Filtered nonlinear cryptanalysis of reduced-round Serpent, and the Wrong-Key Randomization Hypothesis.

James McLaughlin*, John A. Clark

Abstract

We present a deterministic algorithm to find nonlinear S-box approximations, and a new nonlinear cryptanalytic technique; the "filtered" nonlinear attack, which achieves the lowest data complexity of any known-plaintext attack on reduced-round Serpent so far. We demonstrate that the Wrong-Key Randomization Hypothesis is not entirely valid for attacks on reduced-round Serpent which rely on linear cryptanalysis or a variant thereof, and survey the effects of this on existing attacks (including existing nonlinear attacks) on 11 and 12-round Serpent.

Keywords: Nonlinear cryptanalysis, generalized linear cryptanalysis, multidimensional linear cryptanalysis, WKRH, Wrong-Key Randomization Hypothesis, Serpent.

1 Introduction.

Linear cryptanalysis [27, 28] has had several extensions and variations proposed since its discovery in 1993. One such generalisation was the use of *non*linear approximations. That is, instead of being restricted to equations of the form $x_{a_1} \oplus x_{a_2} \oplus \ldots \oplus x_{a_i} \oplus y_{b_1} \oplus y_{b_2} \oplus \ldots \oplus y_{b_j}$ in the input bits x_i and output bits y_i of cipher components, the cryptanalyst could make use of higher-degree terms such as $x_{a_1} x_{a_3}$.

This was first proposed by Harpes, Kramer and Massey [21], and investigated in more depth by Knudsen and Robshaw [23]. It was concluded that nonlinear approximations could replace linear approximations only in the first and last rounds of the distinguisher - and even then, there were problems that would not apply in the case of a purely linear approximation. One of these was the difficulty of finding the nonlinear S-box approximations; for a DES-sized 6×4 S-box, the search space for possible approximations was 2^{64} in size, increasing to 2^{256} for an AES-sized 8×8 S-box.

Courtois [14, 15] demonstrated that the use of nonlinear approximations was in fact possible in other rounds of a Feistel cipher, as long as each round's approximation was a particular form of quadratic expression. This approach, however, could not be generalised to non-Feistel ciphers.

The first attempt to obtain arbitrary-degree nonlinear approximations without restrictions on cipher type was the use of simulated annealing (SA) by Clark et al. [8] to evolve nonlinear approximations to the MARS S-box [5] for use in the first round of nonlinear distinguishers. They were able to obtain nonlinear approximations holding with a much higher absolute bias (151/512) than the best linear approximations for the MARS S-box. However, no attack on reduced-round MARS that could exploit these was known.

Subsequent research [9] built on this, refining the SA algorithm and demonstrating the use of nonlinear approximations in attacks on reduced-round Serpent. Matsui's Algorithm 2 was adapted to deal with nonlinear S-box approximations, and new statistical frameworks were proposed for three different approaches to the nonlinear attack.

In this paper, we build on the above research in the following directions:

- We present a fast, deterministic algorithm for obtaining the full set of nonlinear approximations for a given S-box with the highest possible bias.

*Corresponding author, jmclaugh@cs.york.ac.uk

- The cryptanalyst does not know the values of the key bits xored with the bits involved in the nonlinear approximation. Where n_0 denotes the nonlinear function involved, computing n_0 on the bits exposed through partial encryption/decryption means that the cryptanalyst is in fact computing $n_{\alpha_1\alpha_2\ldots\alpha_h} = n_0(x_1 \oplus k_{\alpha_1}, x_2 \oplus k_{\alpha_2}, \ldots, x_h \oplus k_{\alpha_h})$. There exist 2^h candidates for the correct function, n_i, to compute on these bits, and the cryptanalyst does not know which is correct. In previous research [9], three different approaches were given for dealing with this; two based on multidimensional linear cryptanalysis [7] and one which more directly generalised conventional linear cryptanalysis. We present a new approach, *filtered* nonlinear cryptanalysis, which supersedes the first two approaches, and achieves better data complexity than the third at the cost of increased time and memory complexity.

- In [9], we obtained nonlinear approximations for some of the Serpent S-boxes, with higher bias than the best linear approximations for the same, and used them to attack 11-round Serpent. We incorporate these into filtered nonlinear attacks, and compare their performance to the previous attacks on 11-round Serpent.

- For linear cryptanalysis and its variants, the "Wrong-Key Randomization Hypothesis" (WKRH) states that, for any wrong key value used to partially encrypt/decrypt a cipher during cryptanalysis, the expectation for the bias is 0; and it should certainly be much lower than the bias for the correct key. We demonstrate that in the case of the Serpent cipher, this does not always apply, and quantify its effects on the various attacks on 11- and 12-round Serpent, including our nonlinear and filtered nonlinear attacks.

This paper is structured as follows: The remainder of this section describes the notation used, and provides a brief description of certain key aspects of linear cryptanalysis. Section 2 describes the new search algorithm for S-box approximations, and discusses the ways in which the new approximations affect the attack. It also contains an explanation of how we handle nonlinear approximations differently in the filtered attacks. Section 3 describes the new attacks, in particular the adaptation of Collard et al.'s improved algorithm for the analysis phase [13] to the nonlinear and filtered nonlinear domains. It also contains a detailed discussion of the complexities of this algorithm and the nonlinear attacks. Finally, Section 4 surveys the existing attacks on reduced-round Serpent, recalculates their complexities in light of the issues surrounding the WKRH, and describes the nonlinear and filtered nonlinear attacks on 11-round Serpent.

1.1 Linear cryptanalysis - the Algorithm 2 attack.

We use the following notation:

- N is the number of known plaintext/ciphertext pairs.
- K denotes the cipher's key length.
- l is the number of "active" text bits which are relevant to the attack. In a 1R attack, this includes plaintext bits which are xored together but not partially encrypted.
- The subset of key bits we seek to recover is known as the *target partial subkey* (TPS).
- k is the number of key bits in the TPS. For 2R attacks on SPN-based ciphers such as Serpent, $k = l$. For 1R attacks on SPNs, k is equal to the number of active ciphertext bits.
- k_0 is the correct k-bit value for the TPS.
- In nonlinear attacks, k_1 denotes the subset of TPS bits that are used in the round keys for the outer rounds of the cipher. (All attacked key bits are of this type in a linear attack.)
- In nonlinear attacks, k_2 is the set of TPS bits active in the outer rounds of the *approximation*.

- r is the number of rounds of the cipher.
- P_s is the success probability of the attack.
- If, in a cryptanalytic attack, we aim for the correct key to be one of the 2^{n-a} highest ranked keys, the value a is referred to as the "advantage".

In a 1R attack, the cryptanalyst knows of a linear approximation to rounds $1, 2, \ldots, (r-1)$ of the cipher, and uses candidate key values to partially decipher some of the bits in the known ciphertexts. In a 2R attack, the cryptanalyst only has an approximation to rounds $2, \ldots, (r-1)$, and as well as the aforementioned partial decryption, must partially encrypt certain plaintext bits to obtain the bits on which the probabilistic linear relation is expected to hold.

The theoretical bias for this linear approximation is calculated using the *Piling-Up Lemma* [27]:

Definition 1.1. For $1 \leq i \leq n$, let X_i be independent Bernoulli random variables such that $p_i = P(X_i = 0)$, and $(1 - p_i) = P(X_i = 1)$.

(In the case of linear cryptanalysis, $X_i = 0$ iff the linear approximation to the ith approximated S-box holds.)

Then $P(X_1 \oplus X_2 \oplus \ldots X_n = 0)$ is:

$$(1/2) + 2^{n-1} \prod_{i=1}^{n} (p_i - 1/2).$$

with probability bias:

$$\epsilon = 2^{n-1} \prod_{i=1}^{n} (p_i - 1/2)$$

In reality, the probabilities of the linear approximations to the S-boxes in one round holding are not independent of the probabilities of the linear approximations to other rounds holding, so the Piling-Up Lemma only estimates the true bias. This is usually accurate enough for the purposes of cryptanalysis, although there are situations where it is not [29, 26, 10].

