Differential-Linear Cryptanalysis of Serpent*

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

[1]Computer Science Department, Technion.
Haifa 32000, Israel
{biham,orrd}@cs.technion.ac.il
[2]Mathematics Department, Technion.
Haifa 32000, Israel
nkeller@tx.technion.ac.il

Abstract. Serpent is a 128-bit SP-Network block cipher consisting of 32 rounds with variable key length (up to 256 bits long). It was selected as one of the 5 AES finalists. The best known attack so far is a linear attack on an 11-round reduced variant.
In this paper we apply the enhanced differential-linear cryptanalysis to Serpent. The resulting attack is the best known attack on 11-round Serpent. It requires $2^{125.3}$ chosen plaintexts and has time complexity of $2^{139.2}$. We also present the first known attack on 10-round 128-bit key Serpent. These attacks demonstrate the strength of the enhanced differential-linear cryptanalysis technique.

1 Introduction

Serpent [1] is one of the 5 AES [13] finalists. It has a 128-bit block size and accepts key sizes of any length between 0 and 256 bits. Serpent is an SP-Network with 32 rounds and 4-bit to 4-bit S-boxes.

Since its introduction in 1997, Serpent has withstood a great deal of cryptanalytic efforts. In [8] a modified variant of Serpent in which the linear transformation of the round function was modified into a permutation was analyzed. The change weakens Serpent, as this change allows one active S-box to activate only one S-box in the consecutive round. In Serpent, this is impossible, as one active S-box leads to at least two active S-boxes in the following round. The analysis of the modified variant presents an attack against up to 35 rounds of the cipher.

In [9] a 256-bit key variant of 9-round Serpent[1] is attacked using the amplified boomerang attack. The attack uses two short differentials – one for rounds 1–4 and one for rounds 5–7. These two differentials are combined to construct a 7-round amplified boomerang distinguisher, which is then used to mount a key recovery attack on 9-round Serpent. The attack requires 2^{110} chosen plaintexts and its time complexity is 2^{252} 9-round Serpent encryptions.

* The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-1999-12324.

[1] We use n-round Serpent when we mean a reduced version of Serpent with n rounds.

In [4] the rectangle attack is applied to attack 256-bit key 10-round Serpent. The attack is based on an 8-round distinguisher. The distinguisher treats those 8 rounds as composed of two sub-ciphers: rounds 1–4 and rounds 5–8. In each sub-cipher the attack exploits many differentials. These 4-round differentials are combined to create an 8-round rectangle distinguisher. The attack requires $2^{126.8}$ chosen plaintexts and 2^{217} memory accesses[2] which are equivalent to $2^{208.8}$ 10-round Serpent encryptions[3].

The 10-round rectangle attack was improved in [6] and the improved attack requires $2^{126.3}$ chosen plaintexts, with time complexity of $2^{173.8}$ memory accesses (2^{165} 10-round Serpent encryptions). Thus, using the rectangle attack, it is also possible to attack 192-bit key 10-round Serpent. A similar boomerang attack, which requires almost the entire code book is also presented in [6].

The best known attack so far against Serpent can attack up to 11 rounds. The attack [5] is based on linear cryptanalysis [11]. It requires data complexity of 2^{118} known plaintexts and time complexity of 2^{214} memory accesses ($2^{205.7}$ 11-round Serpent encryptions).

In this paper we combine the differential and the linear results on Serpent to present an attack on 11-round Serpent which has a significantly lower time complexity. The attack is based on the differential-linear technique [10]. The technique was later enhanced and improved in [7]. This technique combines a differential characteristic (or several differential characteristics) together with a linear approximation to construct a chosen plaintext distinguisher. This result sheds more light on the applicability and the power of the enhanced differential-linear technique.

The data complexity of our attack is $2^{125.3}$ chosen plaintexts and the time complexity is about $2^{139.2}$ 11-round Serpent encryptions. Therefore, the attack is faster than exhaustive search even for 192-bit keys 11-round Serpent. We use the same techniques to present a 10-round attack on Serpent that requires $2^{107.2}$ chosen plaintexts and $2^{125.2}$ 10-round Serpent encryptions. This is the first known attack on a 128-bit key 10-round Serpent faster than exhaustive search.

We organize this paper as follows: In Section 2 we give the basic description of Serpent. In Section 3 we briefly describe the differential-linear technique. In Section 4 we present the differential-linear attack on 11-round Serpent and on 10-round Serpent. We summarize our results and compare them with previous results on Serpent in Section 5. In the appendices we describe the differential characteristic and the linear approximation which are used in the attacks.

2 A Description of Serpent

In [1] Anderson, Biham and Knudsen presented the block cipher Serpent. It has a block size of 128 bits and accepts 0–256 bit keys. Serpent is an SP-network

[2] In [4] a different number is quoted, but in [6] this mistake was mentioned, and the true time complexity of the algorithm was computed.

