Preliminary Cryptanalysis of Reduced-Round Serpent

Tadayoshi Kohno[1⋆], John Kelsey[2], and Bruce Schneier[2]

[1] Reliable Software Technologies
kohno@rstcorp.com
[2] Counterpane Internet Security, Inc.
{kelsey,schneier}@counterpane.com

Abstract. Serpent is a 32-round AES block cipher finalist. In this paper we present several attacks on reduced-round variants of Serpent that require less work than exhaustive search. We attack six-round 256-bit Serpent using the meet-in-the-middle technique, 512 known plaintexts, 2^{246} bytes of memory, and approximately 2^{247} trial encryptions. For all key sizes, we attack six-round Serpent using standard differential cryptanalysis, 2^{83} chosen plaintexts, 2^{40} bytes of memory, and 2^{90} trial encryptions. We present boomerang and amplified boomerang attacks on seven- and eight-round Serpent, and show how to break nine-round 256-bit Serpent using the amplified boomerang technique, 2^{110} chosen plaintexts, 2^{212} bytes of memory, and approximately 2^{252} trial encryptions.

1 Introduction

Serpent is an AES-candidate block cipher invented by Ross Anderson, Eli Biham, and Lars Knudsen [ABK98], and selected by NIST as an AES finalist. It is a 32-round SP-network with key lengths of 128 bits, 192 bits, and 256 bits. Serpent makes clever use of the bitslice technique to make it efficient in software. However, because of its conservative design and 32 rounds, Serpent is still three times slower than the fastest AES candidates [SKW+99].

In the Serpent submission document [ABK98], the authors give upper bounds for the best differential characteristics through the cipher. However, no specific attacks on reduced-round versions of the cipher are presented. In this paper we consider four kinds of attacks on reduced-round variants of Serpent: differential [BS93], boomerang [Wag99], amplified boomerang [KKS00], and meet-in-the-middle. To the best of our knowledge, these are the best published attacks against reduced-round versions of Serpent.[1]

The current results on Serpent are as follows (see Table 1):

1. A meet-in-the-middle attack on Serpent reduced to six rounds, requiring 512 known plaintext/ciphertext pairs, 2^{246} bytes of random-access memory, and work equivalent to approximately 2^{247} six-round Serpent encryptions.

⋆ Part of this work was done while working for Counterpane Internet Security, Inc.

[1] Dunkelman cryptanalyzed a Serpent variant with a modified linear transformation in [Dun99].

Rounds	Key Size	Complexity			Comments
		[Data]	[Work]	[Space]	
—	—	—	—	—	no previous results
6	256	512 KP	2^{247}	2^{246}	meet-in-the-middle (§6)
6	all	2^{83} CP	2^{90}	2^{40}	differential (§3.2)
6	all	2^{71} CP	2^{103}	2^{75}	differential (§3.3)
6	192 & 256	2^{41} CP	2^{163}	2^{45}	differential (§3.4)
7	256	2^{122} CP	2^{248}	2^{126}	differential (§3.5)
8	192 & 256	2^{128} CPC	2^{163}	2^{133}	boomerang (§4.2)
8	192 & 256	2^{110} CP	2^{175}	2^{115}	amp. boomerang (§5.3)
9	256	2^{110} CP	2^{252}	2^{212}	amp. boomerang (§5.4)

KP — known plaintext, CP — chosen plaintext, CPC — chosen plaintext/ciphertext.

Table 1. Summary of attacks on Serpent. Work is measured in trial encryptions; space is measured in bytes.

2. A differential attack on Serpent reduced to six rounds, requiring 2^{83} chosen plaintexts, 2^{40} bytes of sequential-access memory, and work equivalent to approximately 2^{90} six-round Serpent encryptions.
3. A differential filtering attack on Serpent reduced to seven rounds, requiring 2^{122} chosen plaintexts, 2^{126} bytes of sequential-access memory, and work equivalent to approximately 2^{248} six-round Serpent encryptions.
4. A boomerang attack on Serpent reduced to eight rounds, requiring all 2^{128} plaintext/ciphertext pairs under a given key, 2^{133} bytes of random-access memory, and work equivalent to approximately 2^{163} eight-round Serpent encryptions.[2]
5. An amplified-boomerang key-recovery attack on Serpent reduced to eight rounds, requiring 2^{110} chosen plaintexts, 2^{115} bytes of random-access memory, and work equivalent to approximately 2^{175} eight-round Serpent encryptions.
6. An amplified-boomerang key-recovery attack on Serpent reduced to nine rounds, requiring 2^{110} chosen plaintexts, 2^{212} bytes of random-access memory, and work equivalent to approximately 2^{252} nine-round Serpent encryptions.

The remainder of this paper is organized as follows: First, we discuss the internals of Serpent and explain the notation we use in this paper. We then use differential, boomerang, and amplified boomerang techniques to break up to nine rounds of Serpent. Subsequently we discuss a six-round meet-in-the-middle attack on Serpent. We then discuss some observations on the Serpent key schedule. We conclude with a discussion of our results and some open questions.

[2] Because this eight-round boomerang attack requires the entire codebook under a single key, one can consider this attack a glorified distinguisher that also recovers the key.

2 Description of Serpent

In this document we consider only the bitsliced version of Serpent. The bitsliced and non-bitsliced versions of Serpent are functionally equivalent; the primary difference between the bitsliced and non-bitsliced versions of Serpent are the order in which the bits appear in the intermediate stages of the cipher. Full details of the bitsliced and non-bitsliced version of Serpent are in [ABK98].

2.1 The Encryption Process

Serpent is a 32-round block cipher operating on 128-bit blocks. In the bitsliced version of Serpent, one can consider each 128-bit block as the concatenation of four 32-bit words.

Let B_i represent Serpent's intermediate state prior to the ith round of encryption. Notice that $B_0 = P$ and $B_{32} = C$, where P and C are the plaintext and ciphertext, respectively.