Definition 1.2. Where a linear approximation holds with bias ϵ, i.e. with probability $1/2 + \epsilon$, the *capacity* C of the approximation is equal to $4 \times \epsilon^2$. More generally, in an attack using multiple approximations A_i $(1 \leq i \leq M)$, each with bias ϵ_i, the set of approximations has capacity $4 \sum_{i=1}^{M} \epsilon_i^2$.

2 Finding and utilising nonlinear approximations.

2.1 Finding the approximations.

The approximations used are of the following form: (linear function of either the input or the output bits) = (nonlinear function of some subset of the other) with bias ϵ. The linear function is defined by some bitmask with 1s in the positions corresponding to the bits involved.

We use the term "projection" to refer to the subset of either the input bits x_i or output bits y_i involved in the nonlinear function. For example, $y_0 \oplus y_1 \oplus y_0 y_3$ has the projection $\{y_0, y_1, y_3\}$.

Let us use Serpent S3 to illustrate the new algorithm. We will search for approximations involving a nonlinear function on the set of input bits, with projection $\{x_0, x_1, x_3\}$ and bitmask 10 (1010).

First of all, we reorder the truth table of the linear function.

Value xyz of bits in projection	000	001	010	011	100	101	110	111
TT entry for $xy0z$	0	0	1	1	1	0	0	1
TT entry for $xy1z$	0	1	1	1	1	0	0	0

We define a template for the approximations as follows: For any value xyz of the bits in the projection, if the truth table of the linear function takes the value 0 more often than the value 1, let template entry xyz equal 0. If the opposite is true, set entry xyz to 1. If the two entries occur equally often, let entry xyz be the character *.

This gives us:

Value xyz of bits in projection	000	001	010	011	100	101	110	111
TT entry for $xy0z$	0	0	1	1	1	0	0	1
TT entry for $xy1z$	0	1	1	1	1	0	0	0
Template entry for xyz	0	*	1	1	1	0	0	*

We can now obtain four approximations with bias 6 by replacing the *s in the template with 0s and 1s. These are:

- 00111000 ($x_1 \oplus x_0 \oplus x_0x_3 \oplus x_0x_1x_3$),
- 00111001 ($x_1 \oplus x_0 \oplus x_0x_3$),
- 01111000 ($x_3 \oplus x_1 \oplus x_0 \oplus x_1x_3$), and
- 01111001 ($x_3 \oplus x_1 \oplus x_0 \oplus x_1x_3 \oplus x_0x_1x_3$).

2.2 The "related" approximations.

We have already mentioned the difficulty faced by the cryptanalyst in working out which of $2^{|k_2|}$ functions is the correct nonlinear function. One possible approach would be to compute all of the functions, and for each guess at the key bits involved, accept the function with the highest bias as correct.

If we wish to include the k_2 bits in our attack, several of the related approximations may also possess biases with high magnitude. In some cases, one or more of the relateds may have a bias with the same magnitude as the original, and even when this is not the case, we may still need to distinguish, say, the correct function and a bias 24 approximation from an incorrect function defining a bias -22 approximation.

Let x_i denote the ith input bit to whichever S-box we are dealing with, and y_j the jth output bit. Consider the nonlinear approximation to Serpent S3 in Table 1:

Related approximation	Nonlinear function	Bias	Bias (filtered)
0	$x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_1y_3$	+6	6
1	$x_3 \oplus x_4 = y_4 \oplus y_3 \oplus (y_1 \oplus 1)y_3$	0	-4
2	$x_3 \oplus x_4 = y_4 \oplus (y_3 \oplus 1) \oplus y_1(y_3 \oplus 1)$	0	0
3	$x_3 \oplus x_4 = y_4 \oplus (y_3 \oplus 1) \oplus (y_1 \oplus 1)(y_3 \oplus 1)$	+2	0
4	$x_3 \oplus x_4 = (y_4 \oplus 1) \oplus y_3 \oplus y_1y_3$	-6	0
5	$x_3 \oplus x_4 = (y_4 \oplus 1) \oplus y_3 \oplus (y_1 \oplus 1)y_3$	0	0
6	$x_3 \oplus x_4 = (y_4 \oplus 1) \oplus (y_3 \oplus 1) \oplus y_1(y_3 \oplus 1)$	0	2
7	$x_3 \oplus x_4 = (y_4 \oplus 1) \oplus (y_3 \oplus 1) \oplus (y_1 \oplus 1)(y_3 \oplus 1)$	-2	-4

Table 1: Nonlinear approximation to Serpent S3.

(The "filtered" biases are explained in the next subsection.)

For this nonlinear approximation, if $y_1y_3 = 1$, any wrong guess for corresponding key bits (k_a, k_c) will result in its value being wrongly calculated as 0. If $y_1y_3 = 0$, by contrast, only one of the three possible wrong guesses for (k_a, k_c) will result in the wrong value being calculated. In general, an

incorrect key guess will not consistently result in the wrong value being assigned to the nonlinear terms affected by it, and so will not simply leave the magnitude of the bias invariant.

It is therefore necessary to guess the key bits involved in the first and last rounds of the approximation, as well as those involved in the first and last rounds of the cipher, to obtain the latter set of key bits.

2.3 Increasing the signal/noise ratio - "filtering" nonlinear cryptanalysis.

Each of the approximations in Section 2.1 has bias $6/8 = 0.75$. However, for two of the possible inputs to the function, when $PARTIAL\_ENCRYPT(P \oplus k_1) \oplus k_2 = 001$ or 111, the approximation has no bias. In the analysis phase, no information is obtained by adding to counter values when these inputs occur, and we therefore have no reason to do so.

In fact, we have very good reason not to do so. Consider that, for each (k_1, k_2) pair, by ignoring (P, C)-pairs such that function inputs 001 or 111 would occur, we effectively increase the bias from 6/8 to 6/6. Since the data complexity is proportional to the square of the bias, we appear to reduce the KP requirements to $(3/4)^2 = 9/16$ of their original value. In actual fact, since the improved bias is obtained by discarding a quarter of the available data, the value of N is only reduced to 3/4 of its original value.

This improvement sometimes comes at a price - in a basic nonlinear attack using one of the four original approximations, we could for two of these approximations ignore certain values of k_2 which simply resulted in the truth table of the nonlinear approximations being flipped. Since we now need the full value of $PARTIAL\_ENCRYPT(P \oplus k_1) \oplus k_2$ to know whether to filter it out, and since different k_2 result in different sets of values being filtered out, we cannot now easily omit these k_2 from the attack. As this previously allowed us to compute the nonlinear function for only half the values of k_2, the time complexity of the attack is doubled.

For example, let us consider the second of the four approximations:

Related approximation	Truth table	Filtered truth table	Bias	Bias of "filtered" approximation
000	00111001	0*11100*	6	6
001	00110110	*01101*0	0	0
010	11000110	110*0*10	-6	-4
011	11001001	11*0*001	0	0
100	10010011	100*0*11	-2	-4
101	01100011	01*0*011	0	0
110	01101100	0*10110*	2	2
111	10011100	*00111*0	0	0

In the basic nonlinear attack, we do not need to compute truth table values for half of the relateds, since the related for $k_2 \oplus 010$ will have the same absolute bias (but opposite sign) to that for k_2. In an attack using filtering, this is clearly no longer the case.

2.4 How unbalanced nonlinear components in the approximation affect the attack.

Let us assume that one end of the overall approximation is balanced. Without loss of generality, we may assume that this is the input end. Let P(function at output end = 0) be denoted α.

Then, for an incorrect key, P(approximation = 0) =

$$P((x_{a_1} \oplus \ldots \oplus x_{a_s} = 0) \cap (y_{a_1} \oplus \ldots \oplus y_{a_s} = 0)) + P((x_{a_1} \oplus \ldots \oplus x_{a_s} = 1) \cap (y_{a_1} \oplus \ldots \oplus y_{a_s} = 1))$$
$$= (0.5 \times \alpha) + (0.5 \times (1 - \alpha))$$
$$= 0.5$$

We see that, as long as either the first or the last round of the overall approximation is a balanced function, it does not matter whether the function at the other end is balanced.