[3] The conversion was done according to the best performance figures, presented in [12], assuming one memory access is equivalent to 3 cycles.

block cipher with 32 rounds. Each round is composed of key mixing, a layer of S-boxes and a linear transformation. There is an equivalent bitsliced description which makes the cipher more efficient, and easier to describe.

In our description we adopt the notations of [1] in the bitsliced version. The intermediate value of the round i is denoted by $\hat{B}_i$ (which is a 128-bit value). The rounds are numbered from 0 to 31. Each $\hat{B}_i$ is composed of four 32-bit words X_0, X_1, X_2, X_3.

Serpent has 32 rounds, and 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. Thus, the first copy of S_0 takes the least significant bits from X_0, X_1, X_2, X_3 and returns the output to the same bits. This can be implemented as a boolean expression of the 4 words.

The set of eight S-boxes is used four times. S_0 is used in round 0, S_1 is used in round 1, etc. After using S_7 in round 7 we use S_0 again in round 8, then S_1 in round 9, and so on. In the last round (round 31) the linear transformation is omitted and another key is XORed.

The cipher may be formally described by the following equations:

$$\begin{aligned} \hat{B}_0 &:= P \\ \hat{B}_{i+1} &:= R_i(\hat{B}_i) \qquad i = 0, \ldots, 31 \\ C &:= \hat{B}_{32} \end{aligned}$$

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 $S_{(i \bmod 8)}$ thirty two times in parallel, and LT is the linear transformation.

Given the four 32-bit words $X_0, X_1, X_2, X_3 := \hat{\mathcal{S}}_i(\hat{B}_i \oplus \hat{K}_i)$, they are linearly mixed by the following linear transformation:

$$\begin{aligned} X_0 &:= X_0 <<< 13 \\ X_2 &:= X_2 <<< 3 \\ X_1 &:= X_1 \oplus X_0 \oplus X_2 \\ X_3 &:= X_3 \oplus X_2 \oplus (X_0 << 3) \\ X_1 &:= X_1 <<< 1 \\ X_3 &:= X_3 <<< 7 \\ X_0 &:= X_0 \oplus X_1 \oplus X_3 \\ X_2 &:= X_2 \oplus X_3 \oplus (X_1 << 7) \\ X_0 &:= X_0 <<< 5 \\ X_2 &:= X_2 <<< 22 \\ \hat{B}_{i+1} &:= X_0, X_1, X_2, X_3 \end{aligned}$$

where $<<<$ denotes bit rotation to the left, and $<<$ denotes bit shift to the left.

The key scheduling algorithm of Serpent is defined for 256-bit keys. Shorter keys are padded by a single bit of 1 followed by as many bits of 0's required to have a total length of 256 bits. This value is loaded into a linear feedback shift register, that outputs blocks of 128 bits. Each block passes through a layer of S-boxes (different layer for each block). This process is repeated until 33 subkeys of 128 bits each are derived. The subkeys are linearly independent, but knowing the subkey which enters an S-box (for any round, for any S-box), we can invert the relevant S-box used in the key schedule and obtain 4 linear equations on the key.

3 Differential-Linear Cryptanalysis

Differential cryptanalysis [3] analyzes ciphers by studying the development of differences during encryption. The attack is a chosen plaintext attack based on a differential distinguisher which uses pairs of plaintexts. The distinguisher exploits the fact that for the attacked cipher, the probability that an input difference Ω_P (i.e., difference between two inputs) results in an output difference Ω_T is higher than for a random permutation. Linear cryptanalysis [11] analyzes the cipher by studying approximate linear relations. The attack is based on building a distinguisher which exploits the fact that for the attacked cipher, the probability that a given input mask λ_P and a given output mask λ_T are related is different than 1/2 (the probability for a random permutation).

In 1994, Langford and Hellman [10] showed that both kinds of analysis can be combined together by a technique called *differential-linear cryptanalysis*. The attack uses a differential characteristic that induces a linear relation between two intermediate encryption values with probability one. In [7] Biham, Dunkelman and Keller extended this technique to the cases where the probability of the differential part is smaller than 1.

We use notations based on [2, 3] for differential and linear cryptanalysis, respectively. In our notations Ω_P, Ω_T are the input and output differences of the differential characteristic, and λ_P, λ_T are the input and output subsets (denoted by bit masks) of the linear characteristic.

Let E be a block cipher which can be described as a cascade of two sub-ciphers – E_0 and E_1, i.e., $E = E_1 \circ E_0$. Langford and Hellman suggested to use a truncated differential $\Omega_P \rightarrow \Omega_T$ for E_0 with probability 1. To this differential they concatenate a linear approximation $\lambda_P \rightarrow \lambda_T$ with probability $1/2 + q$ or with bias q. Their attack requires that the bits masked in λ_P have a constant and known difference in Ω_T.