Let K_i represent the 128-bit ith round subkey and let S_i represent the application of the ith round S-box. Let L be Serpent's linear transformation. Then the Serpent round function is defined as:

$$\begin{aligned} X_i &\leftarrow B_i \oplus K_i \\ Y_i &\leftarrow S_i(X_i) \\ B_{i+1} &\leftarrow L(Y_i) \qquad i = 0, \ldots, 30 \\ B_{i+1} &\leftarrow Y_i \oplus K_{i+1} \; i = 31 \end{aligned}$$

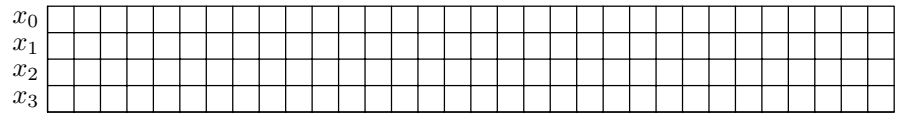
Serpent uses eight S-boxes $S_0, \ldots, S_7$. The indices to S are reduced modulo 8; i.e., $S_0 = S_8 = S_{16} = S_{24}$. The Serpent S-boxes take four input bits and produce four output bits. Consider the application of an S-box S_i to the 128 bit block X_i. Serpent first separates X_i into four 32-bit words x_0, x_1, x_2, and x_3. For each of the 32-bit positions, Serpent constructs a nibble from the corresponding bit in each of the four words, with the bit from x_3 being the most significant bit. Serpent then applies the S-box S_i to the constructed nibble and stores the result in the respective bits of $Y_i = (y_0, y_1, y_2, y_3)$.

The linear transform L on $Y_i = (y_0, y_1, y_2, y_3)$ is defined as

$$\begin{aligned} y_0 &\leftarrow y_0 \lll 13 \\ y_2 &\leftarrow y_2 \lll 3 \\ y_1 &\leftarrow y_0 \oplus y_1 \oplus y_2 \\ y_3 &\leftarrow y_2 \oplus y_3 \oplus (y_0 \ll 3) \\ y_1 &\leftarrow y_1 \lll 1 \\ y_3 &\leftarrow y_3 \lll 7 \\ y_0 &\leftarrow y_0 \oplus y_1 \oplus y_3 \\ y_2 &\leftarrow y_2 \oplus y_3 \oplus (y_1 \ll 7) \\ y_0 &\leftarrow y_0 \lll 5 \\ y_2 &\leftarrow y_2 \lll 22 \\ B_{i+1} &\leftarrow (y_0, y_1, y_2, y_3) \end{aligned}$$

where $\lll$ denotes a left rotation and $\ll$ denotes a left shift.

When discussing the internal state of the Serpent, we will often refer to diagrams such as



where X_i is the internal state under inspection and $X_i = (x_0, x_1, x_2, x_3)$. As suggested by this diagram, we will occasionally refer to an active S-box as a "column."

2.2 The Key Schedule

Serpent's key schedule can accept key sizes up to 256 bits. If a 256-bit key is used, Serpent sets the eight 32-bit words $w_{-8}, w_{-7}, \ldots, w_{-1}$ to the key. If not, the key is converted to a 256-bit key by appending a '1' bit followed by a string of '0's.

Serpent computes the prekeys $w_0, w_1, \ldots, w_{131}$ using the recurrence

$$w_i \leftarrow (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) \lll 11$$

where ϕ is `0x9e3779b9`.

Serpent then computes the 128-bit subkeys K_j by applying an S-box to the prekeys $w_{4j}, \ldots, w_{4j+3}$:

$$\begin{aligned}
K_0 &\leftarrow S_3(w_0, w_1, w_2, w_3) \\
K_1 &\leftarrow S_2(w_4, w_5, w_6, w_7) \\
K_2 &\leftarrow S_1(w_8, w_9, w_{10}, w_{11}) \\
K_3 &\leftarrow S_0(w_{12}, w_{13}, w_{14}, w_{15}) \\
K_4 &\leftarrow S_7(w_{16}, w_{17}, w_{18}, w_{19}) \\
&\vdots \\
K_{31} &\leftarrow S_4(w_{124}, w_{125}, w_{126}, w_{127}) \\
K_{32} &\leftarrow S_3(w_{128}, w_{129}, w_{130}, w_{131})
\end{aligned}$$

3 Differential Cryptanalysis

Differential cryptanalysis, first publicly discussed by Biham and Shamir [BS93], is one of the most well-known and powerful cryptanalytic techniques. Although the original Serpent proposal provided theoretical upper bounds for the highest probability characteristics through reduced-round Serpent variants [ABK98], the Serpent proposal did not present any empirical results describing how successful differential cryptanalysis would be against Serpent in practice.

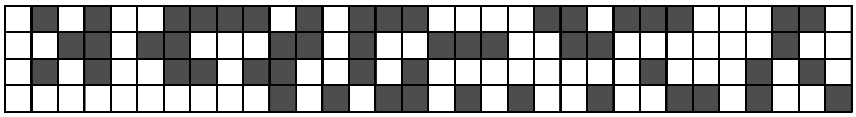
In this section we consider actual differential attacks against reduced-round Serpent variants. Although there may exist other high-probability differentials through several rounds of Serpent, we focus on a particular five-round characteristic, $B'_1 \rightarrow Y'_5$, with probability $p = 2^{-80}$. This characteristic spans Serpent's second through sixth rounds (rounds $i = 1, \ldots, 5$). For completeness, this characteristic is illustrated in Appendix A.1. Notationally, we use X' to represent the XOR difference between two values X and X^*.

3.1 Basic Six-Round Differential Attack

We can use the above-mentioned five-round, probability 2^{-80}, characteristic to attack rounds one through six of 192- and 256-bit Serpent.

