Linear Cryptanalysis of Reduced Round Serpent[*]

Eli Biham[1], Orr Dunkelman[1], and Nathan Keller[2]

[1] Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel, {biham,orrd}@cs.technion.ac.il, http://www.cs.technion.ac.il/~biham/. http://vipe.technion.ac.il/~orrd/me/.

[2] Mathematics Department, Technion – Israel Institute of Technology, Haifa 32000, Israel, nkeller@tx.technion.ac.il

Abstract. Serpent is one of the 5 AES finalists. In this paper we present a 9-round linear approximation for Serpent with probability of $1/2+2^{-52}$. We use it to attack 10-round Serpent with all key lengths with data complexity of 2^{118} and running time of 2^{89}. A variant of this approximation is used in the first attack against an 11-round Serpent with 192-bit and 256-bit keys, which require the same amount of data and 2^{187} running time.

1 Introduction

Serpent [1] is a block cipher which was suggested as a candidate for the Advanced Encryption Standard (AES) [9], and was selected to be among the five finalists.

In [5] a modified variant of Serpent in which the linear transformation was modified into a permutation was analyzed. The permutation allows one active S box to activate only one S box in the consecutive round, a property that cannot occur in Serpent. Thus, it is not surprising that this variant is much weaker than Serpent, and that it can be attacked with up to 35 rounds.

In [7] the 256-bit variant of Serpent up to 9 rounds is attacked using an amplified boomerang attack. The attack is based on building a 7-round distinguisher for Serpent, and using it for attacking up to 9 rounds. The distinguisher is built using the amplified boomerang technique. It uses a 4-round differential characteristics in rounds 1–4, and a 3-round characteristic in rounds 5–7.

In [4] 10-round Serpent with 256-bit keys was analyzed using the rectangle attack. Also the best known 3-round, 4-round, 5-round and 6-round differential characteristics of Serpent have been presented.

In this paper we present the best known linear approximation of Serpent of 9 rounds which has a probability of $1/2 + 2^{-52}$. We use a variant of this approximation (with bias of 2^{-58} in order to attack 10-round Serpent with all key lengths with data complexity of 2^{118} and running time of 2^{89} memory accesses.

[*] This work was supported by the European Union fund IST-1999-12324 – NESSIE

Using another variant of this approximation allows us to attack up to 11-round Serpent with 192-bit and 256-bit keys, using the same amount of data and which requires time equivalent to 2^{187} 11-round Serpent encryptions.

The paper is organized as follows: In Section 2 we give the description of Serpent. In Section 3 we present the 9-round linear approximation of Serpent. In Section 4 we describe the search methodology we used to find the approximation. In Section 5 we present the linear attack on 10-round Serpent with all possible key lengths. In Section 6 we present the linear attack on 11-round Serpent with 256-bit keys. Section 7 summarizes the paper.

2 A Description of Serpent

Serpent [1] is a SP-network block cipher with block size of 128 bits and 0–256 bit keys. It is composed of alternating layers of key mixing, S boxes and linear transformation. We deal only with the equivalent bitsliced description.

The key scheduling algorithm of serpent accepts 128-bit, 192-bit and 256-bit keys. Keys shorter than 256 bits are padded by 1 followed by as many 0's needed to have a total length of 256 bits. The key is then used to derive 33 subkeys of 128 bits.

We use the notations of [1]. Each intermediate value of round i is denoted by $\hat{B}_i$ (a 128-bit value). The 32 rounds are numbered 0–31. Each $\hat{B}_i$ is treated as four 32-bit words X_0, X_1, X_2, X_3, where bit j of X_i is bit $4 * i + j$ of the 128 bit value $\hat{B}_i$ (i.e., bit j of $X_3, \ldots, X_0$ is the j'th nibble, with the bit from X_3 as the most significant bit).

Serpent has a set of eight 4-bit to 4-bit S boxes. Each round function R_i ($i \in \{0, \ldots, 31\}$) uses a single S box 32 times in parallel. For example, R_0 uses S_0, 32 copies of which are applied in parallel. Each S box permute nibble, i.e., for $0 \le j \le 31$ we take the j'th bit of $X_3, \ldots, X_0$ and return the value according to the S box.