If we take a pair of plaintexts P_1 and P_2 that satisfy $P_1 \oplus P_2 = \Omega_P$, then after E_0, $\lambda_P \cdot E_0(P_1) = \lambda_P \cdot E_0(P_2)$ (or the opposite if the scalar product $\Omega_T \cdot \lambda_P$ is 1). This follows from the fact that $E_0(P_1)$ and $E_0(P_2)$ have a fixed difference in the masked bits according to the output difference.

Recall that the linear approximation predicts that $\lambda_P \cdot P = \lambda_T \cdot E_1(P)$ with probability $1/2+q$. Hence, $\lambda_P \cdot E_0(P_1) = \lambda_T \cdot E_1(E_0(P_1))$ with probability $1/2+q$, and $\lambda_P \cdot E_0(P_2) = \lambda_T \cdot E_1(E_0(P_2))$ with probability $1/2 + q$. As the differential predicts that $\lambda_P \cdot E_0(P_1) = \lambda_P \cdot E_0(P_2)$, then with probability $1/2 + 2q^2$, $\lambda_T \cdot C_1 = \lambda_T \cdot C_2$ where C_1 and C_2 are the corresponding ciphertexts of P_1 and P_2, respectively (i.e., $C_i = E_1(E_0(P_i))$.

This fact allows constructing differential-linear distinguishers, based on encrypting many plaintext pairs, and checking whether the ciphertexts agree on the parity of the output subset. The data complexity of the distinguishers is $O(q^{-4})$ chosen plaintexts, when the exact number of plaintexts is a function of the desired success rate, and of the number of possible subkeys.

In [7] Biham, Dunkelman and Keller proposed a way to deal with differentials with probability $p \neq 1$. In case the differential is satisfied (probability p), the above analysis remains valid. In case the differential is not satisfied (probability $1-p$) we assume a random behavior of the input subset parities (and therefore, also the output subset parities). The probability that a pair with input difference Ω_P will satisfy $\lambda_T \cdot C_1 = \lambda_T \cdot C_2$ is in that case $p(1/2 + 2q^2) + (1-p) \cdot 1/2 = 1/2 + 2pq^2$.

Furthermore, in [7] it was shown that the attack can still be applicable if the product $\Omega_T \cdot \lambda_P = 1$. In this case, the analysis remains valid, but instead of looking for the instances for which $\lambda_T \cdot C_1 = \lambda_T \cdot C_2$, we look for the cases when $\lambda_T \cdot C_1 \neq \lambda_T \cdot C_2$. As the analysis remains the same given a pair of plaintexts with the input difference Ω_P, the probability that the pair disagrees on the output subset parity is $1/2 + 2pq^2$. Another interesting result is that we can use the attack even when $\Omega_T \cdot \lambda_P$ is unknown, as long as it remains constant (this is the case, when the difference passes an unknown constant linear function). The data complexity of the enhanced differential-linear attack is $O(p^{-2}q^{-4})$.

4 Attacking 11-Round Serpent

We now present our differential-linear attack on 11-round Serpent. The 11 rounds that we attack are rounds 1–11 of Serpent (i.e., starting at the second round of Serpent). The attack is based on building a 9-round differential-linear distinguisher for rounds 2–10, and retrieving 48 bits of the subkeys of rounds 1 and 11.

The 9-round differential-linear characteristic is based on a 3-round differential with probability 2^{-6} and a 6-round linear approximation with a bias of 2^{-27}. There are 5 active S-boxes before the 3-round differential, and 7 active S-boxes after the 6-round linear approximation.

In the key recovery attack we are only interested in the input difference of the differential and in the output mask of the linear approximation. To present these values we adopt the notations from [4, 5, 9]. The figures describe data blocks by rectangles of 4 rows and 32 columns. The rows are the bitsliced 32-bit words, and each column forms the input of an S-box. The upper row represents X_0, while the lower row represents X_3, and the rightmost column represents the least significant bits of the words.

In the description of differentials a thin arrow represents a probability of 1/8 for the specific S-box (given the input difference, the output difference is