To sketch our attack: we request 2^{82} plaintext pairs with an input difference B'_1. For each last round subkey guess, we initialize a count variable to zero. Then, for each unfiltered pair, we peel off the last round and look for our expected output difference from the fifth round. If we observe our expected output difference, we increment our counter. If we count three or more right pairs, we note this subkey as likely to be correct.

If we apply the linear transformation L to the intermediate difference Y'_5 (Appendix A.1), we get the following expected input difference to the sixth round:

B'_6 

We can immediately identify all but approximately 2^{-47} of our ciphertext pairs as wrong pairs because their differences B'_7 cannot correspond to our desired difference B'_6.

After filtering we are left with approximately 2^{35} ciphertext pairs. Our attack thus requires approximately $2^{36} \times 2^{116}$ partial decryptions, or work equivalent to approximately 2^{150} six-round Serpent encryptions. If we retain only our unfiltered ciphertext pairs, this attack requires approximately 2^{40} bytes of sequential memory. The signal-to-noise ratio of this attack is 2^{83}.

3.2 Improved Six-Round Differential Attack

By counting on fewer than 116 bits of the last round subkey, we can considerably improve the six-round differential attack in the previous section. For example, if we count on two sets of 56 bits, our work is reduced to about 2^{90} Serpent six-round encryptions. This allows us to break six rounds of 128-, 192-, and 256-bit Serpent using less work than exhaustive search.

3.3 Bypassing the First Round

We can use structures to bypass the first round of our five-round characteristic $B'_1 \rightarrow Y'_5$. This gives us an attack that requires fewer chosen plaintexts but more work than the attack in Section 3.2. In this attack we use the four-round, probability 2^{-67}, characteristic $B'_2 \rightarrow Y'_5$. We request 2^{47} blocks of 2^{24} plaintexts such that each block varies over all possible inputs to the active S-boxes in B'_1. This gives us 2^{70} pairs with our desired input difference to the second round. We expect eight pairs with our desired difference Y'_5.

We can mount the attack in Section 3.2 by looking for the last round subkey suggested seven or more times. In this attack we must consider a total of 2^{94} possible ciphertext pairs. As with Section 3.2, we can immediately identify all but 2^{-47} of these pairs as wrong pairs. This attack requires work equivalent to approximately 2^{102} Serpent six-round encryptions and approximately 2^{75} bytes of random-access memory.

3.4 Additional Six-Round Differential Attack

We can modify our basic six-round differential attack by guessing part of the last round subkey and looking at the eight passive S-boxes in B'_5. In order to do this, we must guess 124 bits of the last round subkey.

In this attack we request 2^{40} chosen-plaintext pairs with our input difference B'_1. This gives us 2^9 pairs with difference B'_5 entering the fifth round. For a correct 124-bit last round subkey guess, we expect to count 2^9 pairs with passive S-boxes in Y'_5 corresponding to the passive S-boxes in B'_5. For an incorrect last round subkey guess, the number of occurrences of pairs with passive differences in our eight target S-boxes is approximately normal with mean 2^8 and standard deviation 2^4. Since 2^9 is 16 standard deviations to the right of 2^8, we expect no false positives.

This attack requires 2^{45} bytes of sequential memory and work equivalent to approximately 2^{163} Serpent six-round encryptions.

3.5 Seven-Round Differential Filtering Attack

We can use our filtering scheme in Section 3.1 to distinguish six rounds of Serpent from a random permutation. In this distinguishing attack we request 2^{121} plaintext pairs with our desired input difference B'_1. We expect approximately 2^{41} right pairs. Since our filter passes ciphertext pairs with a probability 2^{-47}, we expect approximately $2^{74} + 2^{41}$ ciphertext pairs to pass our filter.

In a random permutation, the number of unfiltered pairs is approximately a normal distribution with mean 2^{74} and standard deviation 2^{37}. Since $2^{74} + 2^{41}$ is 16 standard deviations to the right of the random distribution's mean of 2^{74}, we can distinguish six-round Serpent from a random permutation. For a random distribution, the probability of observing $2^{74} + 2^{41}$ or more unfiltered pairs is approximately 2^{-190}.

We can extend this six-round distinguishing attack to a seven-round key recovery attack on rounds one through seven by guessing the entire last round subkey K_8 and performing our six-round distinguishing attack. This attack requires approximately 2^{126} bytes of sequential memory 2^{248} Serpent seven-round encryptions.

4 Boomerang Attacks

4.1 Seven-Round Boomerang Distinguisher

In addition to being able to perform traditional differential attacks against Serpent, we can also use Wagner's boomerang attack [Wag99] to distinguish seven rounds of Serpent from a random permutation.

Let us consider a seven-round variant of Serpent corresponding to the second through eighth rounds of the full 32-round Serpent (i.e., rounds $i = 1, \ldots, 7$). Call the first four rounds of this seven-round Serpent E_0 and call the final three rounds E_1. Our seven-round Serpent is thus $E = E_1 \circ E_0$. We can now apply the boomerang technique to this reduced-round Serpent.

Notice that if we only consider the first four rounds of the five-round characteristic in Appendix A.1, we have a four-round characteristic $B'_1 \rightarrow Y'_4$ through E_0 with probability 2^{-31}. Also notice that there exist three-round characteristics through E_1 with relatively high probability. Appendix A.2 illustrates one such characteristic, $B'_5 \rightarrow Y'_7$, with probability 2^{-16}.

To use the terminology in [Wag99], let $\Delta = B'_1$, let $\Delta^* = Y'_4$, let $\nabla = Y'_7$ and let $\nabla^* = B'_5$. We then use $\Delta \rightarrow \Delta^*$ as our differential characteristic for E_0 and $\nabla \rightarrow \nabla^*$ as our differential characteristic for E_1^{-1}.

