Conclusion and further works

This paper first presents a modification of Matsui's branch-and-bound algorithm for the linear approximation search in a block cipher. It enabled us to find the best reported 9-round approximation for the AES candidate Serpent. The algorithm allows speeding up the search of linear approximations at the cost of larger memory requirements. It is generic and especially well suited for ciphers where the linear transformation involves an important avalanche effect (as the AES candidates and most recent ciphers). Moreover, it could be straightforwardly adapted for the research of differential characteristics.

In a second part of the paper, we take advantage of this modified branch-and-bound algorithm in order to investigate the possible use of multiple linear approximations against Serpent. According to the framework of Biryukov *et al.* [4], we provided estimations of the improved data complexity of such attacks against 10-round Serpent that can be down to approximately 2^{80}. Since these results are mainly theoretical (due to an unrealistic time complexity), we also presented several attacks against 7- to 10-round Serpent using reasonable attack parameters that outperform previously known results.

As an important scope for further research, these results should be experimented against real-life ciphers of tractable size in order to determine the actual influence of dependencies between the different approximations used in an attack. That is, to figure out the extend to which the information provided by multiple linear approximations can lead to efficient attack strategies.

Acknowledgements

The authors thank Orr Dunkelman for his helpful answers and advices. We also thank anonymous reviewers from FSE and Inscrypt for their useful comments.

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. E. Biham, *On Matsui's Linear Cryptanalysis*, in the proceedings of Eurocrypt 1994, Lecture Notes in Computer Science, vol. 950, pp. 341-355, Perugia, Italy, May 1994.
3. E. Biham, O. Dunkelman, N. Keller, *Linear Cryptanalysis of Reduced Round Serpent*, in the Proceedings of FSE 2001, Lecture Notes in Computer Science, vol. 2355, pp. 16-27, Yokohama, Japan, April 2001.
4. 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.
5. A. Biryukov, *Linear Cryptanalysis*, in the Encyclopedia of Cryptography and Security, Kluwer Academic Publishers, 2005.
6. 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.
7. L.R. Knudsen, *Iterative characteristics of DES and s^2-DES*, in the proceedings of CRYPTO 1992, Lecture Notes in Computer Science, vol. 746, pp. 497-511, Santa Barbara, California, USA, AUgust 1992.
8. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of Eurocrypt 1993, Lecture Notes in Computer Science, vol. 765, pp. 386–397, Lofthus, Norway, May 1993.
9. 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.
10. K. Ohta, S. Moriai, K. Aoki, *Improving the Search Algorithm for the Best Linear Expression*, in the proceedings of CRYPTO 1995, Lecture Notes in Computer Science, vol. 963, pp. 157-170, Santa Barbara, California, USA, August 1995.
11. E. Biham, O.Dunkelman, N.Keller, *The Rectangle Attack - Rectangling the Serpent*, Advances in Cryptology - Eurocrypt'01 (Lecture Notes in Computer Science no.2045), pp. 340-357, Springer-Verlag, 2001.
12. T. Kohno, J.Kelsey, B.Schneier, *Preliminary Cryptanalysis of Reduced-Round Serpent*, AES Candidate Conference, pp. 195-211, 2000
13. J.Kelsey, T. Kohno, B.Schneier, *Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent*, proceedings of Fast Software Encryption 7, Lecture Notes in Computer Science, vol. 1978, pp. 75-93, Springer-Verlag, 1999.
14. E. Biham, O. Dunkelman, N. Keller, *New Results on Boomerang and Rectangle Attacks*, in the Proceedings of Fast Software Encryption 9, Lecture Notes in Computer Science, vol. 2501, pp. 254-266, Springer-Verlag, 2002.
15. E. Biham, O. Dunkelman, N. Keller, *Differential-linear Cryptanalysis of Serpent*, in the Proceedings of Fast Software Encryption 2003, Lecture Notes in Computer Science, vol. 2887, pp. 9-21, Springer, 2004.

A Matsui's Branch-and-bound for SPN

The following pseudo-code implements Matsui's branch-and-bound algorithm. It can be interpreted as a Depth-First Search algorithm which decides, for each node of the tree (as described in section 3), if it is convenient to continue the descent on the child nodes. The program updates $\overline{B_r}$ each time a better approximation is discovered (in procedure Round-1). As the algorithm virtually goes through the whole tree, $\overline{B_r}$ reaches the optimal value Br at the end of the execution. Note also that if the worst case complexity of the algorithm is still exponential, it can be much less in practice (depending on the choice of the initial values $\{B_1, B_2, B_3, ..., B_{r-1}\}$ and $\overline{B_r}$).

Algorithme 1 branch-and-bound

```
input: $\{B_1, B_2, B_3, ..., B_{r-1}\}$ and $\overline{B_r}$

procedure Round-r:
for each candidate for $\chi_{I_r}$ do
    let $\epsilon_r = max_{\chi_{O_r}}(\chi_{I_r}, \chi_{O_r})$
    if $[B_{r-1}, \epsilon_r] \geq \overline{B_r}$ then
        call procedure Round-(r-1)
    end if
end for
return $\overline{B_r}$
exit the program

procedure Round-i $(2 \leq i \leq r-1)$:
for each candidate for $\chi_{I_i}$ do
    let $\chi_{O_i} = \chi_{I_{i+1}}$ and $\epsilon_i = (\chi_{I_i}, \chi_{O_i})$
    if $[B_{i-1}, \epsilon_i, ..., \epsilon r] \geq \overline{B_r}$ then
        call procedure Round-(i-1)
    end if
end for
return to the upper procedure

procedure Round-1:
let $\chi_{O_1} = \chi_{I_2}$ and $\epsilon_1 = max_{\chi_{I_1}}(\chi_{I_1}, \chi_{O_1})$
if $[\epsilon_1, \epsilon_2, ..., \epsilon_r] \geq \overline{B_r}$ then
    $\overline{B_r} = [\epsilon_1, \epsilon_2, ..., \epsilon_r]$
end if
return to the upper procedure
```