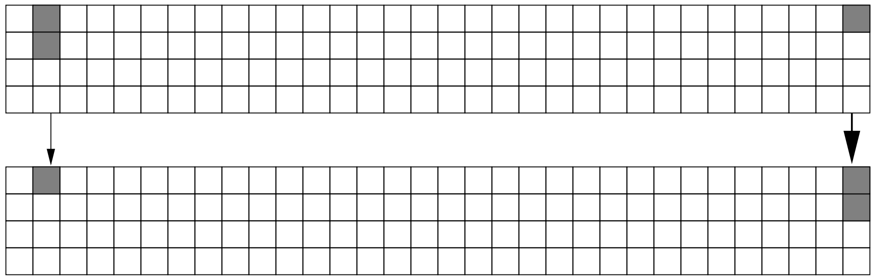


Fig. 1. Differential and Linear Representation Example

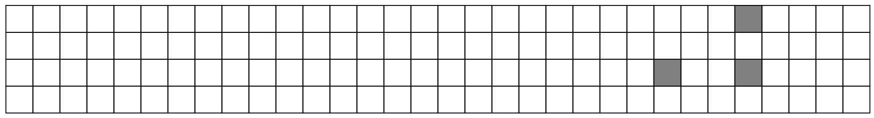


Fig. 2. The Input Difference of the 9-Round Distinguisher

achieved with probability 1/8), and a fat arrow stands for probability 1/4. If there is a difference in a bit, its entry is filled. An example for our notation can be found in Figure 1. The input difference 1 in the first S-box (S-box 0; related to the least significant bits of the 4 words X_0, X_1, X_2, and X_3) causes an output difference 3 with probability 1/4, and input difference 3 in S-box 30 causes an output difference 1 with probability 1/8.

In the description of linear approximations a filled entry represents the subset of bits participating in the approximation, a thin arrow represents a bias of 1/8, and a fat arrow represent a bias of 1/4. Thus, we can treat Figure 1 as an example for a linear approximation: Bit 0 of the input of S-box 0 has the same parity as the subset $\{0, 1\}$ of the S-box's output with probability $1/2 \pm 1/4$ (or a bias of 1/4), and the parity of bits 0 and 1 of the input of S-box 30 has the same value as bit 1 of the output of the S-box with probability $1/2 \pm 1/8$ (a bias of 1/8).

The 3-round differential is based on the best known 4-round differential characteristic of Serpent, taken from [4]. It starts in round 2 with two active S-boxes, and ends after round 4. Note, that we could have taken a differential with higher probability of 2^{-5}, but in exchange we would get 9 active S-boxes in the round before the distinguisher, instead of only 5. When we experimentally verified this result we have found out that there are other differentials which also predict the difference in the bits of λ_P. Summing all these differentials, we get that the probability that $\lambda_P \cdot \Omega_T = 0$ is $1/2 + 2^{-7}$. Hence, we use in our attack and analysis $p = 2^{-7}$. We present the input difference in Figure 2.

The 6-round linear approximation is based on one of the two best known 6-round linear approximations of Serpent, taken from [5]. It starts in round 5, with 2 active S-boxes, and ends in round 10 with 5 active S-boxes and has a bias of 2^{-27}. The output masks can be computed once 7 S-boxes in round 11 are

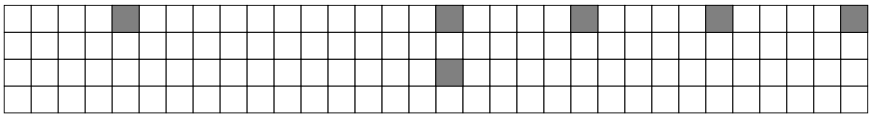


Fig. 3. The Output Mask of the 9-Round Distinguisher

partially decrypted. We present the output mask of the linear approximation in Figure 3.

In the basic 11-round attack, the attacker encrypts many plaintext pairs with a plaintext difference that might lead to the desired input difference, and partially decrypts the ciphertext pairs. Then, the attacker checks whether the partially decrypted values agree on the parity of the output mask. As stated before, the probability that the partially decrypted ciphertexts agree on the output subset mask is $1/2+2pq^2$. Therefore, if the attack uses N plaintext pairs it is expected that about $N(1/2+2pq^2)$ of them agree on the parity of the output subset.

For the analysis stage of the attack, we define a random variable for each subkey candidate. The variable counts how many pairs agree on the parity of the output mask after the partial decryption. As there are 2^{48} possible subkeys in the 12 S-boxes, there are $2^{48}-1$ wrong subkeys. We assume that all random variables related to the wrong subkeys behave like a normal random variable with a mean value of $N/2$ and a variance of $N/4$. The right subkey's random variable is also a normal random variable, but with mean of $N(1/2+2pq^2)$ and a variance of about $N/4$.

The attack succeeds if the random variable related to the right subkey has the highest value (or among the top fraction). Our analysis shows that for $2^{124.3}$ pairs, the random variable related to the right subkey has probability 72.1% to be the highest of all random variables.

To optimize the data and time complexity of the attack, we use structures of chosen plaintexts. Each structure contains 2^{20} chosen plaintexts, which vary on the input of the 5 active S-boxes in round 1. We use the following algorithm for the attack:

1. Select $N = 2^{125.3}$ plaintexts, consisting of $2^{105.3}$ structures, each is chosen by selecting:
   (a) Any plaintext P_0.
   (b) The plaintexts $P_1,\ldots,P_{2^{20}-1}$ which differ from P_0 by all the $2^{20}-1$ possible (non-empty) subsets of the twenty bits which enter the 5 active S-boxes in round 1.
2. Request the ciphertexts of these plaintext structures (encrypted under the unknown key K).
3. For each value of the 20 bits of $K1$ entering these 5 S-boxes:
   (a) Initialize an array of 2^{28} counters to zeroes.
   (b) Partially encrypt each plaintext the 5 active S-boxes in round 1 and find the pairs which satisfy the difference Ω_P before round 2.