The set of eight S-boxes is used four times. In round i we use $S_{i \bmod 8}$, i.e., S_0 is used in round 0, etc. The last round (round 31) is slightly different from the others: apply S_7 on $\hat{B}_{31} \oplus \hat{K}_{31}$, and XOR the result with $\hat{K}_{32}$ rather than applying the linear transformation.

The cipher may be formally described by the following equations:

$$\begin{aligned} \hat{B}_0 &:= P \\ \hat{B}_{i+1} &:= R_i(\hat{B}_i) \\ C &:= \hat{B}_{32} \end{aligned}$$

where

$$\begin{aligned} R_i(X) &= LT(\hat{\mathcal{S}}_i(X \oplus \hat{K}_i)) \qquad i = 0, \ldots, 30 \\ R_i(X) &= \hat{\mathcal{S}}_i(X \oplus \hat{K}_i) \oplus \hat{K}_{32} \quad i = 31 \end{aligned}$$

where $\hat{\mathcal{S}}_i$ is the application of the S-box $S_{i \bmod 8}$ 32 times in parallel, and LT is the linear transformation.

The linear transformation: The 32 bits in each of the output words are linearly mixed by

$$\begin{aligned}
X_0, X_1, X_2, X_3 &:= \mathcal{S}_i(B_i \oplus K_i) \\
X_0 &:= X_0 <<< 13 \\
X_2 &:= X_2 <<< 3 \\
X_1 &:= X_1 \oplus X_0 \oplus X_2 \\
X_3 &:= X_3 \oplus X_2 \oplus (X_0 << 3) \\
X_1 &:= X_1 <<< 1 \\
X_3 &:= X_3 <<< 7 \\
X_0 &:= X_0 \oplus X_1 \oplus X_3 \\
X_2 &:= X_2 \oplus X_3 \oplus (X_1 << 7) \\
X_0 &:= X_0 <<< 5 \\
X_2 &:= X_2 <<< 22 \\
B_{i+1} &:= X_0, X_1, X_2, X_3
\end{aligned}$$

where $<<<$ denotes rotation, and $<<$ denotes shift.

In the last round, this linear transformation is replaced by an additional key mixing: $B_{32} := S_7(B_{31} \oplus K_{31}) \oplus K_{32}$.

3 A 9-Round Linear Approximation

We adopt the representation similar to the differential characteristics as in [6,4], and add more data to the figures. In our representation, the rows are the bitsliced 32-bit words, and each column is the input to a different S box. The first line represents X_0, the last line represents X_3, and the rightmost column represents the least significant bit of the words. A thin arrow represents an approximation with probability $\frac{1}{2} \pm \frac{1}{8}$ for the specific S box (given the input parity, the output parity is achieved with probability $\frac{1}{2} \pm \frac{1}{8}$), and a fat arrow stands for probability $\frac{1}{2} \pm \frac{1}{4}$. If the bit is taken into consideration of the parity, the box related to it is filled. Example for our notation can be found in Figure 1, in which in the first S box (related to bits 0) the parity of 1 equals to the parity of 3 in the output with probability $1/2 \pm 1/4$ (1/4 or 3/4), and in S box 30 the parity of the input 3 causes an output parity of 1 with probability $1/2 \pm 1/8$ (3/8 or 5/8).

This 9-round linear approximation has a probability of $1/2 + 2^{-52}$, in round 3 (or 11 or 19 or any other round using S_3) the following approximation holds with bias of 2^{-11}:

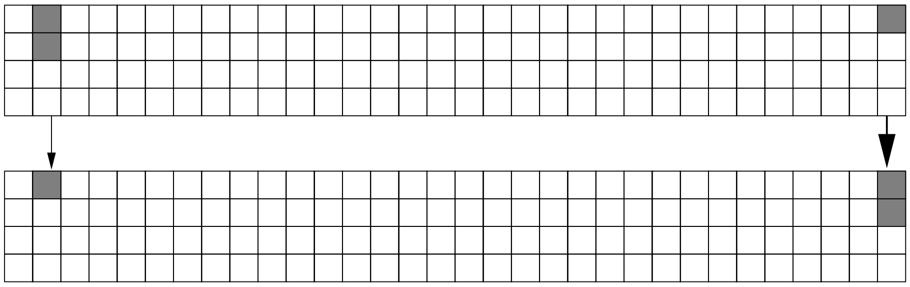
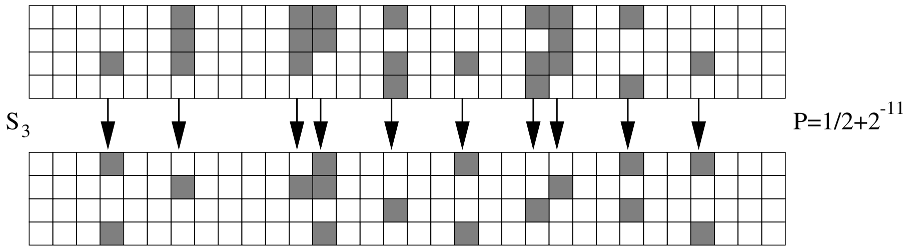
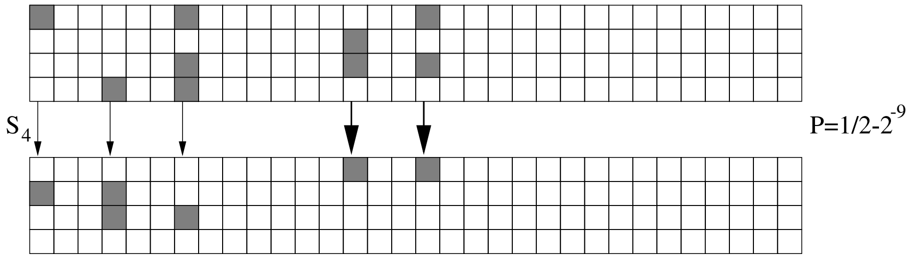


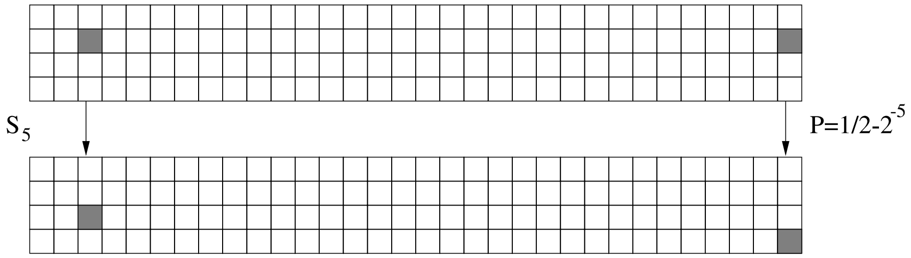
Fig. 1. Linear Approximation Representation Example



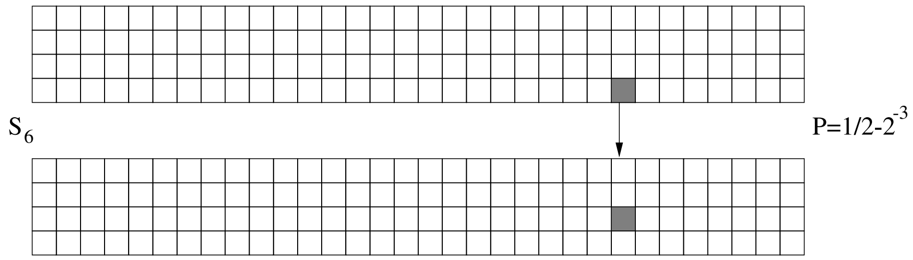
After the linear transformation, and the application of S_4 we get the following linear approximation with bias of 2^{-9}:



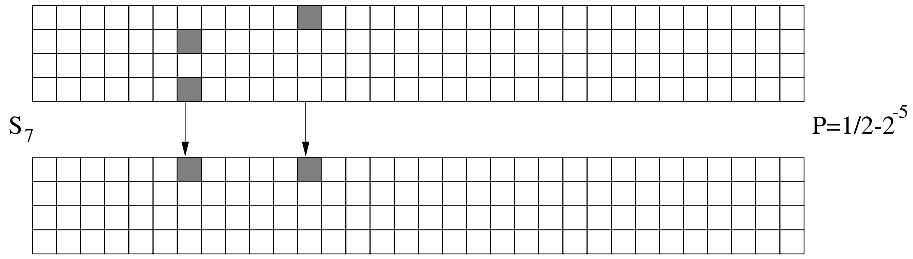
After the linear transformation, and the application of S_5 we get the following linear approximation with bias of 2^{-5}:



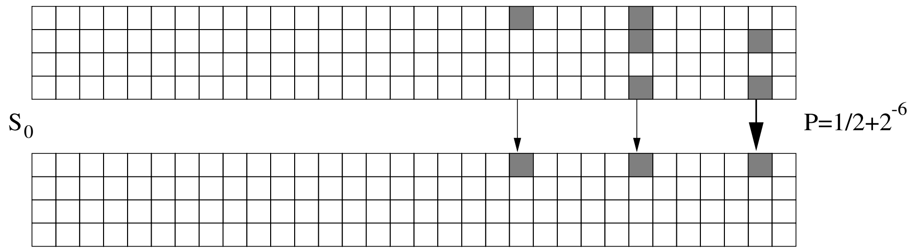
After the linear transformation, and the application of S_6 we get the following linear approximation with bias of 2^{-3}:



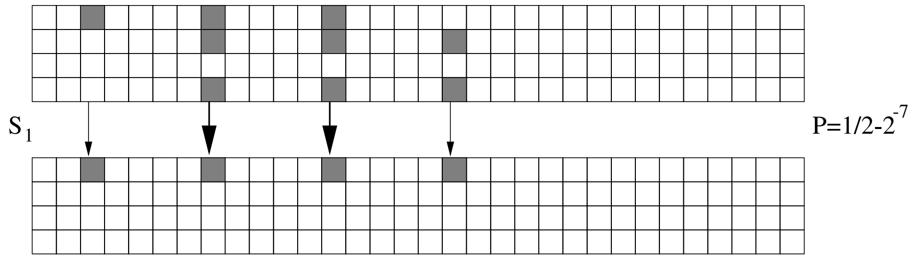
After the linear transformation, and the application of S_7 we get the following linear approximation with bias of 2^{-5}:



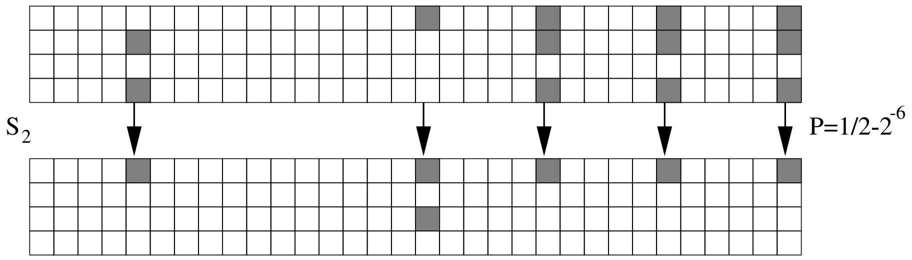
After the linear transformation, and the application of S_0 we get the following linear approximation with bias of 2^{-6}:



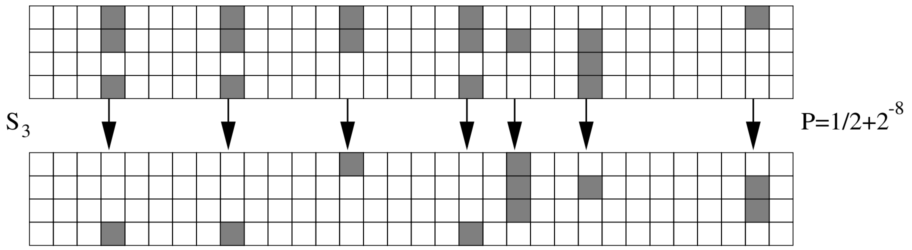
After the linear transformation, and the application of S_1 we get the following linear approximation with bias of 2^{-7}:



After the linear transformation, and the application of S_2 we get the following linear approximation with bias of 2^{-6}:



After the linear transformation, and the application of S_3 we get the following linear approximation with bias of 2^{-8}:



The total bias of the approximation is therefore $2^8 \cdot 2^{-11} \cdot 2^{-9} \cdot 2^{-5} \cdot 2^{-3} \cdot 2^{-5} \cdot 2^{-6} \cdot 2^{-7} \cdot 2^{-6} \cdot 2^{-8} = 2^{-52}$.

We can play with the input subsets and output subsets. There are 2^{10} possible input biases (all with the same bias, half of them has probability of $1/2 + 2^{-11}$ and half with probability $1/2 - 2^{-11}$), and 2^7 output subsets with the same bias. Thus, receiving 2^{17} possible approximations with the same bias. 2^{16} have probability $1/2 + 2^{-52}$, and the other 2^{16} have probability $1/2 - 2^{-52}$.

These approximations can also be rotated by one or two bits to the left, due to the nature of the linear transformation. Thus, we have in total $3 \cdot 2^{17}$ approximations.

4 Search Methodology

We have searched for the the linear characteristic in the following way:

- We searched over all one bit inputs to the linear transformation and looked for the cases where the minimal number of S boxes are affected in the next round.
- We searched over all one bit outputs from the linear transformation and looked for the cases where the minimal number of S boxes are affected in the previous round.
- We observed that knowing a least significant bit of a nibble (bit which is in X_0) in the entrance to the linear transformation, affects 2 to 3 S boxes in the next round.
- We observed that knowing a most significant bit of a nibble (bit which is in X_3) in the exit from the linear transformation, affects 2 to 3 S boxes in the previous round.

- For S box 7, 8 and 9 (out of the parallel 32), the input has only two active S boxes in the round before, and the output cause only two active S boxes in the next round. Thus, we found a 3-round approximation with 5 active S boxes.
- We tried to add more rounds to the approximation, using the fact that when we go backward to the previous round, it is better to take subsets of bits from X_1 and X_3, and while going forward it is better to take subsets of bits from X_0 and X_2.
- By iteratively adding rounds in the beginning and in the end of the approximation we have found a 9-round approximation.

The approximations' details we found in the way on fewer rounds are given in Table 1. The approximations themselves can be easily derived from the 9-round approximation by removing the unnecessary rounds, and making the edge rounds (first and last round) of the approximation with maximal bias. From this table it becomes clear that the authors of Serpent made a mistake in their claimed bounds, as the biases found here are 4-8 times higher than the bounds in [1]. However, this mistake does not affect the security claims for the whole cipher, as there is a huge distance between a 9-round approximation and attacking 32 rounds, or even 16 rounds of Serpent.

Table 1. Linear Approximations for Serpent

Number of Rounds	Starting from	Number of Active S boxes	Bias
3	S_5	5	2^{-7}
4	S_5	8	2^{-12}
5	S_5	12	2^{-18}
6	S_4	17	2^{-25}
6	S_5	17	2^{-25}
7	S_4	22	2^{-32}
8	S_4	29	2^{-39}
9	S_3	39	2^{-52}

5 Attacking 10-Round Serpent

We can attack 10-round Serpent (in rounds 3–12) using a 9-round linear approximation (in rounds 3–11), which is similar to the one presented in Section 3, besides the last round of the approximation (round 11) which is shown in Figure 2. This replacement gives us a linear approximation with probability of $1/2 - 2^{-58}$ (instead of $1/2 + 2^{-52}$), but we reduce by 12 the number of active S boxes in round 12, as this last round activates only 11 S boxes in round 12 instead of 23 in the original approximation.

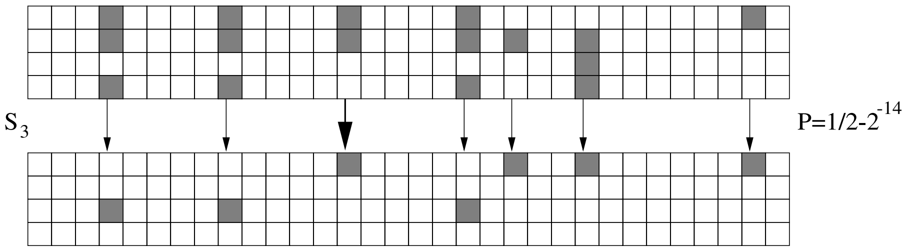


Fig. 2. Last Round of the 9-round Linear Approximation used in the Attack

The basic algorithm of the attack is based on the basic M2 algorithm [8]. Given 2^{118} plaintexts perform the following algorithm:

– Initialize array of 2^{44} counters (counters must be at least 118 bits long).
– For each of the 44-bit subkey value of the 11 S boxes in the last round:
  - Decrypt each ciphertext using the subkey value in the 11 S boxes in the last round.
  - Calculate the input subset parity and the output subset parity required by the approximation, and calculate the XOR of two. If the XOR value is zero increase the counter related to the subkey by 1, otherwise decrease by 1.
– look over the counters, and choose the one with the maximal absolute value.

It is expected that the maximal deviation would be suggested for the right subkey. Moreover, we can take into consideration several subkeys with the highest Absolute values, and increase the success rate (on the expense of running time).

We can use other approximations involving as few as possible S boxes with unknown subkeys (there are many such approximations, even with one new S box), and receive additional bits of subkey material. This way one can find as many as 112 bits of the last round subkey. We now decrypt one round (guessing 16 bits from the last round subkey) and perform attack on 9-round Serpent (which is much faster than attacking 10-round Serpent). Due to time complexity this attack is applicable only on the 192-bit and 256-bit keys variants.

The time complexity for this algorithm is the time needed to decrypt 11 S boxes (out of 320) under 2^{44} keys for 2^{118} ciphertexts. Thus, the time complexity is $2^{118} \cdot 2^{44} \cdot 11/320 \approx 2^{157.14}$ 10-round Serpent encryptions, with data complexity of 2^{118} plaintexts, and memory of 2^{44} counters.

Note that for each ciphertext we look only on 44 bits, and decrypt many times the same value under the same key. These give rise to a better algorithm:

– Initialize array of 2^{45} counters (counters must be at least 118 bits long).
– For each plaintext P calculate the input subset parity, and advance the counter which is related to the parity and the 44 bits from the output of 11 S boxes in the last round.
– Initialize array of 2^{44} counters (counters must be at least 118 bits long).

- For each value of the subkey do the following:
  - Decrypt each possible 44 bits (of ciphertext) and calculate the output subset parity. Increase/decrease the related counter related to subkey according the XOR of the parities (the parity from the first table, and the calculated parity) by the number of plaintext/ciphertext pairs satisfying the parity and the 44 bits of ciphertext.
- look over the counters, and choose the one with the maximal absolute value.

This way, each 44-bit ciphertext value is decrypted only twice under each subkey, and we note that the variant of the algorithm which uses only one decryption is trivial.

Thus, the time complexity of the attack is the time required to perform $2^{45} \cdot 2^{44} = 2^{89}$ decryptions of 11 S boxes (as the first step is just counting the number of occurrences of a bit string), which is equivalent to $2^{89} \cdot 11/320 = 2^{84.13}$ 10-round Serpent encryptions. Again we can use other approximations (there are another 15 approximations with 11 active S boxes in the last round), and again retrieve 112 bits from the last round subkey. For 128-bit keys we can now just guess the remaining 16 bits, and for longer keys we just guess those 16 bits, and decrypt one round to attack 9-round Serpent.

We can also use other linear approximations. There is a linear approximation with bias 2^{-57}, with only 12 active S boxes in the last round. Using this approximation instead, would result in 2^{116} plaintexts needed, with 2^{49} counters, and $2^{49} \cdot 2^{48} = 2^{97}$ times we decrypt 12 S boxes, which is equivalent to $2^{97} \cdot 12/320 = 2^{92.26}$ 10-round Serpent encryptions.

For 192-bit and 256-bit keys we can use the original 9-round linear approximation, counting on 23 S boxes and reducing the required plaintexts to 2^{106}. The memory complexity would be in such a case 2^{93} counters, and the time needed is equivalent to $2^{93} \cdot 2^{92} = 2^{185}$ times decrypting 23 S boxes.

6 Attacking 11-Round 256-Bit Key Serpent

We can attack 11-round Serpent (in rounds 2–12) using a 9-round linear approximation (in rounds 3–11), which is similar to the one presented in Section 5 (the modified approximation with probability $1/2 + 2^{-58}$), besides the first round which is shown in Figure 3. This replacement gives us approximation with bias of 2^{-58} and we reduce by 5 the number of active S boxes in round 2, as this first round of the approximation has only 24 active S boxes in round 2 (instead of 29 in the original approximation) and 11 active S boxes in the last round (instead of 23 in the original approximation).

The basic algorithm in this attack is quite similar to the algorithm from Section 5. Given 2^{118} plaintexts:

- Initialize array of $2^{44+96} = 2^{140}$ counters (counters must be at least 118 bits long).
- For each of the 44-bit subkey value in the 11 S boxes in the last round and 96-bit subkey value in the 24 S boxes in the first round:

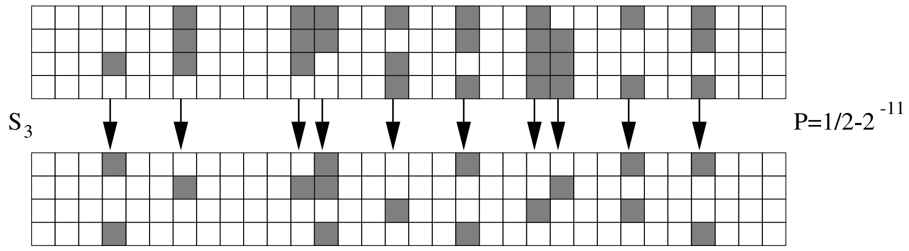


Fig. 3. First Round of the 9-round Linear Approximation used in the 11-round Attack

  - Decrypt each ciphertext through the 11 S boxes in round 12.
  - Encrypt each plaintext through the 24 S boxes in round 2.
  - Calculate the input subset parity and the output subset parity required by the approximation, and calculate the XOR of two. If the XOR value is zero increase the counter related to the subkey by 1, otherwise decrease by 1.
- look over the counters, and choose the one with the maximal absolute value.

The time complexity for this algorithm would therefore be the time needed to encrypt/decrypt 35 S boxes (out of 352) under 2^{140} keys for 2^{118} plaintexts/ciphertexts. Thus, the time complexity is $2^{118} \cdot 2^{140} \cdot 35/352 \approx 2^{255}$ 11-round Serpent encryptions, with data complexity of 2^{118} plaintexts, and memory needed for 2^{140} counters. As one can see this attack is marginally faster than exhaustive search.

We can also use the better algorithm from the previous section. This time we will count according the 96 bits from the plaintext, and take the output subset parity. However, in order to perform the attack, we need to know the parity itself, which we do not know due to the additional round. The solution would be to try for each of the 2^{44} possible subkeys of the last round the improved attack. The algorithm we receive is as follows:

- Initialize array of 2^{44} entries (with entry size of 96+118=214 bits at least)
- For each 44-bit subkey of the last round:
  - decrypt all ciphertext under the subkey, and calculate the output subset parity.
  - Initialize array of 2^{97} counters (counters must be at least 118 bits long).
  - For each plaintext P advance the counter which is related to the output subset parity and the 96 bits from the 24 S boxes.
  - Initialize array of 2^{96} counters (counters must be at least 118 bits long).
  - For each value of the subkey encrypt each possible 96 bits (of plaintext) and calculate the output subset parity. Increase/decrease the related counter related to subkey according the XOR of the parities (the parity from the first table, and the calculated parity) by the number of plaintext/ciphertext pairs satisfying the parity and the 96 bits of plaintext.

    - look over the counters, and choose the one with the maximal absolute value.
    - Keep this 96-bit subkey value for this subkey with the value of the counter in the smaller table.
- Run over the smaller table and look for the entry with the maximal absolute value. The 44-bit subkey related to it, and the 96-bit subkey value stored in the entry has a good chance of being the right ones.