In the boomerang distinguishing attack, we require approximately $4 \cdot 2^{94}$ adaptive-chosen plaintext/ciphertext queries, or approximately 2^{94} quartets P, P', Q, and Q' and their respective ciphertexts C, C', D, and D'. More specifically, in our distinguishing attack we request the ciphertext C and C' for about 2^{94} plaintexts P and P' where $P \oplus P' = \Delta$. From C and C' we compute the ciphertexts $D = C \oplus \nabla$ and $D' = C' \oplus \nabla$. We then apply the inverse cipher to D and D' to obtain Q and Q'. For any quartet P, P', Q, and Q', we expect the combined properties $P \oplus P' = Q \oplus Q' = \Delta$ and $C \oplus D = C' \oplus D' = \nabla$ to hold with probability 2^{-94}.

4.2 Eight-Round Boomerang Key Recovery Attack

We can extend our seven-round boomerang distinguisher to an eight-round key recovery attack on 192- and 256-bit Serpent reduced to rounds $i = 1, \ldots, 8$ (or rounds $i = 9, \ldots, 16$ or rounds $i = 17, \ldots, 24$). The basic idea is that we peel off the last round by guessing the last round subkey and look for our property in the preceding seven rounds.

A difficulty arises because the boomerang attack makes adaptive chosen plaintext *and* ciphertext queries. Suppose we encrypt P and P' to get C and

C'. To get D and D', we must peel off one round from each ciphertext C and C', XOR the result with ∇, and then re-encrypt the last round with the guessed subkey. To do this, we will have to guess the 68 bits of the last round subkey corresponding to the 17 active S-boxes of B'_8. Assume we consider 2^{94} plaintext pairs P and P'. For each of these pairs, we will have to compute 2^{68} different pairs Q and Q' (for each of the 2^{68} possible last round subkeys). Unfortunately, this means we will likely end up working with the entire codebook of all 2^{128} possible plaintext/ciphertext pairs.

If we are willing to work with the entire codebook of 2^{128} plaintexts and ciphertexts, then we can extract the last round subkey in the following manner. We request the ciphertexts C and C' of 2^{96} plaintext pairs with an input difference Δ. Then for each of our 2^{68} possible last round subkeys and for each of our 2^{96} ciphertext pairs, we compute the boomerang ciphertexts D and D'. We then request the plaintexts Q and Q' corresponding to these ciphertexts. If we correctly guess the last round subkey, we should expect to see the plaintext difference $Q \oplus Q' = \Delta$ with probability 2^{-94}. That is, for the correct subkey we should expect to see the difference $Q \oplus Q' = \Delta$ approximately four times. (Or, put yet another way, if we guess the correct subkey, we should generate about four right quartets.)

This attack requires $2^{68} \times 2^{97}$ partial decryptions and encryptions, or approximately 2^{163} eight-round Serpent encryptions. This attack also requires access to the entire codebook, and thus 2^{128} plaintexts and 2^{133} bytes of random-access memory.

5 Amplified Boomerang Attacks

In [KKS00] we introduced a new class of cryptanalytic attacks which we call "amplified boomerangs." Amplified boomerang attacks are similar to traditional boomerang attacks but require only chosen plaintexts. The chosen-plaintext–only requirement makes the amplified boomerang attacks more practical than the traditional boomerang attacks in many situations. In [KKS00] we describe a seven-round boomerang amplifier distinguishing attack and an eight-round boomerang amplifier key recovery attack requiring 2^{113} chosen plaintext pairs, 2^{119} bytes of random-access memory, and roughly 2^{179} Serpent eight-round encryptions.

5.1 Amplified Seven-Round Distinguisher

In this section we review the seven-round amplified boomerang distinguishing attack presented in [KKS00]. We request 2^{112} plaintext pairs with our input difference Δ. After encrypting with the first half of the cipher E_0, we expect roughly 2^{81} pairs to satisfy the first characteristic $\Delta \rightarrow \Delta^*$. There are approximately 2^{161} ways to form quartets using these 2^{81} pairs. We expect there to be approximately 2^{33} quartets (Y_4^0, Y_4^1) and (Y_4^2, Y_4^3) such that $Y_4^0 \oplus Y_4^2 = \nabla^*$. However, because (Y_4^0, Y_4^1) and (Y_4^2, Y_4^3) are right pairs for the first half of the

cipher, and $Y_4^0 \oplus Y_4^1 = Y_4^2 \oplus Y_4^3 = \Delta^*$, we have that $Y_4^1 \oplus Y_4^3$ must also equal ∇^*. In effect, the randomly occurring difference between Y_4^0 and Y_4^2 has been "amplified" to include Y_4^1 and Y_4^3.

At the input to E_1 we expect approximately 2^{33} quartets with a difference of (∇^*, ∇^*) between the pairs. This gives us approximately two quartets after the seventh round with an output difference of (∇, ∇) across the pairs. We can identify these quartets by intelligently hashing our original ciphertext pairs with our ciphertext pairs XORed with (∇, ∇) and noting those pairs that collide. For a random distribution, the probability of observing a single instance of our cross-pair difference (∇, ∇) is approximately 2^{-33}.

5.2 Amplified Eight-Round Key Recovery Attack

In [KKS00] we extended the previous distinguishing attack to an eight-round key-recovery attack on rounds one through eight of Serpent requiring 2^{113} chosen-plaintext pairs, 2^{119} bytes of random-access memory, and work equivalent to approximately 2^{179} eight round Serpent encryptions. In this attack we guess 68 bits of Serpent's last round key K_9. For each key guess, we peel off the last round and perform the previous distinguishing attack.

5.3 Experimental Improvements to the Eight-Round Attack

We can improve our eight-round boomerang amplifier attack by observing that we do not need to restrict ourselves to using only one specific cross-pair difference (∇^*, ∇^*) after E_0. That is, rather than considering only pairs of pairs with a cross-pair difference of (∇^*, ∇^*) after E_0, we can use pairs of pairs with a cross-pair difference of (x, x) after E_0, for *any* x, provided that both pairs follow the characteristic $x \rightarrow \nabla$ through E_1 with sufficiently high probability.