(c) Given those $2^{124.3}$ pairs, perform for each ciphertext pair:
   i. Try all the 2^{28} possible values of the 28 bits of subkey $K11$ that enter the 7 active S-Boxes in round 11.
   ii. For each value of the subkey, partially decrypt the ciphertexts through the 7 active S-boxes in round 11, and compute the parity of the subset of bits in λ_T after round 10.
   iii. If the parities in both members of the pair are equal, increment the counter in the array related to the 28 bits of the subkey.

(d) The highest entry in the array should correspond to the 28 bits of $K11$ entering the 7 active S-boxes in round 11.

4. Each trial of the key gives us $20 + 28 = 48$ bits of the subkeys (20 bits in round 1 and 28 bits in round 11), along with a measure for correctness. The correct value of the 48 bits is expected to be the most frequently suggested value (with more than 72.1% success rate).
5. The rest of the key bits are then recovered by auxiliary techniques.

The time complexity of a naive implementation is $2^{125.3} \cdot 2^{48} \cdot 12/352 = 2^{172.4}$ 11-round Serpent encryptions. The memory requirements of this algorithm are mostly for keeping the plaintexts, but we can handle each structure independently from other structures, thus, 2^{28} counters (of 20 bits each) would suffice. Hence, the memory complexity of the attack is 2^{30} bytes.

The time complexity of the attack can be improved to $2^{125.3} \cdot 2^{20} \cdot 5/352 = 2^{139.2}$ 11-round Serpent encryptions. We note that for each guess of the subkey of round 11 we perform $2^{125.3}$ decryptions. However, there are only 2^{28} possible values of the 28 bits which we actually decrypt. The improvement is based on keeping a precomputed table that holds for any possible value of the 28 ciphertext bits (which enter the 7 active S-boxes) and any possible value of the corresponding 28-bit subkey, the parity of the partially decrypted value. The optimized Step 3(b) of the attack counts over all pairs how many times each of the 2^{56} possibilities of the 56 bits (28 bits from each of the two ciphertexts) entering the 7 active S-boxes n round 11 occurs. After counting the occurrences, we find how many pairs agree on the output subset parity, and how many disagree, for this subkey guess.

This is the first theoretical attack on 11-round Serpent with 192-bit keys. It requires $2^{125.3}$ chosen plaintexts, and has time complexity of $2^{139.2}$ 11-round Serpent encryptions. The memory requirement of the attack is 2^{60} bytes of RAM.

We can attack 10-round Serpent by reducing the distinguisher to 8 rounds. We remove the last round of the linear approximation to get a 5-round linear approximation with bias $q = 2^{-22}$. The data complexity of the attack drops to $2^{107.2}$ chosen plaintexts. The time complexity of the attack in this case is $2^{107.2} \cdot 2^{20} \cdot 5/320 = 2^{125.2}$ 10-round Serpent encryptions. This is the first known attack against 10-round Serpent with 128-bit key. Note, that in this attack we retrieve only 40 subkey bits, as there are only 5 active S-boxes in the last round (round 10). The rest of the bits can be found by exhaustive search with time complexity of $2^{128-40} = 2^{88}$ 10-round Serpent encryptions.

5 Summary

In this paper we present the best published attack on 11-round Serpent. The attack is applicable against up to 11-round Serpent with keys of sizes 140–256 bits. It is faster than exhaustive search, and has a success rate of 72.1%. The attack requires $2^{125.3}$ chosen plaintexts. Its best time complexity is $2^{139.2}$ 11-round Serpent encryptions, using 2^{60} bytes of RAM for the analysis. We also present an attack on 10-round Serpent requiring $2^{107.2}$ chosen plaintexts and whose time complexity is $2^{125.2}$ 10-round Serpent encryptions. We summarize our new attacks, and previously published attacks against Serpent in Table 1.