The time complexity for the attack is the time needed to decrypt 2^{118} ciphertexts 11 S boxes under 2^{44} possible subkeys, and for each subkey to run an internal loop which requires $2^{96} \cdot 2^{97} = 2^{193}$ times encrypting 24 S boxes. Thus, the total running time is $2^{44} \cdot (2^{118} \cdot 11/352 + 2^{197}) = 2^{241}$.

Table 2. Known Attacks on Serpent

Rounds	Key Size	Complexity			Source
		Data	Time	Memory	
7	256	2^{122}	2^{248}	2^{126}	[6] - Section 3.5
	all	2^{85}	2^{86} MA	2^{52}	[5] - Section 3
8	192 & 256	2^{128}	2^{163}	2^{133}	[6] - Section 4.2
	192 & 256	2^{110}	2^{175}	2^{115}	[6] - Section 5.3
	256	2^{85}	2^{214} MA	2^{85}	[5] - Section 3.1
9	256	2^{110}	2^{252}	2^{212}	[6] - Section 5.4
10	256	$2^{126.4}$	2^{208}	$2^{131.4}$	[5] - Section 4
	256	$2^{126.4}$	$2^{204.8}$	$2^{195.5}$	[5] - Section 4.1
	all	2^{118}	$2^{89} MA$	2^{45}	This paper
	192 & 256	2^{106}	2^{185}	2^{96}	This paper
11	192 & 256	2^{118}	2^{187}	2^{193}	This paper

MA - Memory Accesses

However, the heavy internal loop (for each of the 2^{97} possible entries and for each 96-bit subkey) can be done only once, and by keeping the result in a large table we reduce the time for each iteration of the external loop to just copy the values in the right place.

Using this better algorithm would require to run once the internal loop (in the cost of 2^{192} times encrypting 24 S boxes, and for each of the 2^{44} last round subkeys to decrypt 2^{118} ciphertext under 11 S boxes, and than access 2^{96} cells in the precomputed table. Thus, the total running time of the attack is equivalent to $2^{192} \cdot 24/352 + 2^{44} \cdot (2^{118} \cdot 11/352 + 2^{96}) = 2^{187}$ 11-round Serpent encryptions. The memory needed for this attack is 2^{193} entries (containing one bit - the input subset parity).

We conclude that this attack is better than exhaustive search for 192-bit and 256-bit keys.

7 Summary

In this paper we performed the first linear cryptanalysis of Serpent, we presented the best known 9-round linear approximation for Serpent with probability $1/2 + 2^{-52}$, and explained our search method.

Using the approximations we have found, we have attacked 10-round Serpent with all bit lengths with time complexity $2^{84.13}$ and 2^{118} known plaintexts. We have also presented an attack on 11-round Serpent with 192-bit and 256-bit keys with time complexity of 2^{187} 11-round Serpent encryptions, with the same data complexity (2^{118}) and 2^{193} memory cells (of one bit).

References

[1] R. Anderson, E. Biham and L. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, NIST AES Proposal 1998.

[2] E. Biham, *A Note on Comparing the AES Candidates*, Second AES Candidate Conference, 1999.

[3] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

[4] E. Biham, O. Dunkelman, N. Keller, *The Rectangle Attack - Rectangling the Serpent*, To appear in proceedings of Eurocrypt 2001. Available on-line at *http://vipe.technion.ac.il/ orrd/crypt/*

[5] O. Dunkelman, *An Analysis of Serpent-p and Serpent-p-ns*, rump session, Second AES Candidate Conference, 1999.

[6] T. Kohno, J. Kelsey and B. Schneier, *Preliminary Cryptanalysis of Reduced-Round Serpent*, Third AES Candidate Conference, 2000.

[7] J. Kelsey, T. Kohno and B. Schneier, *Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent*, FSE 7, to appear.

[8] M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, Eurocrypt 93, Springer Verlag LNCS 765, pp. 386-397.

[9] NIST, *A Request for Candidate Algorithm Nominations for the AES*, available on-line at *http://www.nist.gov/aes/*.