Experimentally, we find that $\sum_x Pr[x \rightarrow \nabla \text{ through } E_1]^2$ is approximately 2^{-23}.[3] Consequently, if we request 2^{109} chosen-plaintext pairs with our input difference Δ to E_0, we should expect approximately 16 pairs of pairs with a cross-pair difference of (∇, ∇) after E_1. This reduces the work of our attack in Section 5.2 to approximately 2^{175} eight-round Serpent encryptions. As noted in [Wag99], this observation can also be used to improve the standard boomerang attack.

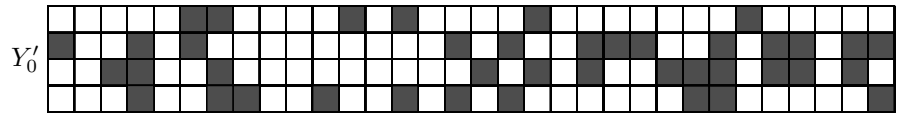
5.4 Amplified Nine-Round Key Recovery Attack

We can further extend the above eight-round attack to break nine rounds of 256-bit Serpent using less work than exhaustive search. To do this, let us consider a nine-round Serpent variant corresponding to rounds zero through eight of

[3] We generated 2^{28} pairs of ciphertext pairs with a cross-pair difference (∇, ∇). We decrypted each pair through E_1^{-1} and counted the number of pairs with a cross pair difference (x, x) for any x. We observed 35 such pairs of pairs.

Serpent. Let us still refer to rounds one through four as E_0 and rounds five through seven as E_1.

If we apply the inverse linear transformation to Δ we get



where Y_0' has 24 active S-boxes. We request 2^{14} blocks of 2^{96} chosen plaintexts such that each block varies over all the possible inputs to the active S-boxes in Y_0'. This gives us 2^{109} pairs with our desired difference Δ into E_0 and 16 pairs of pairs with a cross-pair difference (∇, ∇) after E_1. In order to identify our amplified boomerang, we must guess 96 bits of the first round subkey K_0 and 68 bits of the last round subkey K_9.

We first guess the 96 bits of K_0 corresponding to the 24 active S-boxes in Y_0'. For each 96 bit key guess and for each plaintext P, we encrypt P one round to Y_0. We store P with satellite data Y_0 in HASH0$[K_0]$ and we store Y_0 with satellite data P in HASH1$[K_0]$. This step takes approximately 2^{212} bytes of random-access memory and work equivalent to 2^{203} Serpent eight-round encryptions.

Next, for each 68-bit key guess of K_9, we want to establish a list of all pairs (P^0, P^2) that have difference ∇ as the output of the eighth round. To do this, for each ciphertext C^0, we decrypt up one round to X_8^0, compute $X_8^2 = X_8^0 \oplus B_8'$, and store (X_8^0, X_8^2) or (X_8^2, X_8^0) in a hash table (where the order of X_8^0 and X_8^2 depends on whether X_8^0 is less than X_8^2). The satellite data in our hash table entry includes the plaintext P^0 corresponding to C^0. If a collision occurs in our hash table, we have found two plaintexts P^0 and P^2 that have our desired difference ∇ after the eighth round. We store these pairs (P^0, P^2) in LIST2$[K_9]$ and HASH2$[K_9]$. This step takes approximately 2^{184} bytes of random-access memory and work equivalent to 2^{175} Serpent eight-round encryptions.

The following algorithm counts the number of occurrences of our boomerang amplifier through $E_1 \circ E_0$. This algorithm can be thought of as sending a boomerang from the ciphertext to the plaintext and back again:

for each 96-bit subkey guess of K_0 **do**
  for each 68-bit subkey guess of K_9 **do**
    $count \leftarrow 0$
    for each pair (P^0, P^2) in LIST2$[K_9]$ **do**
      lookup Y_0^0, Y_0^2 corresponding to P^0, P^2 in HASH0$[K_0]$
      $Y_0^1 \leftarrow Y_0^0 \oplus Y_0'$, $Y_0^3 \leftarrow Y_0^2 \oplus Y_0'$
      lookup P^1, P^3 corresponding to Y_0^1, Y_0^3 in HASH1$[K_0]$
      if (P^1, P^3) in HASH2$[K_9]$ **then**
        $count \leftarrow count + 1$
    if $count \geq 15$ **then**
      save key guess for K_0, K_9

For each subkey guess guess of K_9, we expect LIST2$[K_9]$ will contain approximately $2^{219} \times 2^{-128} = 2^{91}$ pairs. Consequently, we expect the inner loop of the

above algorithm to execute 2^{255} times. This attack requires work equivalent to approximately 2^{252} Serpent nine-round encryptions.

6 Meet-in-the-Middle Attacks

Although not as powerful as our previous attacks, we can use the meet-in-the-middle technique to attack six-round Serpent. In the meet-in-the-middle attack, we try to determine the value of a set of intermediate bits in a cipher by guessing key bits from both the plaintext and ciphertext sides. The attack looks for key guesses that match on the predicted values of the intermediate bits.