Rounds	Type of Attack	Key Sizes	Complexity		
			Data	Time	Memory
6	Differential [9]	all	2^{83}CP	2^{90}	2^{44}
	Differential [9]	all	2^{71}CP	2^{103}	2^{79}
	Differential [9]	192 & 256	2^{41}CP	2^{163}	2^{49}
7	Differential [9]	256	2^{122}CP	2^{248}	2^{130}
	Differential [4]	all	2^{84}CP	$2^{78.9}$	2^{56}
8	Amp. Boomerang [9]	192 & 256	2^{128}CP	2^{163}	2^{137}
	Amp. Boomerang [9]	192 & 256	2^{110}CP	2^{175}	2^{119}
	Differential [4]	256	2^{84}CP	$2^{206.7}$	2^{89}
9	Amp. Boomerang [9]	256	2^{110}CP	2^{252}	2^{212}
10	Rectangle [4]	256	$2^{126.8}$CP	$2^{207.4}$	$2^{131.8}$
	Rectangle [4]	256	$2^{126.8}$CP	2^{205}	2^{196}
	Rectangle [6]	192&256	$2^{126.3}$CP	2^{165}	$2^{131.8}$
	Boomerang [6]	192&256	$2^{126.3}$ACPC	2^{165}	2^{89}
	Enhanced Diff.-Lin. (this paper)	all	$2^{105.2}$CP	$2^{123.2}$	2^{40}
11	Linear [5]	256	2^{118}KP	$2^{205.7}$	2^{183}
	Enhanced Diff.-Lin. (this paper)	192 & 256	$2^{125.3}$CP	$2^{172.4}$	2^{30}
	Enhanced Diff.-Lin. (this paper)	192 & 256	$2^{125.3}$CP	$2^{139.2}$	2^{60}

Complexity is measured in encryption units. Memory is measured in bytes.
CP - Chosen Plaintexts, KP - Known Plaintexts,
ACPC - Adaptive Chosen Plaintexts and Ciphertexts.

Table 1. Summary of Attacks on Serpent with Reduced Number of Rounds

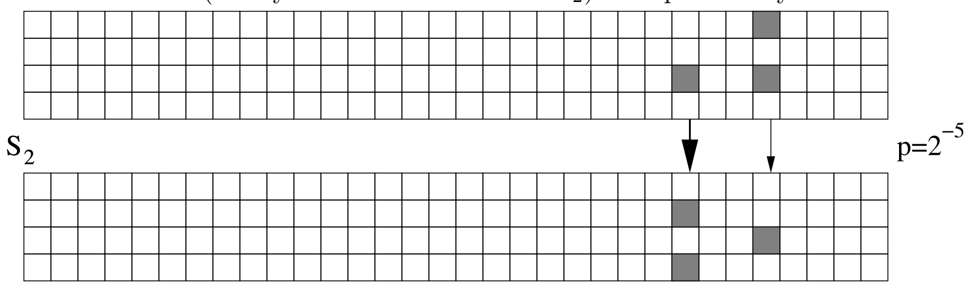
References

1. Ross Anderson, Eli Biham, Lars R. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, NIST AES Proposal, 1998.
2. Eli Biham, *On Matsui's Linear Cryptanalysis*, Advances in Cryptology, proceeding of EUROCRYPT 1994, Lecture Notes in Computer Science 950, pp. 341–355, Springer-Verlag, 1994.

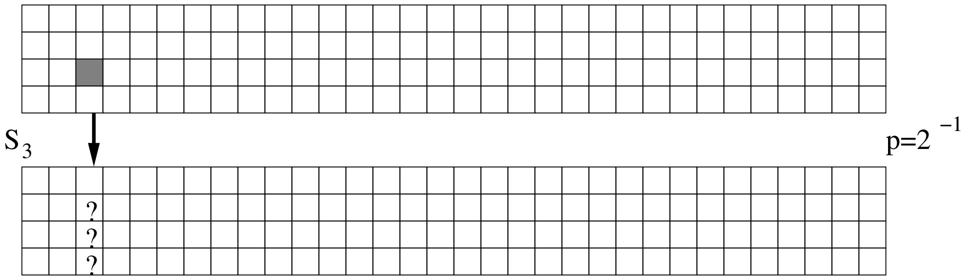
3. Eli Biham, A Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
4. Eli Biham, Orr Dunkelman, Nathan Keller, *The Rectangle Attack – Rectangling the Serpent*, Advances in Cryptology, proceeding of EUROCRYPT 2001, Lecture Notes in Computer Science 2045, pp. 340–357, Springer-Verlag, 2001.
5. Eli Biham, Orr Dunkelman, Nathan Keller, *Linear Cryptanalysis of Reduced Round Serpent*, proceedings of Fast Software Encryption 8, Lecture Notes in Computer Science 2355, pp. 16–27, Springer-Verlag, 2002.
6. Eli Biham, Orr Dunkelman, Nathan Keller, *New Results on Boomerang and Rectangle Attacks*, proceeding of Fast Software Encryption 9, Lecture Notes in Computer Science 2365, pp. 1–16, Springer-Verlag, 2002.
7. Eli Biham, Orr Dunkelman, Nathan Keller, *Enhancing Differential-Linear Cryptanalysis*, Advances in Cryptology, proceeding of ASIACRYPT 2002, Lecture Notes in Computer Science 2501, pp. 254–266, Springer-Verlag, 2002.
8. Orr Dunkelman, *An Analysis of Serpent-p and Serpent-p-ns*, presented at the rump session of the Second AES Candidate Conference, 1999. Available on-line at *http://vipe.technion.ac.il/∼orrd/crypt/*.
9. John Kelsey, Tadayoshi Kohno, Bruce Schneier, *Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent*, proceedings of Fast Software Encryption 7, Lecture Notes in Computer Science 1978, pp. 75–93, Springer-Verlag, 2001.
10. Suzan K. Langford, Martin E. Hellman, *Differential-Linear Cryptanalysis*, Advances in Cryptology, proceedings of CRYPTO '94, Lecture Notes in Computer Science 839, pp. 17–25, 1994.
11. Mitsuru Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology, proceedings of EUROCRYPT '93, Lecture Notes in Computer Science 765, pp. 386–397, 1994.
12. NESSIE, *Performance of Optimized Implementations of the NESSIE Primitives*, NES/DOC/TEC/WP6/D21/a, available on-line at *http://www.nessie.eu.org/nessie*.
13. NIST, *A Request for Candidate Algorithm Nominations for the AES*, available on-line at *http://www.nist.gov/aes/*.
14. David Wagner, *The Boomerang Attack*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 156–170, 1999.