Unfortunately, in general we cannot use unbalanced approximations at both ends. Let β denote the probability that the nonlinear function at the input end equates to zero, and let γ be the corresponding probability for the function at the output end. Then, for an incorrect key, P(approximation = 0) =

$$P((x_{a_1} \oplus \ldots \oplus x_{a_s} = 0) \cap (y_{a_1} \oplus \ldots \oplus y_{a_s} = 0)) + P((x_{a_1} \oplus \ldots \oplus x_{a_s} = 1) \cap (y_{a_1} \oplus \ldots \oplus y_{a_s} = 1))$$
$$= (\beta \times \gamma) + ((1 - \beta) \times (1 - \gamma))$$

which is not always equal to 0.5.

3 The new cryptanalysis algorithm.

3.1 Adapting the new analysis phase to nonlinear cryptanalysis of SPNs.

Where the cipher being attacked is an SPN, we present an adaptation of Collard et al.'s new algorithm for the analysis phase [13, 30] to nonlinear (and filtered nonlinear) cryptanalysis. An adaptation of this method to nonlinear cryptanalysis of Feistel ciphers was previously described in [9].

- Let $f(i, j)$, where i is the value of the active text bits, and j the value of the k_1 bits with which they are xored, be a $2^{|k_2|}$-long string of values. We compute it as follows:

  1. Partially encrypt/decrypt i using j. This yields a string δ of text bits entering/leaving the outer rounds of the approximation, $|k_2|$ of which are involved in the nonlinear component.
  2. For each possible value μ of k_2, compute the nonlinear function on $(\delta \oplus \mu)$.
  3. If the attack does not use filtering, set the μth entry in the string of values to -1 if the nonlinear approximation does not hold, 1 if it does hold.
  4. For a filtered attack, set the μth entry to 0 if $(\delta \oplus \mu)$ is one of the inputs being "filtered out". Otherwise, assign either -1 or 1 as a value in the same way as before.

- Since δ is obtained by applying a sequence of functions to a set of bits determined entirely by the value of $(i \oplus j)$, the matrix C such that $C_{ij} = f(i \oplus j)$ = the value $\in \{-1, 1\}$ or $\{-1, 0, 1\}$ which we have just computed can be defined as it was in [13], except that C_{ij} is now a string of values instead of just one. We only need to compute one column of C.

- Where x is the vector containing the frequency with which each value for the l active text bits occurred, since C is a circulant matrix, Cx can be calculated using the Fast Fourier Transform. Each entry in Cx is a $2^{|k_2|}$-string of integers.

- The memory complexity, and time complexities in terms of AOs and MAs, of the corresponding stages of the linear version of this method can be multiplied by $2^{|k_2|}$ to obtain the complexity of the new method up to this point. Since we do not need this many copies of the "interim" arrays y and z, the memory complexity is in fact slightly lower.

  However, if we do not employ key ranking, we can optimise much further in terms of memory. Instead of calculating and storing the entire $2^{|k_1|} \times 2^{|k_2|}$ matrix Cx (the final column of which can use the space originally occupied by x), we could compute one column at a time and search it for its maximal absolute entry. In an array indexed by k_2 value, we store this entry and its corresponding value of k_1. The highest value will correspond to the most likely (k_1, k_2) and we therefore need only enough memory for two columns of Cx (half of which will in fact be used to store x). For a 2R attack, this reduces memory requirements to $2^{|k_1|+|k_2|} + 2^{|k_1|+4+1} + 2^{max(l_1,l_2)+5} = 2^{|k_1|+5} + 2^{|k_1|+|k_2|} + 2^{max(l_1,l_2)+5}$ bytes (The $2^{max(l_1,l_2)+5}$ bytes are explained in the discussion of the original method's complexity below) instead of $2^{|k_1|+|k_2|+4.087} + 2^{max(l_1,l_2)+5}$.

The time complexity of each partial encryption/decryption may be higher than in the case of linear cryptanalysis, due to the complexity of computing the nonlinear function.

- We assign to each Cx_i a score equal to the maximum absolute value therein. The highest-scoring Cx_i corresponds to the most likely key. This requires $(2^{|k_1|} + 2^{|k_1|+|k_2|})$ MAs, to access all values in all strings and to write the scores to an array.

  The array of scores should need at most (block size of cipher) bits per entry. For block size 128, this adds $16 \times 2^{|k_1|}$ bytes to the memory complexity.

- This allows us to deduce k_1. We can then proceed to obtain information on k_2 by analysing the biases of the relateds for the correct k_1 candidate.

We can use the maximum absolute bias of all the related approximations to calculate the data complexity in the same way that the bias of one approximation is used in linear cryptanalysis.

3.2 The complexity of the method.

We discuss the complexity of this analysis method for linear attacks in more detail.

The column of C has 2^k entries, all -1 or 1. We need 2^k bytes to store it in signed char variables. Variable types using fewer bits are unlikely to be efficiently implemented on any platform.

The vector x has 2^k entries, each of which must be at least $log_2(N)$ bits in size. On a 64-bit processor, a cipher with 128-bit block size will require 2^{k+1} words here, or 2^{k+4} bytes.

During the calculation of Cx, two "interim" arrays, y and z, are used [13]. Based on Carlet's description [6] of a version of the FFT which is equivalent to both the Fast Walsh-Hadamard Transform and the k-dimensional FFT of size 2^k [24], we note that the same data type can be used for these as for x, and hence these arrays will require 2^{k+5} bytes.

This gives us a memory complexity of $2^k + 2^{k+4} + 2^{k+5} \approx 2^{k+5.615}$ bytes. The space used by one of the previous arrays, such as x, can be reused to store Cx.

We now consider the time complexity. The algorithm requires 2^k partial encryption/decryptions (PEDs) to calculate a column of C, followed by $O(3{\cdot}k{\cdot}2^k)$ memory accesses (MAs) and arithmetic operations (AOs) to calculate Cx.

Based on the aforementioned version of the FFT [6], we estimate $\approx (2k+3) \cdot 2^k$ MAs per transform. Where y and z denote the output arrays from the first two transforms, calculating the dot product $y \cdot z$ requires 3×2^k MAs. Multiplying the per-transform complexity by three, and adding the complexity of the dot product and the 2^k MAs when the first column of C was calculated and written to memory, gives us $\approx (6k+13) \cdot 2^k$ MAs in total. As for AOs, the calculation of the dot product requires 2^k AOs, and we estimate $\approx (2k+1) \cdot 2^k$ AOs per transform, giving us a total of $\approx (6k+4) \cdot 2^k$.

This is a significant improvement over the $O(2^{2k})$ memory accesses of the original analysis phase; although in most cases that phase was able to access contiguously stored array elements in sequence (work with $COUNTERS\_2[i]$ and $COUNTERS\_1[j+1]$ would occur immediately after work with $COUNTERS\_2[i]$ and $COUNTERS\_1[j]$ (stored at the address prior to $COUNTERS\_1[j+1]$)) and it may be that the extent of the improvement is reduced if this factor aided the CPU's cache management/location-seeking in main memory.

Equating complexity in terms of memory accesses to complexity in terms of partial cipher encryptions is a difficult matter [19], depending on several factors such as; whether the CPU's memory controller is on-die or off-die, whether the memory access is to L1 cache, L2 cache, higher-level cache or main memory, the instruction set of the CPU, the efficiency of physical address extension... Previous work on the cryptanalysis of reduced-round Serpent [2, 4, 3] was not always consistent in converting between the two, and assumed 3 processor cycles per memory access - which would seem to require all memory accesses to be to L1 processor cache. Estimates for the time required to access data in main memory in the event of a cache miss vary from 75 to 300 cycles, and it is not clear if this figure is likely to increase or decrease over time, as processor performance improvements increasingly rely on

multiple cores and parallel execution rather than increased clock speed. In 2003, the NESSIE project [32] gave a figure of 50 cycles per encrypted byte on either the PowerPC G3 or G4 processor as the best performance for full Serpent; if we extrapolate from this to 800 cycles per block we have a worst-case estimate of 1 MA = 3/8 of a full Serpent encryption, and we do not have up-to-date figures for more recent processors to compare this to. It is becoming accepted that there is no easy means to compare complexity in terms of memory accesses to complexity in terms of cipher operations [19], and this is a problem we ourselves will encounter when discussing the performance of our attacks in a later section.

For 2R attacks, later research [30] allows us to trade very slight increases in MA and AO complexity for reduced memory and PED complexities. Let l_1, l_2 be such that $(l_1 + l_2) = k$, where l_1 denotes the number of TPS bits acting on the plaintext, and l_2 the number of TPS bits acting on the ciphertext. Then instead of 2^k PEDs, the method need only execute 2^{l_1} partial encryptions (PEs) and 2^{l_2} partial decryptions (PDs), in addition to est. $(2^{l_2} \cdot (6l_1 + 4) \cdot 2^{l_1} + 2^{l_1} \cdot (6l_2 + 4) \cdot 2^{l_2}) = (6k + 8) \cdot 2^k$ AOs and est. $(2^{l_2} \cdot (6l_1 + 13) \cdot 2^{l_1} + 2^{l_1} \cdot (6l_2 + 13) \cdot 2^{l_2}) = (6k + 26) \cdot 2^k$ MAs. Memory complexity is also improved, since the arrays y and z need only have $2^{max(l_1,l_2)}$ entries each, reducing the total to $2^k + 2^{k+4} + 2^{max(l_1,l_2)+5} \approx 2^{k+4.087} + 2^{max(l_1,l_2)+5}$ bytes.

This algorithm was also generalised for multidimensional linear attacks [30]. Where m is the number of dimensions, the generalised algorithm requires $2^m\times$ the number of MAs and AOs for the one-dimensional case, plus the complexity of computing $2^{l_1+l_2}$ more transforms on a data set of size 2^m to convert correlations to empirical probability distributions, plus the complexity of applying the convolution method [22] to these distributions.

3.3 Other issues affecting the complexity of the new attack.

The time complexity is affected by the cost of computing a nonlinear function compared to the cost of a linear function (usually considered negligible), and by the differing numbers of active S-boxes. For example, this is the nonlinear component of an approximation to DES S5:

$$1 \oplus x_5 \oplus x_5x_6 \oplus x_2x_6 \oplus x_1x_5 \oplus x_1x_2 \oplus x_1x_5x_6 \oplus x_1x_2x_6$$

It is not clear how to compare the complexity of this to the complexity of the full S-box, as it is unlikely that an S-box implementation would rely solely on XOR, AND and NOT (to add the constant term) gates. Moreover, the difficulty of finding, for a given basis and function, the circuit for that function with the smallest number of gates is a difficult and still open problem [16]. It is to be assumed that the cryptanalyst would be using S-box implementations chosen to maximise speed, without regard to such factors as resistance to side-channel attacks which most cipher implementations would have to address.

Since this may be represented by a lookup table with as many elements as the S-box:

$$1101110111011101100010001000100011111111111111110000000000000000,$$

and since its algebraic normal form has a much smaller weight than any co-ordinate function of the S-box, we will assume that the complexity of calculating this function is $\leq$ that of computing the full S-box. Since it must be calculated $2^{|k_2|}$ times for each PED, where S_c denotes the total number of S-boxes in all the rounds of the cipher, we estimate the time required for each PED to be $\leq$ (number of active outer round boxes)$/S_c + 2^{|k_2|}/S_c$ of the time required for a full encryption.

In a filtered attack, prior to computing the nonlinear function we must check whether $(\delta \oplus \mu)$ is filtered. This requires either another lookup table or the computation of a second function, and so we upper-bound the PED complexity with (no. of active outer round boxes)$/S_c + 2^{|k_2|+1}/S_c$.

4 Cryptanalysing reduced-round Serpent.

4.1 Survey of existing attacks.

The various linear, differential-linear and multidimensional linear attacks on reduced-round Serpent fall into two categories; those based on Collard et al.'s approximations [11, 12, 13, 30] and those based on the approximation of Dunkelman, Keller et al. [2, 4, 20]. In Appendix A, we point out a few errors in the existing descriptions of Dunkelman et al.'s approximation.

However, the data complexities of some of these attacks have been underestimated.

Let C denote capacity, and p the probability that the linear approximation holds (so $(p-1/2)$ is the bias). Let P_s denote the success probability of the attack. In [30], N is equal to $4C^{-1}$. This figure is intended to match the values for N used by Collard et al. in multiple linear attacks. However, Collard et al. also used $N = 4 \cdot |p - 1/2|^{-2}$ in conventional linear attacks (apparently to obtain $P_s = 0.785$ as predicted by Matsui in Table 3 of [27]), and this is $16C^{-1}$, not $4C^{-1}$. Moreover, Table 3 of [27] assumes that $l = 6$ - which is not the case in any of the attacks on Serpent - and the values therein are calculated using a double integral which does not match that obtained in Selçuk's later research [33].

The below equation is Selçuk's [33] double integral. It allows the success probabilities for various x such that $N = x \cdot |p - 1/2|^{-2}$ to be calculated for arbitrary l, assuming that the Wrong-Key Randomization Hypothesis holds:

$$P_s = \int_{-2\sqrt{N}|p-1/2|}^{\infty} \left(\int_{-u-2\sqrt{N}|p-1/2|}^{u+2\sqrt{N}|p-1/2|} \phi(v)\mathrm{d}v \right)^{2^l-1} \phi(u)\mathrm{d}u \tag{4.1}$$

However, in the case of Collard et al.'s approximation, the WKRH does not always apply, and so we cannot use Equation 4.1 directly. If we look at Figure 1, we see that an input difference of 0010 or 1000 to Serpent S2 will always cause the value of y_4 to flip - and input difference 1010 will always leave it invariant. Likewise, input differences 0010, 0100 and 0110 will cause the value of $y_0 \oplus y_1 \oplus y_2$ to flip with probability bias $\pm 1/2$.

Absolute input and XOR-truncated output difference distribution table for Serpent S2.

Table entries should be divided by 16.

XORed BITS OF OUTPUT DIFFERENCE

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
I 0	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-128
N 1	-8	0	+4	0	+4	0	0	0	+4	+4	-4	0	0	-4	0	0	0
P 2	-8	+8	0	0	0	0	0	0	0	0	0	0	0	0	+8	-8	0
U 3	-8	0	0	+4	0	+4	-4	+4	0	0	0	+4	-4	0	0	0	0
T 4	-8	0	+4	+4	0	0	+4	-4	+4	-4	0	0	+4	+4	-8	0	0
5	-8	0	-4	0	0	+4	0	0	-4	+4	0	+4	0	+4	0	0	0
D 6	-8	0	0	0	0	0	0	0	0	0	0	0	0	0	+8	0	0
I 7	-8	0	0	-4	+4	0	+4	+4	0	0	-4	0	+4	0	0	0	0
F 8	-8	+8	0	0	+4	-4	0	0	0	0	+4	-4	0	0	0	0	0
F 9	-8	0	0	-4	-4	0	0	0	+4	+4	+4	0	+4	0	0	0	0
E 10	-8	-8	+4	+4	0	0	0	0	0	0	0	0	+4	+4	0	0	0
R 11	-8	0	-4	0	0	+4	-4	+4	0	0	0	+4	0	+4	0	0	0
E 12	-8	0	0	0	0	0	+4	-4	+4	-4	0	0	0	0	0	+8	0
N 13	-8	0	0	+4	0	+4	0	0	-4	+4	0	+4	-4	0	0	0	0
C 14	-8	0	0	0	+4	-4	0	0	0	0	+4	-4	0	0	0	+8	0
E 15	-8	0	+4	0	-4	0	+4	+4	0	0	+4	0	0	-4	0	0	0
Total	-128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 1: Table showing probability biases for truncated differentials for Serpent S2, in which input difference ΔX causes linear combination λY of the output bits to flip with bias ε.

Figure 2 shows that four of the input-end S-boxes for which we guess key bits in the Collard/Standaert/Quisquater attack are affected by this; two with output bitmask 1 and two with bitmask 14. This means that we can only recover eight of the sixteen key bits for these S-boxes.

Likewise, from Figure 4, we see that an input difference to S4's inverse (i.e. an output difference to S4) of 0010, 1100 or 1110 causes the value of $x_0 \oplus x_2 \oplus x_3$ to flip with bias 0.5. Since one of the active ciphertext S-boxes contributes the parity of these bits to the approximation, the number of bits that can be recovered is reduced by 2 again. Instead of recovering 108 key bits, we can only recover 98.

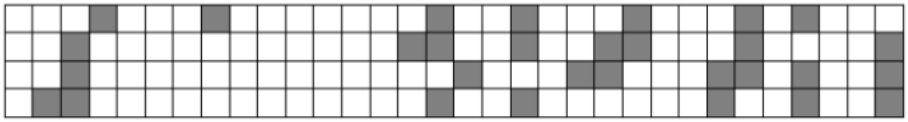

Figure 2: Input end S-boxes in Collard et al.'s attack. Dark cells signify active output bits.

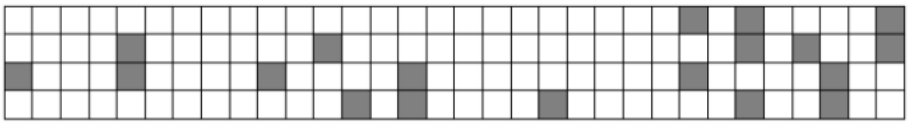

Figure 3: Output end S-boxes in Collard et al.'s attack.

Absolute input and XOR-truncated output difference distribution table for Serpent S4 inverse.

Table entries should be divided by 16.

XORed BITS OF OUTPUT DIFFERENCE

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
I	0	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-128
N	1	-8	+4	0	0	0	+4	+4	-4	+4	0	0	0	0	-4	0	0	0
P	2	-8	+4	0	+4	0	0	0	0	+4	0	+4	-8	0	0	0	0	0
U	3	-8	-4	0	0	+4	0	0	0	-4	0	0	0	+4	0	+4	+4	0
T	4	-8	+4	+4	0	+4	0	-4	0	0	-4	+4	0	0	+4	0	-4	0
	5	-8	-4	0	0	0	0	+4	0	+4	0	0	0	+4	+4	-4	0	0
D	6	-8	0	+4	+4	0	-4	0	+4	+4	-4	0	0	-4	0	+4	0	0
I	7	-8	+4	0	0	-4	+4	0	+4	-4	0	0	0	0	0	0	+4	0
F	8	-8	0	+4	0	+4	0	0	0	0	+4	0	0	0	0	0	-4	0
F	9	-8	0	0	+4	0	0	0	-4	0	0	+4	0	0	0	+4	0	0
E	10	-8	0	+4	0	0	0	-4	0	0	+4	0	0	0	+4	0	0	0
R	11	-8	0	0	-4	-4	+4	+4	0	0	0	-4	0	+4	+4	0	+4	0
E	12	-8	0	-4	0	0	-4	0	0	0	+4	0	+8	0	0	+4	0	0
N	13	-8	0	0	+4	0	+4	0	0	0	0	-4	0	+4	0	0	0	0
C	14	-8	0	-4	0	0	0	0	+4	0	+4	0	+8	-4	0	0	0	0
E	15	-8	0	0	-4	+4	0	+4	+4	0	0	+4	0	0	-4	-4	+4	0
Total		-128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 4: Table showing probability biases for truncated differentials for Serpent S4's inverse, in which input difference ΔX causes linear combination λY of the output bits to flip with bias ε.

This leaves 22 active S-boxes. For each of these, there are six incorrect keys such that the approximation is expected to hold with an absolute bias equal to half the absolute bias of the correct key. Let ε denote the value $|p - 1/2|$. We compared the results of computing P_s using Equation 4.1 with:

$$\int_{-2\sqrt{N}\varepsilon}^{\infty} \left(\prod_{i=1}^{22} \left(\int_{-u-(2\sqrt{N}\varepsilon(1+1/2^i))}^{u+2\sqrt{N}\varepsilon(1-1/2^i)} \phi(v)\mathrm{d}v \right)^{6^i\binom{22}{i}} \right) \left(\int_{-u-2\sqrt{N}\varepsilon}^{u+2\sqrt{N}\varepsilon} \phi(v)\mathrm{d}v \right)^{2^l - \sum_{i=1}^{22}\left(6^i\binom{22}{i}\right)} \phi(u)\mathrm{d}u \quad (4.2)$$

for $l = 98$. However, the difference was negligible. We deduced that $N = 37.63|p-1/2|^{-2} \approx 2^{121.234}$ was necessary to achieve $P_s = 0.785$.

In Biham et al.'s linear attack [2], five active plaintext boxes have incorrect key values which cause the parity of their active output bits to flip with bias 0.5. This reduces the number of key bits which can be recovered from 140 to 130. The other active plaintext/ciphertext S-boxes all have six input/output differences which flip the parities of their active bits with bias 0.25, but these have a negligible effect on the value of P_s.

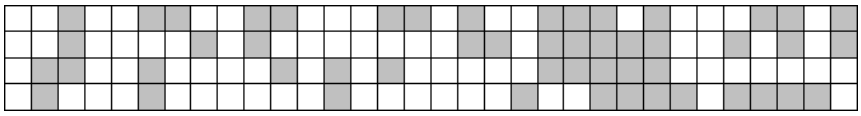

Figure 5: Active plaintext S-boxes in Biham et al.'s attack.

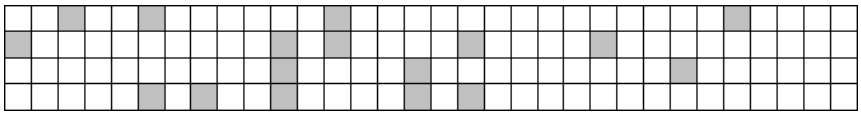

Figure 6: Active ciphertext S-boxes in Biham et al.'s attack.

Nguyen et al.'s multidimensional "Method 2" attack on 12-round Serpent [30] modifies Collard et al.'s 9-round approximation by adding a 56-dimensional approximation to the preceding round, resulting in a multidimensional 10-round approximation. The attack aims for maximum advantage $a = l = 172$ with $M = (2^{56} - 1)$.

There are two main statistical frameworks for multidimensional linear cryptanalysis; one based on the χ^2 statistic and the other on the log-likelihood ratio (LLR) [7]. None of the attacks in [30] are feasible with the χ^2 statistic, so we assume that the LLR is used and generalise Equation 4.1 to the multidimensional case:

$$\begin{aligned} P_s &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{x} f_W(y) \mathrm{d}y \right)^{2^l - 1} f_R(x) \mathrm{d}x \\ &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{x} (M+1) \left(\Phi_{\mu_W, \sigma_W^2}(y)^M \right) \frac{1}{\sigma_W} \phi \left(\frac{y - \mu_W}{\sigma_W} \right) \mathrm{d}y \right)^{2^l - 1} \frac{1}{\sigma_R} \phi \left(\frac{x - \mu_R}{\sigma_R} \right) \mathrm{d}x \\ &= \int_{-\infty}^{\infty} \left(\Phi_{\mu_W, \sigma_W^2}(x)^{M+1} \right)^{2^l - 1} \frac{1}{\sigma_R} \phi \left(\frac{x - \mu_R}{\sigma_R} \right) \mathrm{d}x \end{aligned}$$

For large M and l, this, and its generalisations when the WKRH does not hold, are not easy to calculate numerically. With $M = 2^{56} - 1$, Wolfram Mathematica fails to complete the calculation. As a result, we are currently forced to rely on the approximate statistical framework for the case in which key-ranking is used [7], based on Normally-approximated order statistics [18, 33, 7]. We begin by addressing an error in this framework.

Let b denote the value $\Phi^{-1}(\sqrt[M+1]{1 - 2^{-a}})$. The following equation is derived in [7] using the incorrect approximation $a \approx (b^2/2) - log_2(M + 1)$:

$$a \approx (\sqrt{NC} - \Phi^{-1}(P_s))^2/2 - log_2(M + 1) \tag{4.3}$$

Using the approximation $b \approx \Phi^{-1}(1 - 2^{-a - log_2(M+1)})$ instead, we obtain a very different equation:

$$a \approx 0.72(\sqrt{NC} - \Phi^{-1}(P_s))^2 + log_2(\sqrt{NC} - \Phi^{-1}(P_s)) + 1.325 - log_2(M + 1) \tag{4.4}$$

There is also the "linear hull" effect to consider. Approximations with the same input and output bitmasks, but following different paths through the cipher, may cause the actual distribution to differ

from that predicted theoretically. Figure 4 of [10] shows the results of experiments on a cipher similar to Serpent [25]. In these, as the number of rounds increases, the magnitude of the bias calculated with the Piling-Up Lemma increasingly underestimates that of the actual bias, and the extent of this underestimate varies significantly depending on the key value. The LLR statistic in multidimensional linear cryptanalysis rewards high Kullback-Leibler distance from the uniform distribution, and low distance from the theoretical distribution, equally [17], and the linear hull effect clearly interferes with the second part of this.

The capacity claimed by Nguyen et al. for this attack is 2^{-116}. However, this is incorrect:

- The various 2^{-4m} terms in their Equation 2 correspond to ± 2s in the columns of Serpent S2's linear approximation table, and should therefore be 2^{-3m}.
- The equation multiplies ($4\times$ the square of the bias of the rest of the approximation) by (the sum of some individual S-box biases). These S-box biases should be multiplied by 2 when calculating the overall bias using the Piling-Up Lemma, and Equation 2 should have multiplied by the sum of the squares of these doubled biases.
- The term 8^m assumes that all ± 2s in the relevant LAT columns for the relevant active S-boxes contribute towards the attack's capacity. However, any approximation which is the sum of an even number of the 56 base approximations will have output bitmask 0, and hence zero bias. After writing a script to quantify the effect of this, we discovered that at least four nonzero entries in the LAT columns for output bitmasks other than 0001, 1110 and 1111 must fail to contribute to the attack's capacity. The highest value this term can take is therefore 4^m.

We therefore recalculate the capacity as follows:

$$\begin{aligned} C &\leq (2^{-58})^2 \sum_{m=0}^{11} \binom{11}{m} 4^m 2^{11-m} 4^4 [(2^{15}(2^{-3m}2^{-2(11-m)}2^{-2\times 4}))^2] \\ &= 2^{-120.565} \end{aligned}$$

Solving Equation 4.4 with the recalculated capacity, we obtain a data complexity of $N \approx 2^{128.956}$, in excess of the size of the codebook and hence invalidating the 12-round Method 2 attack.

A wrong key is far more likely for this sort of attack to have a randomising effect, since each active S-box may contribute more than one bit or sum of bits to the 56 "base" approximations in the multidimensional attack, and a wrong key value for one S-box is less likely to flip all of these with high or indeed any bias than just one. We believe that the WKRH is sufficiently valid for the active plaintext boxes to make little or no difference to P_s in the multidimensional attacks of [30].

The "Method 1" attack from the same paper consists of 2^{128} separate 1R attacks with key guessing on 48 bits in the final round (only 46 of which we can recover). The data complexity for one such 1R attack must lower-bound the value of N. For capacity $2^{-118.565}$, we obtain $N \geq\approx 2^{125.813}$, but note that this may be adversely affected by the linear hull effect.

We also consider [30]'s attacks on 11-round Serpent. In the case of the attack with twelve active S-boxes in the final round, only 46 of the 48 attacked bits can be recovered. We solve Equation 4.4 for capacity $2^{-118.565}$ and $P_s = 0.785$, and obtain $N \approx 2^{125.813}$. In the case of the attack with eleven active final-round S-boxes, we obtain $N \approx 2^{127.784}$.

(These figures do not take into account the linear hull effect, as there is no way to quantify it.)

If the LLR statistic is used, the convolution method [22] for converting empirical probability distributions into scores for key candidates requires $2^k((6m+13)\cdot 2^m)$ MAs + $2^k((6m+4)\cdot 2^m)$ AOs.

Finally, we consider the differential-linear attacks. In Indesteege et al.'s chosen-ciphertext attack on 11-round Serpent [20], two active plaintext boxes (both Serpent S4) contribute bit y_4 to the attack. Since three input differences cause this bit to flip with bias ± 0.5, the attack recovers 56 key bits instead of 60. For the other differential-linear attacks [20], all key bits are obtainable and the effect of the bias ± 0.25 parity flips on P_s is negligible.

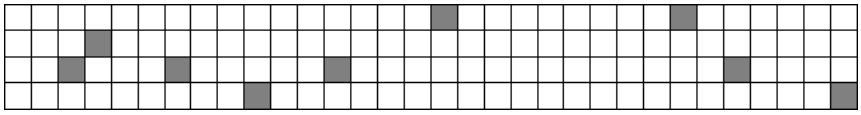

Figure 7: Active plaintext S-boxes in Biham et al.'s reverse-direction CC differential-linear attack.

Rounds	Type of attack	Data	Time (analysis)
11	Linear [2]	$2^{121.728}$ KP	$2^{188.1}$ E
11	Linear [2]	$2^{121.728}$ KP	2^{96} PE + 2^{44} PD + $2^{149.73}$ AO + $2^{149.76}$ MA
11	Linear [13]	$2^{121.234}$ KP	2^{60} PE + 2^{48} PD + $2^{117.36}$ AO + $2^{117.4}$ MA
11	**Multidim. lin.** [30]	$2^{125.813}$ KP	2^{48} PD + $2^{114.087}$ AO + $2^{114.134}$ MA
11	**Multidim. lin.** [30]	$2^{127.784}$ KP	2^{44} PD + $2^{110.055}$ AO + $2^{110.103}$ MA
11	Differential-linear [20]	$2^{121.8}$ CP	$2^{135.7}$ MA
11	Nonlinear (this paper)	$2^{120.357}$ KP	2^{80} PE + 2^{48} PD + $2^{139.6}$ AO + $2^{139.63}$ MA
11	Nonlinear (this paper)	$2^{117.317}$ KP	2^{60} PE + 2^{76} PD + $2^{149.69}$ AO + $2^{149.72}$ MA
11	Filtered NL (this paper)	$2^{116.508}$ KP	2^{60} PE + 2^{76} PD + $2^{151.69}$ AO + $2^{151.72}$ MA
11	Nonlinear (this paper)	$2^{115.44}$ KP	2^{60} PE + 2^{80} PD + $2^{153.73}$ AO + $2^{153.76}$ MA
11	Filtered NL (this paper)	$2^{114.55}$ KP	2^{60} PE + 2^{80} PD + $2^{155.73}$ AO + $2^{155.76}$ MA
11	Differential-linear [20]	$2^{113.7}$ CC	$2^{137.7}$ MA
12	Differential-linear [20]	$2^{123.5}$ CP	$2^{249.4}$ E
12	**Multidim. lin.** [30]	$\geq 2^{125.813}$ KP	2^{128} PE + 2^{48} PD + $2^{242.087}$ AO + $2^{242.134}$ MA

Table 2: Attack complexities. In most cases $P_s = 0.785$ (or slightly higher.) The chosen plaintext attacks of Biham et al. have $P_s = 0.84$, and the chosen-ciphertext attack has $P_s = 0.93$. The time complexity for Biham et al.'s linear cryptanalysis varies depending on whether the new analysis method of Collard et al. is used, or whether an earlier method [2] is. Table entries in bold signify that the method may not work as claimed depending on the linear hull effect. E = full encryptions of the reduced round cipher. PE = partial encryptions. PD = partial decryptions.

Rounds	Type of attack	Time (analysis) summary	Mem	Bits recovered
11	Linear [2]	$2^{188.1}$ E	*	130
11	Linear [2]	$2^{137.08}$ E + $2^{149.76}$ MA	$2^{144.087}$	130
11	Linear [13]	$2^{104.71}$ E + $2^{117.4}$ MA	$2^{112.087}$	98
11	**Multidim. linear** [30]	$2^{101.437}$ E + $2^{114.134}$ MA	2^{108}	46
11	**Multidim. linear** [30]	$2^{97.405}$ E + $2^{110.103}$ MA	2^{104}	44
11	Differential-linear [20]	$2^{135.7}$ MA	2^{76}	48
11	Nonlinear (this paper)	$2^{126.95}$ E + $2^{139.63}$ MA	$2^{133.17}$	118 k_1
11	Nonlinear (this paper)	$2^{137.04}$ E + $2^{149.72}$ MA	$2^{141.585}$	128 k_1, 4 k_2
11	Filtered NL (this paper)	$2^{139.04}$ E + $2^{151.72}$ MA	$2^{142.585}$	128 k_1, 6 k_2
11	Nonlinear (this paper)	$2^{141.08}$ E + $2^{153.76}$ MA	$2^{145.585}$	130 k_1, 4 k_2
11	Filtered NL (this paper)	$2^{143.08}$ E + $2^{155.76}$ MA	$2^{146.585}$	132 k_1, 6 k_2
11	Differential-linear [20]	$2^{137.7}$ MA	2^{99}	56
12	Differential-linear [20]	$2^{249.4}$ E	$2^{128.5}$	160
12	**Multidim. linear** [30]	$2^{229.437}$ E + $2^{242.134}$ MA	2^{108}	174

Table 3: Attack complexities cont. Memory is measured in bytes. The memory required for the attack of [2] when the analysis method of [13] is not used is unclear, and the relevant sources [2, 13] disagree on this. $2^{12.65}$ AOs are needed for an 11-round Serpent encryption, and $2^{12.78}$ AOs for twelve rounds.

4.2 Nonlinear attacks on 11-round Serpent-192 and Serpent-256.

In this section, we describe various modifications to Collard et al.'s Approximation D2 [12], which will allow us to attack 11-round Serpent using nonlinear approximations.

The simplest change is to replace the (input bitmask 12, output bitmask 10) approximation in the first round (affecting bits 16, 17, 18, 19) with $x_2 \oplus x_1 \oplus x_1x_4 = y_1 \oplus y_3$. Doing this gives us 20 active plaintext S-boxes, increasing $|k_1|$ to 128. However, four such boxes contribute a bit (or parity) that can flip with bias ± 0.5 for various input differences, as does one of the active ciphertext boxes. We can therefore recover only 118 bits of k_1. One of the bits affected by this is involved in the quadratic term of the nonlinear approximation, so we cannot recover any k_2 bits. The memory requirements are increased to $2^{(128+2)} + 2^{(128+5)} = 2^{133.17}$ bytes.The time complexity of the analysis phase also increases, and is dominated by $4 \cdot (6 \times 128 + 8) \cdot 2^{128} = 2^{139.6}$ AOs and $4 \cdot (6 \times 128 + 26) \cdot 2^{128} = 2^{139.63}$ MAs.

We estimate the number of AOs per reduced-round encryption by counting the number of AOs in Serpent's bitslice implementation [1]. We assume that Osvik's implementation of S-box 6 [31] is used. This gives us $2^{12.65}$ AOs per 11-round encryption, and $2^{12.78}$ per 12-round encryption. We therefore obtain time complexity of $2^{126.95}$ encryptions + $2^{139.63}$ MAs.

The capacity is multiplied by $(6/4)^2 = 2.25$. The increased number of k_1 bits, and the need to deal with 2^2 relateds, effectively raise l to 130. The ten bits which cannot be deduced reduce this to 120 for the purposes of calculating N and we obtain $P_s = 0.8$ with $N = 2^{120.357}$.

We now consider a situation in which the entire first round approximation remains linear. We replace the final-round $x_1 \oplus x_3 \oplus x_4 = y_2$ approximation on state bits 96-99 with $x_1 \oplus x_3 \oplus x_4 = y_2 \oplus y_1 \oplus y_2y_4$, and also replace $x_3 \oplus x_4 = y_4$ (bits 76-79) with $x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_1y_3$. The total number of active S-boxes increases to 34. We have replaced a bias 4 approximation and a bias 2 approximation with two bias 6 approximations, multiplying capacity by $\left(\frac{6\times6}{4\times2}\right)^2 = 20.25$. The value of l is effectively increased to (140-8) = 132 for the purposes of calculating N, and $N = 2^{117.317}$ gives $P_s \approx 0.8$. The memory requirements are increased to $2^{141.585}$. The time complexity of the analysis phase is dominated by $16 \cdot (6 \cdot 136 + 26) \cdot 2^{136} = 2^{149.72}$ MAs and $16 \cdot (6 \cdot 136 + 8) \cdot 2^{136} = 2^{149.69}$ AOs.

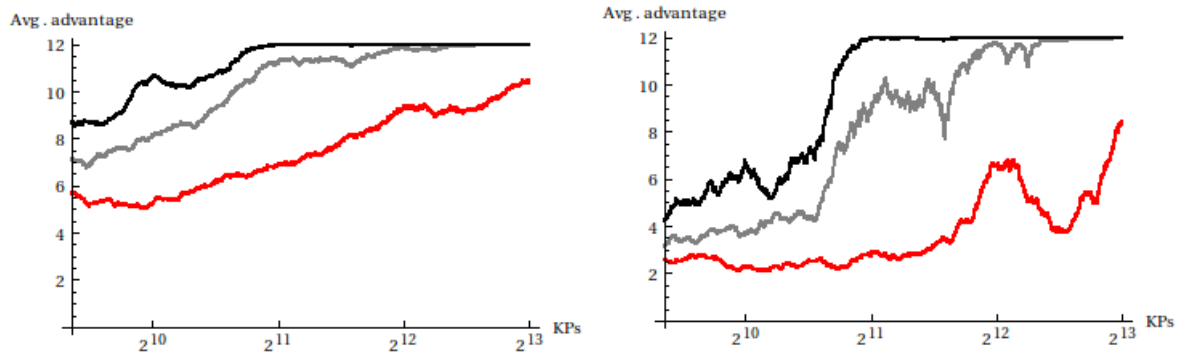

Figure 8: Left-hand graph shows mean advantages for attack on four round SPN with 4×4 S-boxes using: linear approximation (red), nonlinear approximation (grey), nonlinear approximation in filtered nonlinear attack (black). Right-hand graph shows results of alternate calculation for average advantage in which the mean rank obtained was input to the formula for advantage.

If we utilise filtering here, the number of active S-boxes does not change. The biases of the eight relateds for each of the S-box approximations become $(6, -4, -4, 2, 0, 0, 0, 0)$, allowing us to attack all six k_2 bits. Memory requirements increase to $2^{142.585}$, and the time complexity of the analysis phase increases to $64 \cdot (6 \cdot 136 + 26) \cdot 2^{136} = 2^{151.72}$ MAs and $64 \cdot (6 \cdot 136 + 8) \cdot 2^{136} = 2^{151.69}$ AOs. For the purposes of calculating N, l is effectively equal to 134, and the capacity is multiplied by $(16/9)^2$. However, the need to effectively discard 9/16 of our KP pairs means we must calculate N as if it were only multiplied by (16/9), and we obtain $N = 2^{116.508}$.

To reduce the data complexity further, we could replace the $x_1 \oplus x_3 \oplus x_4 = y_2$ approximation on

state bits 116-119 with a nonlinear approximation, instead of replacing $x_3 \oplus x_4 = y_4$. The capacity is then multiplied by 81 instead of 20.25, and we activate 35 S-boxes. We obtain time complexity $16 \cdot (6 \cdot 140 + 26) \cdot 2^{140} = 2^{153.76}$ MAs and $16 \cdot (6 \cdot 140 + 8) \cdot 2^{140} = 2^{153.73}$ AOs with memory complexity $2^{145.585}$. For the purposes of calculating N, l is effectively increased to (144-10)=134, and $N = 2^{115.338}$ yields $P_s = 0.8$.

Again, we can employ filtering here. The number of active S-boxes is still 35, and all six k_2 bits can now be attacked. The memory complexity increases to $2^{146.585}$, and the time complexity to $2^{155.76}$ MA and $2^{155.73}$ AO. To calculate N, since one less S-box is affected by the "WKRH max-bias" issue than before, l is effectively (146-8)=138 and we obtain $N = 2^{114.55}$.

5 Conclusion.

We have obtained nonlinear approximations for block cipher S-boxes with higher absolute bias than their best linear approximations. We have also derived algorithms which can use the new approximations in attacks, and calculated the complexities for these new attacks. Having done this, we have presented nonlinear attacks on 11-round Serpent with better data complexity than any other known-plaintext attack, as well as the best time complexity of any attack so far on 11-round Serpent-256.

References

[1] R. Anderson, E. Biham, and L. Knudsen. Serpent: A Proposal for the Advanced Encryption Standard. `http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf`.

[2] E. Biham, O. Dunkelman, and N. Keller. Linear cryptanalysis of reduced round Serpent. In M. Matsui, editor, *Proceedings of the Eighth International Workshop on Fast Software Encryption (FSE 2001)*, volume 2355 of *Lecture Notes in Computer Science*, pages 16–27. IACR, Springer, April 2001.

[3] E. Biham, O. Dunkelman, and N. Keller. New results on boomerang and rectangle attacks. In J. Daemen and V. Rijmen, editors, *Proceedings of the Ninth International Workshop on Fast Software Encryption (FSE 2002)*, volume 2365 of *Lecture Notes in Computer Science*, pages 1–16. IACR, Springer, February 2002.

[4] E. Biham, O. Dunkelman, and N. Keller. Differential-linear cryptanalysis of Serpent. In T. Johansson, editor, *Proceedings of the Tenth International Workshop on Fast Software Encryption (FSE 2003)*, volume 2887 of *Lecture Notes in Computer Science*, pages 9–21. IACR, Springer, February 2003.

[5] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S.M. Matyas Jr, L. O'Connor, M. Peyravian, D. Safford, and N. Zunic. MARS - a candidate cipher for AES. Technical report, IBM, September 1999. `http://www.research.ibm.com/security/mars.pdf`.

[6] C. Carlet. Boolean functions for cryptography and error-correcting codes. In Y. Crama and P. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Cambridge University Press, 2010. The chapter is downloadable from `http://www.math.univ-paris13.fr/~carlet/chap-fcts-Bool-corr.pdf`.

[7] J.Y. Cho, M. Hermelin, and K. Nyberg. Multidimensional extension of Matsui's algorithm 2. In O. Dunkelman, editor, *Proceedings of the Sixteenth International Workshop on Fast Software Encryption (FSE 2009)*, volume 5665 of *Lecture Notes in Computer Science*, pages 209–227. IACR, Springer, February 2009.

[8] J.A. Clark, J.C Hernández-Castro, and J.M.E. Tapiador. Non-linear cryptanalysis revisited: Heuristic search for approximations to S-boxes. In S.D. Galbraith, editor, *Proceedings of the 11th IMA International Conference on Cryptography and Coding*, volume 4887 of *Lecture Notes in Computer Science*, pages 99–117. Springer, December 2007.

[9] J.A. Clark and J.D. McLaughlin. Nonlinear cryptanalysis of reduced-round Serpent and meta-heuristic search for s-box approximations. Cryptology ePrint Archive, Report 2013/. January 2013. `http://eprint.iacr.org/2013/`.

[10] B. Collard and F.-X. Standaert. Experimenting linear cryptanalysis. 2011. `http://perso.uclouvain.be/fstandae/PUBLIS/90.pdf`.

[11] B. Collard, F.-X. Standaert, and J.-J. Quisquater. Improved and multiple linear cryptanalysis of reduced round Serpent. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Proceedings of the 3rd SKLOIS Conference on Information Security and Cryptology (Inscrypt 2007)*, volume 4990 of *Lecture Notes in Computer Science*, pages 383–398. Springer, August 31 - September 5 2007.

[12] B. Collard, F.-X. Standaert, and J.-J. Quisquater. Improved and multiple linear cryptanalysis of reduced round Serpent - description of the linear approximations. 2007. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.522&rep=rep1&type=pdf`.

[13] B. Collard, F.-X. Standaert, and J.-J. Quisquater. Improving the time complexity of Matsui's linear cryptanalysis. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *Proceedings of the 10th International Conference on Information Security and Cryptology (ICISC 2007)*, volume 4817 of *Lecture Notes in Computer Science*, pages 77–88. Springer, November 2007.

[14] N.T. Courtois. Feistel schemes and bi-linear cryptanalysis (extended abstract). In M. Franklin, editor, *Advances in Cryptology - Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 23–40. IACR, Springer, August 2004.

[15] N.T. Courtois. Feistel schemes and bi-linear cryptanalysis. Cryptology ePrint Archive, Report 2005/251. August 2005. `http://eprint.iacr.org/2005/251`.

[16] N.T. Courtois, D. Hulme, and T. Mourouzis. Solving circuit optimisation problems in cryptography and cryptanalysis. Cryptology ePrint Archive, Report 2011/475. September 2011. `http://eprint.iacr.org/2011/475`.

[17] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley-Interscience, second edition, 2006.

[18] H.A. David. *Order Statistics*. Wiley, second edition, 1981.

[19] O. Dunkelman. Private communication.

[20] O. Dunkelman, S. Indesteege, and N. Keller. A differential-linear attack on 12-round Serpent. In D.R. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - Indocrypt 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 308–321. Springer, December 2008.

[21] C. Harpes, G.G. Kramer, and J.L. Massey. A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma. In L.C. Guillou and J-J. Quisqater, editors, *Advances in Cryptology - Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 24–38. IACR, Springer, 1995.

[22] M. Hermelin and K. Nyberg. Dependent linear approximations: The algorithm of Biryukov and others revisited. In J. Pieprzyk, editor, *Proceedings of the Cryptographers' Track at the RSA Conference, 2010 (CT-RSA 2010)*, volume 5985 of *Lecture Notes in Computer Science*, pages 318–333. Springer, March 2010.

[23] L.R. Knudsen and M.J.B. Robshaw. Non-linear approximations in linear cryptanalysis. In U. Maurer, editor, *Advances in Cryptology - Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 224–236. IACR, Springer, 1996.

[24] H.O. Kunz. On the equivalence between one-dimensional discrete Walsh-Hadamard and multidimensional discrete Fourier transforms. *IEEE Transactions on Computers*, C-28(3):267–268, March 1979.

[25] G. Leander. Small scale variants of the block cipher PRESENT. Cryptology ePrint Archive, Report 2010/143. March 2010. `http://eprint.iacr.org/2010/143`.

[26] G. Leander. On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In K.G. Paterson, editor, *Advances in Cryptology - Eurocrypt 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 303–322. IACR, Springer, May 2011.

[27] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology - Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. IACR, Springer, 1993.

[28] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y.G. Desmedt, editor, *Advances in Cryptology - Crypto '94*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. IACR, Springer, 1994.

[29] S. Murphy. The effectiveness of the linear hull effect. Technical Report RHUL-MA-2009-19, Royal Holloway, University of London, October 2009. `http://www.isg.rhul.ac.uk/~sean/Linear_Hull_JMC-Rev2-llncs.pdf`.

[30] Phuong Ha Nguyen, Hongjun Wu, and Huaxiong Wang. Improving the algorithm 2 in multidimensional linear cryptanalysis. In Udaya Parampalli and Philip Hawkes, editors, *Proceedings of the Sixteenth Australasian Conference on Information Security and Privacy (ACISP 2011)*, volume 6812 of *Lecture Notes in Computer Science*, pages 61–74. Springer, July 2011.

[31] D.A. Osvik. Speeding up Serpent. In *Proceedings of the 3rd Advanced Encryption Standard Candidate Conference (AES 2000)*, April 2000.

[32] B. Preneel, B. Van Rompay, S. B. Ors, A. Biryukov, L. Granboulan, E. Dottax, M. Dichtl, M. Schafheutle, P. Serf, S. Pyka, E. Biham, E. Barkan, O. Dunkelman, J. Stolin, M. Ciet, J-J. Quisquater, F. Sica, H. Raddum, and M. Parker. Performance of optimized implementations of the NESSIE primitives (version 2.0). February 2003. `http://www.cosic.esat.kuleuven.be/nessie/deliverables/D21-v2.pdf`.

[33] A.A. Selçuk. On probability of success in linear and differential cryptanalysis. *Journal of Cryptology*, 21(1):131–147, January 2008.

Appendices

A Errors in the description of the Dunkelman/Keller approximation

In the original description of Biham et al.'s linear approximation [2], on page 20, after S6 is applied the only active bit in the state is bit 30. In later papers [4, 20], after the application of S6, bit 28 is shown as active instead of bit 30. In private email correspondence, one of the authors informed us that bit 28 was correct.

The Serpent diffusion layer is then applied, after which the active bits according to the diagram are 80, 101 and 103. However, the xor of diffusion layer output bits {80, 101, 103} is the xor of input

bits {4, 22, 35, 44, 46, 57, 62, 75, 86, 96, 97} - and is therefore unaffected by either bit 28 or bit 30. In the same correspondence mentioned above, this was revealed to be a typographical error - the active bits shown at this point should have been 81, 83 and 100.