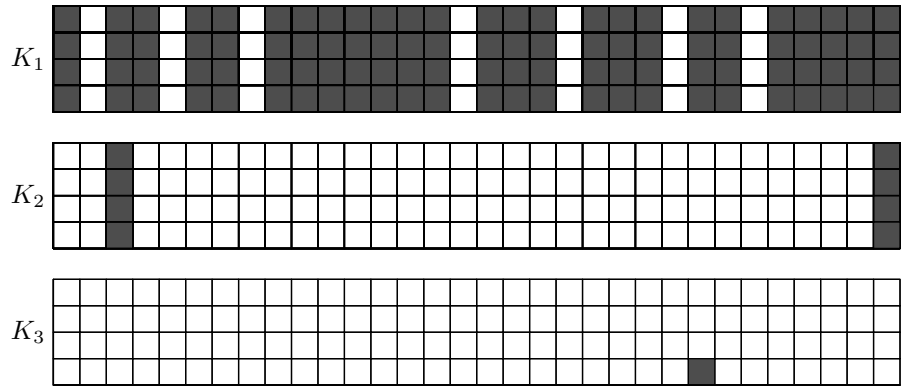
We did a computer search for the best meet-in-the-middle attacks that isolate a set of bits in one column of an intermediate state of Serpent. Table 2 summarizes our results. Although we can also use the meet-in-the-middle technique to predict bits in more than one column of an intermediate state of Serpent, doing so requires additional key guesses and is thus undesirable.

Rounds	b	s	Key guess from top	Key guess from bottom
6	1	B_3	236	239
5	2	B_2	152	223
5	3	B_2	176	224
5	4	B_2	204	225
6	1	X_3	237	238
5	2	X_2	154	221
5	3	X_2	179	221
5	4	X_2	208	221
5	1	Y_2	200	104
5	2	Y_2	200	178
5	3	Y_2	208	198
5	4	Y_2	208	221

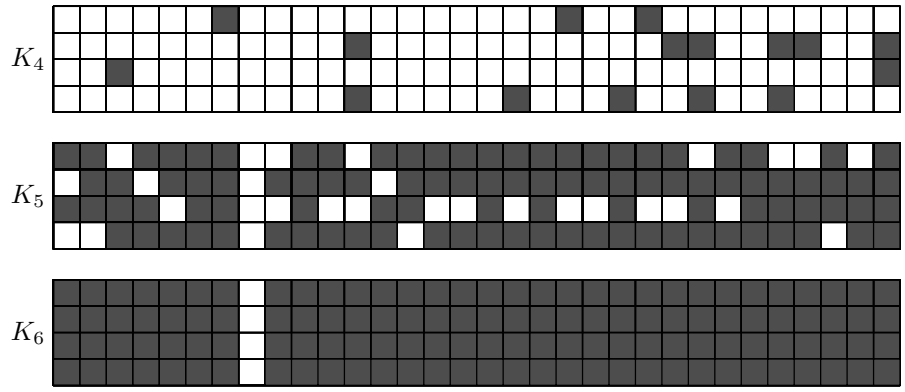
Table 2. Meet-in-the-middle requirements to determine b intermediate bits of internal state s in a given number round Serpent variant.

The clearest way to illustrate the meet-in-the-middle attack on Serpent is through diagrams similar to those used in Section 3 and Appendix A. The plaintext in this attack on six-round Serpent is B_0 and the ciphertext is B_6. The bit we are trying to predict is the eighth most significant bits of x_3 where x_3 is the fourth word of X_3, $X_3 = (x_0, x_1, x_2, x_3)$.

The 237 key bits guessed from the plaintext side are



and the 238 key bits guessed from the ciphertext side are



where the shaded cells denote the bits we guess.

The attack proceeds as follows. We obtain 512 known plaintexts and their corresponding ciphertexts. For each plaintext key guess, we compute the target bit of X_3 for each of our 512 plaintexts. We concatenate these bits for each plaintext into a 512-bit value. We then store this 512-bit value, along with the associated key guess, in a hash table.

For each ciphertext key guess, we proceed along the same lines and compute the target bit of X_3 for each of our 512 ciphertexts. We concatenate these bits for each ciphertext into a 512-bit value and look for this value in our hash table. If we find such a value, then the plaintext and ciphertext keys suggested by the match will likely be correct. This attack requires approximately 2^{246} bytes of random-access memory and work equivalent to 2^{247} six-round encryptions.

7 Key Schedule Observations

This section addresses some observations we have about the Serpent key schedule. We currently do not know of any cryptanalytic attacks that use these observations.

As described in Section 2, the prekeys $w_0, w_1, \ldots, w_{131}$ are computed using the recurrence

$$w_i \leftarrow (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) \lll 11 \tag{1}$$

where $w_{-8}, \ldots, w_{-1}$ is the initial 256 bit master key. If we ignore the rotation and the internal XOR with ϕ and i, we get the linear feedback construction

$$w_i \leftarrow w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \tag{2}$$

Let us now consider two keys K and K^* that have a difference $K' = K \oplus K^*$. The prekeys for K and K^* expand to $w_0, \ldots, w_{131}$ and $w_0^*, \ldots, w_{131}^*$, respectively. By virtue of Equation 2, the prekey differences for K' can be computed using the recurrence

$$w_i' = w_i \oplus w_i^* = w_{i-8}' \oplus w_{i-5}' \oplus w_{i-3}' \oplus w_{i-1}' \tag{3}$$

for $i = 0, \ldots, 131$. If we use the original recurrence (Equation 1) to compute the prekeys rather than Equation 2, the recurrence for w_i' becomes

$$w_i' = (w_{i-8}' \oplus w_{i-5}' \oplus w_{i-3}' \oplus w_{i-1}') \lll 11 \tag{4}$$

for $i = 0, \ldots, 131$.

For any key K, the ith round subkey K_i is computed from the four prekeys $w_{4i}, w_{4i+1}, w_{4i+2}, w_{4i+3}$. The same can be said for the key K^*. If for any given round i the four prekeys for K are equivalent to the corresponding four prekeys for K^*, then the subkeys K_i and K_i^* will be equivalent; this occurs when the prekey differences $w_{4i}', w_{4i+1}', w_{4i+2}', w_{4i+3}'$ are zero.

Let us now observe some situations where the prekey differences for the ith round subkey are zero. As a simple example, let us consider Figure 1. The shaded cells in Figure 1 depict prekeys that are different for K and K^*. The unshaded areas are equivalent between the keys. Notice that six out of the 33 128-bit subkeys are equivalent.