A The Differential Characteristic

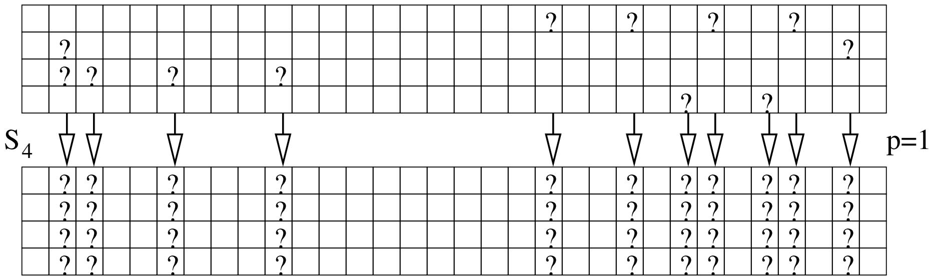
The 3-round truncated differential used in the attack is based on the first 3 rounds of the best 4-round differential of Serpent. The first round of the differential is round 2 (or any other round that uses S_2) with probability 2^{-5}:



After the linear transformation and the application of S_3 we get the following truncated differential with probability 2^{-1}:



Where the '?' means that we do not care what the value of the difference is, and the bold and thick arrow means that this happens with probability 1/2. After the linear transformation, we get the following truncated differential in the input to S_4:

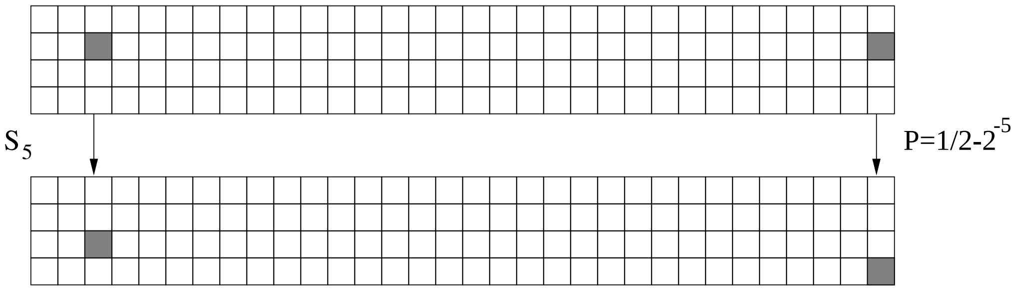


We checked all the various outputs of this S_4 and found that all the possible output differences do not affect the linear approximation used in the attack. The empty arrow means that this holds with probability 1.

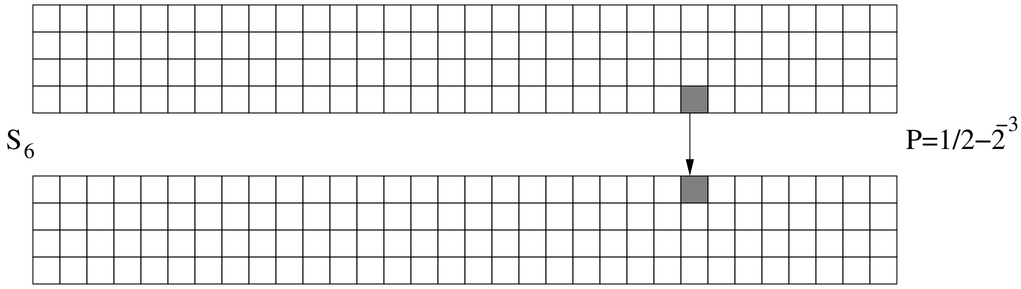
As mentioned before, we have experimentally verified that with probability $1/2+2^{-7}$ this input difference causes even difference (no difference or difference in even number of bits) in the masked bits of λ_P.

B The Linear Approximation

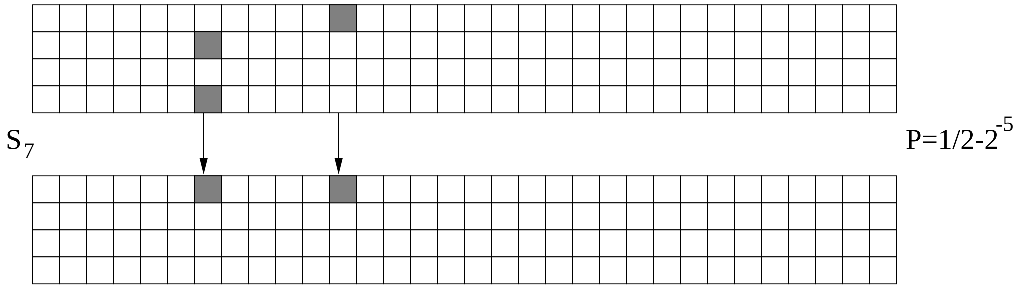
The 6-round linear approximation used in the attack is part of the best 9-round linear approximation of Serpent. It starts in a round with S_5 as the S-box:



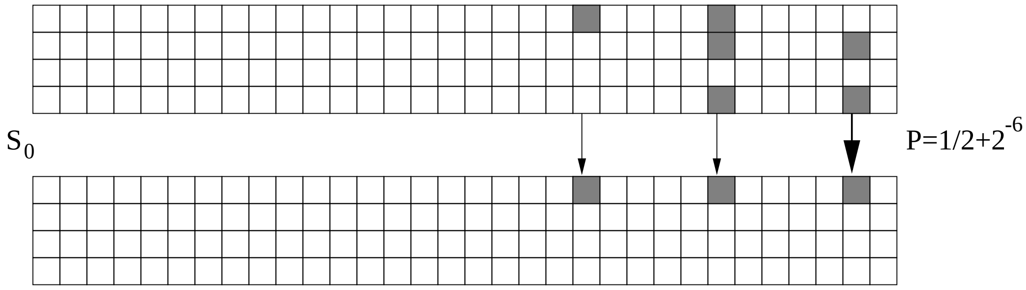
After the linear transformation, and the application of S_6 we get the following linear approximation with bias of 2^{-3}:



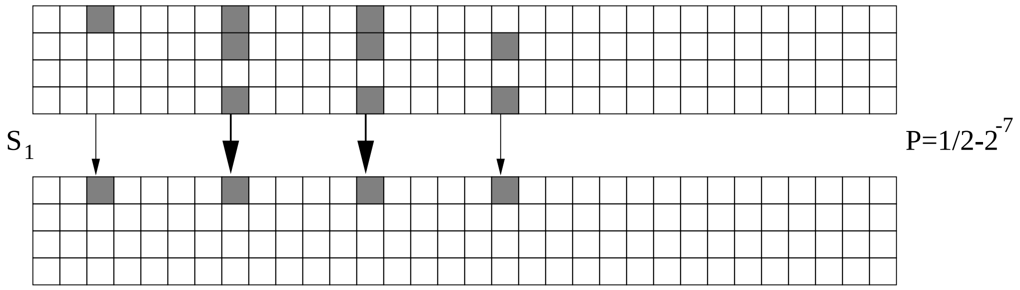
After the linear transformation, and the application of S_7 we get the following linear approximation with bias of 2^{-5}:



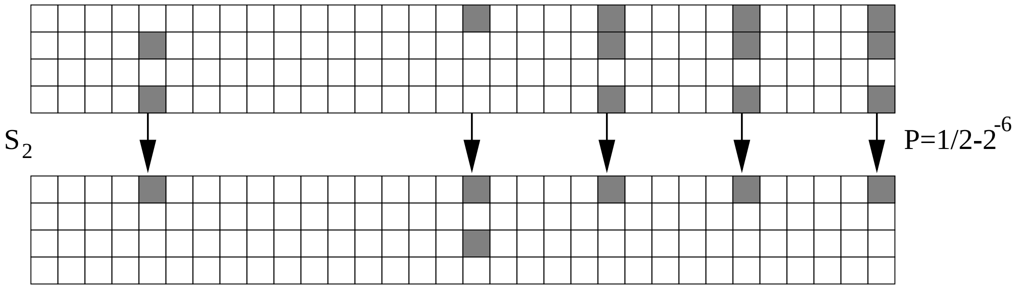
After the linear transformation, and the application of S_0 we get the following linear approximation with bias of 2^{-6}:



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 this round, there are 7 active S-boxes in the following round: S-boxes 1, 8, 11, 13, 18, 23 and 28.