There is a heavy restriction on Figure 1: all the differences must be the same. That is, when Equation 2 is used for the prekey computation, it must be that $w_{-5}' = w_{-3}' = w_{-1}' = \cdots = w_{127}' = k$ for some constant k. If we consider the original prekey recursion (Equation 4), this example works only when $k =$ `0xFFFFFFFF`. Furthermore, when the non-zero prekey differences are `0xFFFFFFFF`, six out of 33 subkeys are equivalent and five out of 33 subkeys have complementary prekeys.

8 Conclusions

In this paper we consider several attacks on Serpent. We show how to use differential, boomerang, and amplified boomerang techniques to recover the key for Serpent up to nine rounds. We also show how to break six rounds of Serpent

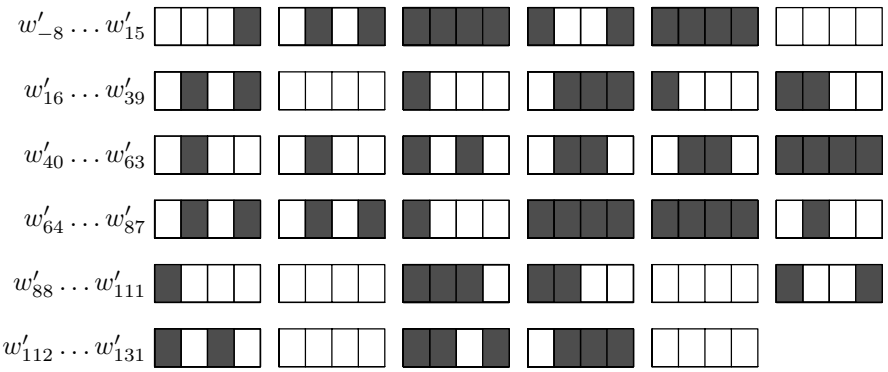


Fig. 1. Difference propagation in the key schedule when $w'_{-5} = w'_{-3} = w'_{-1} =$ `0xFFFFFFFF`.

using a meet-in-the-middle attack. We then provide key schedule observations that may someday be used as the foundation for additional attacks.

Although these attacks do not come close to breaking the full 32-round cipher, we feel that these results are worth reporting for several reasons. Specifically, the results and observations in this paper provide a starting point for additional research on Serpent. These results also provide a security reference point for discussions about modifying the number of rounds in Serpent.

In conjunction with the previous observation, we would like to point out that there are several avenues for further research. Although our current paper addresses differential attacks against Serpent, we have not yet tried linear and differential-linear attacks. We are also attempting to mount additional boomerang variants against Serpent. We expect that all these attacks, while quite capable of breaking reduced-round versions of Serpent, will fail to break the entire 32-round Serpent. In order to break a substantial portion of Serpent's 32 rounds, we suspect that entirely new attacks may need to be invented.

9 Acknowledgements

The "extended Twofish team" met for two week-long cryptanalysis retreats during Fall 1999, once in San Jose and later in San Diego. This paper is a result of those collaborations. Our analysis of Serpent has very much been a team effort, with everybody commenting on all aspects. The authors would like to thank Niels Ferguson, Chris Hall, Mike Stay, David Wagner, and Doug Whiting for useful conversations and comments on these attacks, and for the great time we had together.

References

[ABK98] R. Anderson, E. Biham, and L. Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard," NIST AES Proposal, 1998.

[BS93] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

[Dun99] O. Dunkelman, "An Analysis of Serpent-p and Serpent-p-ns," rump session, *Second AES Candidate Conference*, 1999.

[KKS00] J. Kelsey, T. Kohno, and B. Schneier, "Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent," *Fast Software Encryption, 7th International Workshop*, to appear.

[SKW+99] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Performance Comparison of the AES Submissions," *Second AES Candidate Conference*, 1999.

[Wag99] D. Wagner, "The Boomerang Attack," *Fast Software Encryption, 6th International Workshop*, Springer-Verlag, 1999.

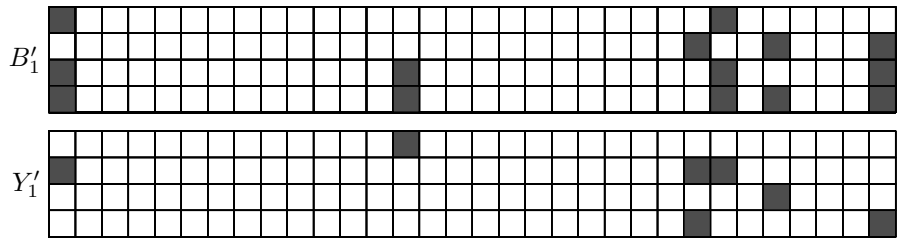
A Differential Characteristics

A.1 Five-Round Characteristic

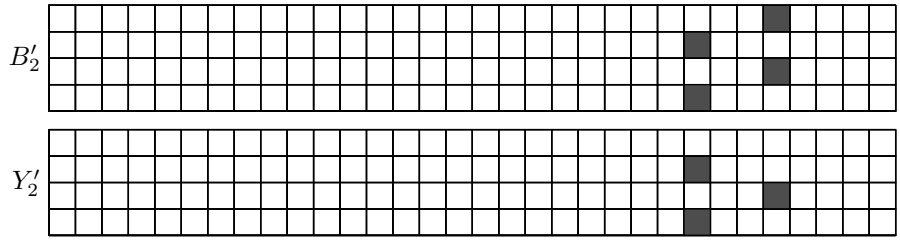
The following is an example of a five-round differential characteristic with probability $p = 2^{-80}$. We used this characteristic in Section 3. This characteristic passes between rounds $i = 1 \bmod 8$ and $i = 5 \bmod 8$. We used only the first four rounds of this five-round characteristic for our boomerang attack in Section 4.

We illustrate this characteristic by showing five one-round characteristics that can be connected with the Serpent linear transformation L. The shaded bits in the figures denote differences in the pairs. We feel that these figures provide an intuitive way to express Serpent's internal states.

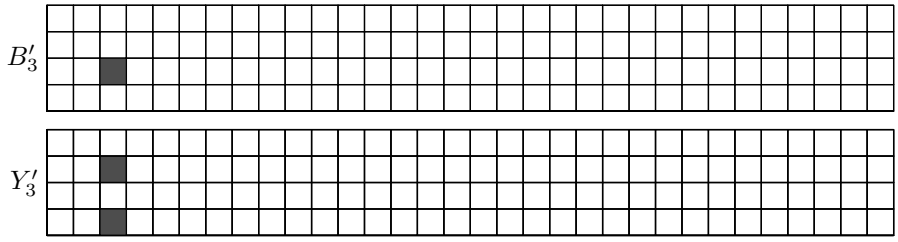
The first-round characteristic, $B'_1 \rightarrow Y'_1$, has probability 2^{-13}:



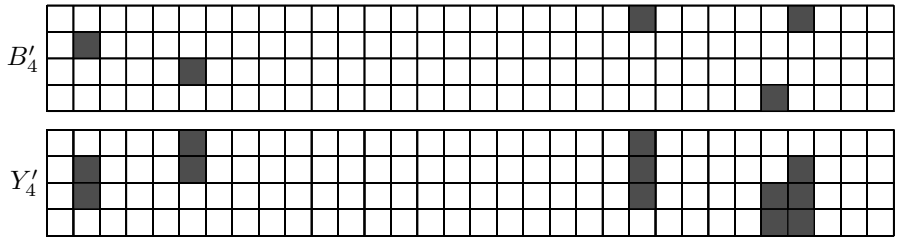
The second-round characteristic has probability 2^{-5}:



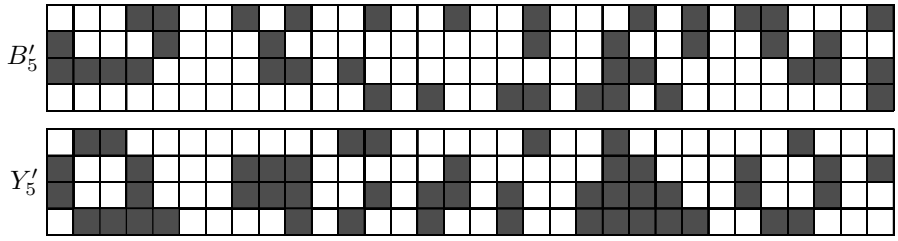
The third-round characteristic has probability 2^{-3}:



The fourth-round characteristic has probability 2^{-10}.



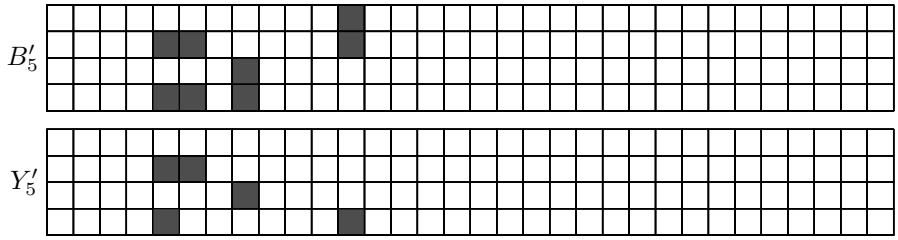
The fifth-round characteristic has probability 2^{-49}.



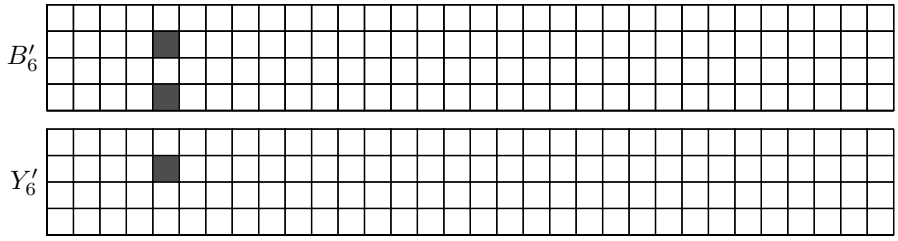
A.2 Boomerang Characteristic

The following is an example of a three-round characteristic with probability $p = 2^{-16}$. We used this characteristic in Section 4. This characteristic passes between rounds $i = 5 \bmod 8$ and $i = 7 \bmod 8$.

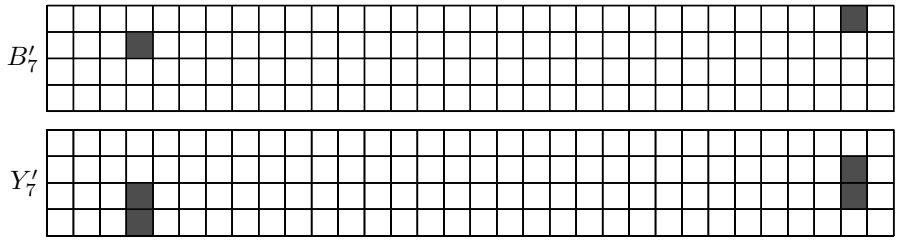
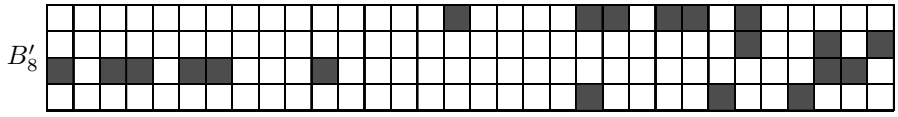
The fifth-round characteristic has probability 2^{-10}:



The sixth-round characteristic has probability 2^{-2}:



The seventh-round characteristic has probability 2^{-4}:



If we apply the linear transformation L to Y'_7, we get: