

# Nonlinear cryptanalysis of reduced-round Serpent and metaheuristic search for S-box approximations.

James McLaughlin\*, John A. Clark

## Abstract

We utilise a simulated annealing algorithm to find several nonlinear approximations to various S-boxes which can be used to replace the linear approximations in the outer rounds of existing attacks. We propose three variants of a new nonlinear cryptanalytic algorithm which overcomes the main issues that prevented the use of nonlinear approximations in previous research, and we present the statistical frameworks for calculating the complexity of each version. We present new attacks on 11-round Serpent with better data complexity than any other known-plaintext or chosen-plaintext attack, and with the best overall time complexity for a 256-bit key.

**Keywords:** Nonlinear cryptanalysis, generalized linear cryptanalysis, metaheuristics, simulated annealing, multidimensional linear cryptanalysis, Serpent.

## 1 Introduction.

The basic linear cryptanalytic method [52, 53] has had several extensions and variations proposed since its discovery in 1993. The use of multiple approximations was first seen, in a somewhat ad hoc way with limited scope for generalisation, in 1994 [53]. Later that same year, Kaliski and Robshaw conducted a dedicated investigation into linear cryptanalysis with multiple approximations [41], and subsequent research in the use of multiple approximations [9, 55] finally culminated in the new method known as *multidimensional* linear cryptanalysis [15, 16, 17, 18, 13, 60], used in the best cryptanalysis to date of reduced-round PRESENT [10] [14] and Serpent [1] [60].

Another research direction proposed was the generalisation of the method to make use of *nonlinear* approximations. That is, instead of being restricted to equations of the form  $x_{a_1} \oplus x_{a_2} \oplus \dots \oplus x_{a_i} \oplus y_{b_1} \oplus y_{b_2} \oplus \dots \oplus y_{b_j}$  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  $x_{a_1}x_{a_3}$  - in other words, terms that needed the AND operation to be evaluated.

This was first proposed by Harpes, Kramer and Massey [36], and investigated in more depth by Knudsen and Robshaw [46], in which 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 (as described by Knudsen and Robshaw) 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  $6 \times 4$  S-box, the search space for possible approximations was  $2^{64}$  in size, increasing to  $2^{256}$  for an AES-sized  $8 \times 8$  S-box. This was handled by restricting the search to nonlinear approximations with degree below a certain threshold  $d$ ; significantly reducing the size of the search space but also preventing better approximations of higher degree from being found [45].

The assumption that nonlinear approximations could only be used in the outer rounds of the distinguisher was partially challenged by Courtois [26, 27]. Courtois 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 bi-linear expression using no nonlinear parts that were not of the form

---

\*Corresponding author, [jmclaugh@cs.york.ac.uk](mailto:jmclaugh@cs.york.ac.uk)

$(L_{i_\alpha} \oplus L_{i_\beta} \oplus \dots \oplus L_{i_\omega}) \cdot (R_{i_a} \oplus R_{i_b} \oplus \dots \oplus R_{i_q})$  (where  $L_i$  and  $R_i$  were variables from the left and right-hand ciphertext blocks in round  $i$  respectively). He did, however, have to accept a certain amount of *key-dependence*, in that a given bi-linear approximation  $B$  could hold with lower bias for some key values than with others. His attack also strongly relied on the Feistel structure, and could not be generalised to attack SPN-based ciphers.

The first, and so far only, use of metaheuristics in the context of nonlinear cryptanalysis was the use of simulated annealing by Clark et al. [21] to evolve nonlinear approximations to the MARS S-box [11] of the form  $f(x_{i_1}, x_{i_2}, \dots, x_{i_k}) = (y_{j_1} \oplus y_{j_2} \oplus \dots \oplus y_{j_k})$ , for use in the first round of nonlinear distinguishers. Their approach, building on similar work in the context of stream ciphers [20], found various nonlinear approximations holding with a significantly higher absolute bias (151/512) than the best-known linear approximations for the MARS S-box (84/512). However, no means of exploiting these in an attack on reduced-round MARS was known.

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

- We look at the question of which moves have a smooth search landscape defined when they are used as the move function in a local optimisation-based metaheuristic for finding nonlinear S-box approximations. We adapt the metaheuristic search method of Clark et al. to prioritise these over the other move types previously defined.
- 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  $n_{\alpha_1 \alpha_2 \dots \alpha_l} = n_0(x_1 \oplus k_{\alpha_1}, x_2 \oplus k_{\alpha_2}, \dots, x_l \oplus k_{\alpha_l})$ . There exist  $2^l$  candidates for the correct function,  $n_i$ , to compute on these bits, and the cryptanalyst does not know which is correct. Furthermore, the incorrect functions may still define approximations with nonzero bias, and hence
  - may contain additional information that would be of use even if the cryptanalyst *did* know the correct function.
  - may not be possible to distinguish from the correct approximation if their biases are too close to each other.

We devise various statistical frameworks for nonlinear modifications of Matsui’s Algorithm 2 which can succeed in spite of - or even with the assistance of - these “related” functions (or, equivalently, the “related approximations” they define). We describe the new attack, and how to incorporate recent advances in linear cryptanalysis into it. We also adapt the metaheuristic search algorithm to take into account the properties of the related functions. Finally, we present newly-obtained nonlinear approximations for the S-boxes of various ciphers, with bias in excess of the best linear approximations for the same, and utilise these in new attacks on reduced-round Serpent.

Figure 1 below depicts a 1R nonlinear approximation, which we have used successfully in attacking the Heys toy cipher [40]. The  $(r - 1)$ -round approximation is composed of an  $(r - 2)$ -round linear approximation, followed by a nonlinear approximation which replaces the linear approximation to round  $(r - 1)$ . As explained above, the cryptanalyst cannot simply guess the bits from the final round key, but is also forced to deal with the incorrect approximations (and one correct approximation) derived from guessing the involved values of the penultimate round key bits.

This paper is structured as follows: The remainder of this section provides a brief description of linear cryptanalysis, as well as an important refinement to it due to Collard et al. [25]. Section 2 discusses the ways in which using nonlinear approximations affects the attack. Section 3 describes the new attack, including the adaptation of Collard et al.’s improved methodology to the nonlinear domain and the Feistel structure, and addresses the question of how its complexity is to be calculated.

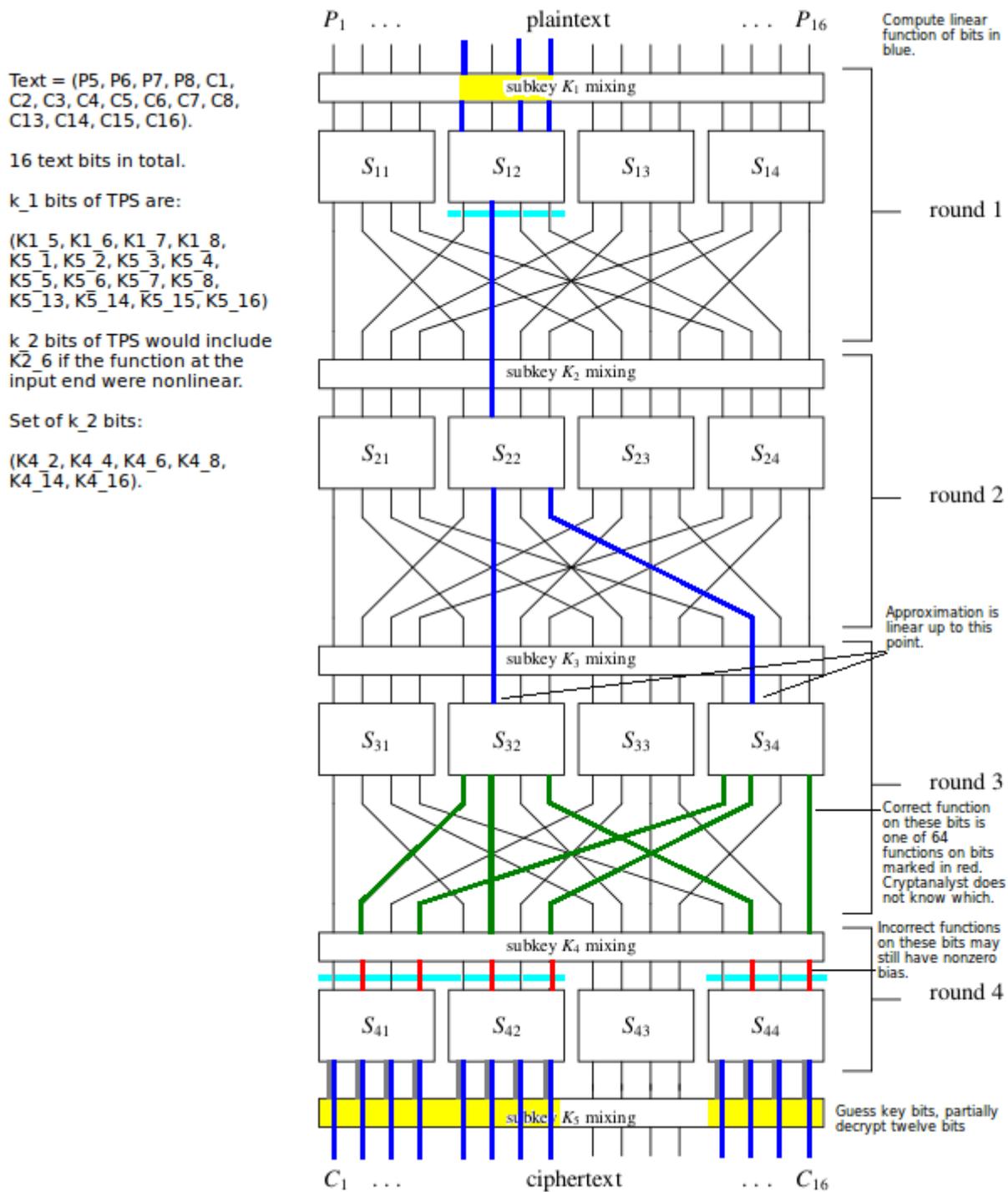


Figure 1: Diagram of a 1R nonlinear approximation to the Heys toy cipher [40].

At this point in the paper, we will have described a well-defined scenario in which we can use the evolved nonlinear approximations, and will have addressed in detail the question of how the related approximations can and should be handled. This means that we will finally be in a position where we can construct cost functions taking all this into account, and so in Section 4 we will finally describe our experiments with the new simulated annealing algorithm; including how it differs from that originally used by Clark et al. [21].

A large proportion of Section 3 is focused on the Data Encryption Standard, as the best-known example of a Feistel cipher. Section 5 discusses the application of the new technique to the DES further. It also discusses the application of the new technique to other ciphers. In particular, we:

- Describe new attacks on reduced-round Serpent,
- Give the results of our search for S-box approximations for the AES, DES and PRESENT S-boxes,
- Demonstrate the workings of an attack on the DES using the best of the new approximations. The attack is not in practice as efficient as the best-known attack against DES [53]; it is presented chiefly to demonstrate how a nonlinear attack on a Feistel cipher would work.

Finally, Section 6 discusses avenues for further research.

### 1.1 Linear cryptanalysis - the three main phases of an Algorithm 2 attack.

Unless otherwise stated, the linear cryptanalytic attack will be assumed to be a 2R attack, in which the cryptanalyst knows of a linear approximation to rounds  $2, 3, \dots, (r-1)$  of the cipher, and by using candidate key bit values to partially decipher parts of the known ciphertexts (reversing the effects of round  $r$  on certain key bits), as well as to partially encrypt certain bits in the known plaintexts, obtains values for the bits on which the probabilistic linear relation should hold. Note in particular that, unlike the 1R attack shown in the earlier diagram, this means that key bits are guessed in both round 1 and round  $r$ .

The theoretical bias for this linear approximation is calculated by starting with the biases of the linear approximations to each individual S-box, and then using the *Piling-Up Lemma* [52]:

**Definition 1.1.** For  $1 \leq i \leq n$ , let  $X_i$  be independent Bernoulli random variables such that:

$$\begin{aligned} p_i &= P(X_i = 0) \\ (1 - p_i) &= P(X_i = 1) \end{aligned}$$

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

Then  $P(X_1 \oplus X_2 \oplus \dots \oplus X_n = 0)$  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 yields an estimate of the true bias. This is usually accurate enough for the purposes of cryptanalysis, although situations where it is not are discussed by Murphy [56] and Leander [51].

**Definition 1.2.** Where a linear approximation holds with bias  $\epsilon$ , i.e. with probability  $1/2 + \epsilon$ , the *capacity* 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  $\epsilon_i$ , the capacity of the set of the approximations is  $4 \sum_{i=1}^M \epsilon_i^2$ .

We note in particular that the bias  $(p - 1/2)$  of the linear approximation  $x_{a_1} \oplus x_{a_2} \oplus \dots \oplus x_{a_n} = y_{b_1} \oplus \dots \oplus y_{b_m}$  as calculated using the Piling-Up Lemma may be either positive or negative, and that the values of various bits in the round keys affecting the approximated rounds may cause the actual bias to possess the opposite sign. In Algorithm 2, the cryptanalyst is only interested in the magnitude of the bias and hence this is not a problem; whereas in Algorithm 1 the cryptanalyst actively exploits this phenomenon to deduce the parity of said key bits.

A linear cryptanalytic attack may be divided into three main phases; each of which we need to calculate the complexity of separately. These are:

1. The *distillation phase*. In this phase, the cryptanalyst has access to  $N$  pairs (plaintext, corresponding ciphertext), all encrypted with the same key  $k_0$ . These are “known plaintext” pairs, as opposed to “chosen plaintext”, because the cryptanalyst is not assumed to have been able to make any choices regarding any of the  $N$  plaintext values encrypted. The cryptanalyst needs to extract the relevant data from these pairs and discard the rest.

Certain bit positions in the plaintext and ciphertext will have been identified as relevant, in that they are the bits which must be partially encrypted/decrypted to obtain the values of the bits involved in the approximation. Let the number of such positions be denoted  $l$ . The cryptanalyst allocates memory for an array *COUNTERS\_1* of  $2^l$  integer variables, each of which must be capable of holding any integer between 0 and  $N$ , and initialises these to 0. These variables are the first of several sets of counters used in the attack.

For each known plaintext/ciphertext pair in turn, the cryptanalyst extracts the  $l$  relevant data bits. Where  $j$  denotes the  $l$ -bit value corresponding to the values of the  $l$  bits, the cryptanalyst increments the value in *COUNTERS\_1*[ $j$ ] by 1, discards the current pair, and moves on to the next pair until all  $N$  pairs have been processed.

Clearly this phase has complexity  $O(N)$ . Let  $\epsilon$  denote the bias of the linear approximation, then the cryptanalyst needs  $N$  to be equal (for some  $a$ ) to  $a/\epsilon^2$ . Advice on the value of  $a$  to choose to achieve a desired success probability was provided in Matsui’s original paper [52], and later updated with a more accurate statistical framework by Selçuk [66].

2. The *analysis phase*. We shall refer to the set of key bits which are to be recovered as the *target partial subkey* (TPS). Let  $k$  denote the number of bits therein. For most ciphers,  $k$  and  $l$  will be equal; however DES’s expansion phase makes it an example of a cipher for which this is not the case.

The cryptanalyst allocates memory for an array of integers, *COUNTERS\_2*, with  $2^k$  entries, such that each array entry should be able to take any value between 0 and  $N$ . She then, for every possible TPS value  $i$ , uses it to partially encrypt/decrypt every possible value  $j$  of the relevant text bits in turn. If the linear approximation holds for the pair  $(i, j)$ , *COUNTERS\_2*[ $i$ ] is incremented by the value in *COUNTERS\_1*[ $j$ ].

When this process is complete, the values in *COUNTERS\_2* should be converted into the absolute values of the biases with which the approximation held for the various key guesses; this is done by mapping each value *COUNTERS\_2*[ $i$ ] to  $v[i] = |\text{COUNTERS}_2[i] - N/2|$ . The higher the value of  $v[i]$ , the more likely it is that  $i$  is the correct TPS.

This phase as described has  $O(2^{k+l})$  (usually  $2^{2k}$ ) time complexity, in that this many partial encryptions/decryptions must be carried out, each potentially requiring data to be written to an array in memory. However, this is the phase for which the aforementioned improvement exists, which we will soon address.

3. The *search phase*. During this, the correct value of the TPS bits must be obtained from the counter values calculated in the analysis phase, and the remaining key bits must also be found.

It may be that the cryptanalyst will simply accept the highest value in *COUNTERS\_2* as corresponding to the correct key guess. The correct key guess should have reversed the effect of the outer rounds and yielded bits for which this high bias was expected; the wrong key guesses, by contrast, would not have resulted in the data bits being mapped to such values and would in effect have applied a function with a randomizing effect to them.

If the cryptanalyst proceeds thus, various formulae exist in terms of the approximation's bias  $\epsilon$  [52, 66] which can be used to calculate the number of known plaintexts  $N$  required for the attack to succeed with probability  $p$ . The cryptanalyst needs  $O(2^k)$  time to search the array  $v$  for the highest value therein. After this, where  $K$  denotes the key size of the cipher, the cryptanalyst is faced with the problem of finding the remaining  $(K - k)$  key bits, requiring an exhaustive search (time complexity  $O(2^{K-k})$  encryptions), unless further attacks (whether linear with another approximation, or some other technique) can be applied to recovering some or all of these bits.

However, this is not always the strategy employed. The number of possible values for the TPS bits involved is likely to be extremely high, and some of these will result, by pure chance, in high biases themselves (rather than the expected near-zero bias). If the number of known plaintexts  $N$  does not provide a sufficiently large sample, some of these biases may be more extreme than that for the correct TPS.

In Matsui's attack on the full DES [53], this behaviour was predicted, and dealt with in a way that allowed a much lower value of  $N$  to be used than would otherwise have been the case. The correct key was expected to result in one of the  $X$  highest-ranking values of  $v[i]$  (in this case,  $X$  was equal to  $2^{13}$ ), but not necessarily the highest such value. With this as the goal of the previous phases, the data complexity was much lower than it would have been had the correct key been required to yield the highest value of  $v[i]$ . However, this came at the cost of increased time complexity, as the search for the remaining  $2^{K-k}$  key bits was repeated for up to  $X$  different TPS candidates.

This technique is known as *key ranking*.

The complexity of sorting the vector  $v$  to identify the highest biases is  $O(k2^k)$ . It may, for small values of  $X$ , be faster simply to search for the  $X$  highest biases, in at most  $O(X2^k)$  time. In either case, starting with the highest-ranked TPS candidate, and continuing on for each successive candidate until the right one is found among the  $X$  highest ranked keys, the cryptanalyst must (probably through exhaustive search), search for a value for the remaining  $(K - k)$  key bits such that the full key value resulting correctly decrypts the known ciphertexts.

Without key ranking, this stage should be presumed to have complexity  $O(2^k + 2^{K-k})$  unless there is reason (such as another high-bias linear approximation involving the other key bits) to believe that the remaining key bits can be obtained without exhaustive search. With key ranking, this stage has a higher complexity of  $O(\min(k, X) \cdot 2^k + X \cdot 2^{K-k})$ , although as stated the use of key ranking will probably reduce the number of known plaintexts needed and hence reduce the time complexity of the distillation phase and the data complexity of the attack as a whole.

### 1.1.1 The improved method for the analysis phase, due to Collard, Standaert and Quisquater.

(Note: we now switch from referring to the  $i$ th element of an array  $a$  as  $a[i]$ , and will henceforth use the notation  $a_i$ .)

Other methods for the analysis phase do exist, including one used by Biham, Dunkelman and Keller [6] to overcome a situation in which the time complexity for the naive method described above

would have been infeasible. We focus here on a newer method with very significantly improved time complexity due to Collard, Standaert and Quisquater [25]; originally defined for 2R linear attacks, and later adapted to 1R attacks by Nguyen, Wu and Wang [60].

(In both cases, the method applied only to SPN ciphers like AES and Serpent where the number of active key bits was equal to the number of active text bits (let  $k$  denote this number), and not to Feistel ciphers such as DES. In the particular case of DES, the expansion phase of the round function was another factor inhibiting compatibility, in addition to the Feistel structure. We will describe an adaptation of the method that overcomes these obstacles later on in this paper.)

In this method (in the notation of Collard et al.):

- $N$ , as stated earlier, is the number of known plaintext/ciphertext pairs.
- $k$ , as also stated earlier, is the number of key bits in the TPS. It was originally assumed [25] that this was also the number  $l$  of data bits that had to be partially encrypted/decrypted, since for an SPN cipher each of these key bits would be xored with its corresponding data bit during said process. In the interests of simplicity, we will limit ourselves for the time being to ciphers such that this assumption is valid.
- $C$  is a  $2^k \times 2^k$  matrix. If the approximation holds for target partial subkey value  $i$  and value  $j$  for the relevant plaintext/ciphertext bits,  $C_{ij} = 1$ . If not,  $C_{ij} = -1$ .
- Where the “active” text bits are those which we partially encrypt/decrypt during the attack,  $x$  is a vector such that  $x_j$  is the number of (plaintext, ciphertext) pairs in which the  $k$ -bit number represented by these bits is  $j$ . Note that  $x$  is the vector we previously referred to as *COUNTERS\_1*; the computation of  $x$  is in fact the distillation phase and has complexity  $O(N)$  as previously stated.
- In the case of a 1R attack,  $T$  is calculated during the distillation phase and replaces  $x$  in the algorithm.  $T_j$  is defined as (the number of P/C-pairs such that the value of the active ciphertext bits is  $j$  and the parity of the active plaintext bits is 0) - (number of pairs such that the active ciphertext bits have value  $j$  and the active plaintext bits have parity 1.)

The matrix/vector product  $Cx$ , when all entries within are divided by 2, is the previously-defined vector  $v$  such that  $v_i$  is the sample bias for TPS candidate  $i$ . We do not need to carry out this division, as the values currently present ( $v_i =$  the number of pairs such that the approximation held for candidate  $i$ , minus the number such that it did not hold.) suffice equally well. For this reason, we will engage in a minor abuse of notation and refer to  $Cx$  as  $v$  from here on. Where key ranking is involved, this vector would need to be sorted (in  $O(k \cdot 2^k)$  time); otherwise it would need to be searched (in  $O(2^k)$  time) for the maximum absolute value therein.

To compute and store the entire matrix  $C$  would require  $O(2^{2k})$  time and memory, in addition to the  $O(2^{2k})$  time complexity of the multiplication  $Cx$ . However, by relying on various properties of  $C$ , and on the Fast Fourier Transform, we are able to derive the vector  $v = Cx$  using only one column of  $C$ . We can do this with time complexity  $O(2^k)$  to calculate the column of  $C$ ,  $O(3 \cdot k \cdot 2^k)$  to compute the transforms, and  $O(2^k)$  memory since only one column of  $C$  is needed for the new technique.

This is a significant improvement on the  $O(2^{2k})$  complexity of the original algorithm for this phase.

(The key property of the matrix  $C$  is that the value of  $C_{ij}$  is entirely dependent on  $(i \oplus j)$ . Any  $C_{ij}$  and  $C_{gh}$  such that  $(i \oplus j) = (g \oplus h)$  will have the same value. This means that the set of values in any one column of  $C$  is the same as the set of values in any other column - just in a different order. This redundancy is the key to the complexity improvements obtained. We do not have the space to provide a full explanation here, but refer the reader to Collard et al.’s paper [25] for the full explanation.)

Let us be a little more precise with regards to the memory requirements. The column of  $C$  has  $2^k$  entries, all -1 or 1. This implies that we need no more than  $2^k$  bytes to store it in signed char variables. Variable types using fewer bits are unlikely to be present on any compiler, or to have the same speed of implementation.

We also need to store  $x$ . This has  $2^k$  entries, each of which must be at least  $\log_2(N)$  bits in size. On a modern processor with 64-bit word size, most ciphers will require no more than  $2^{k+1}$  words here, or  $2^{k+4}$  bytes. We do not know of any block ciphers in widespread use with block size  $> 128$ , although pre-AES versions of Rijndael did support up to a 256-bit block.

During the calculation of  $Cx$ , two “interim” arrays,  $y$  and  $z$ , are used [25]. Based on Carlet’s description [12] of a version of the FFT using finite-field arithmetic over  $GF(2)^x$ , which is equivalent to both the Fast Walsh-Hadamard Transform and the  $k$ -dimensional FFT of size  $2^k$  [47], and on our own implementation of the same, we can say with confidence that the same data types can be used for these as for  $x$ , and hence that these arrays should require  $2^{k+5}$  bytes between them.

This gives us a memory complexity of  $2^k + 2^{k+4} + 2^{k+5} \approx 2^{k+5.615}$  bytes. One of the previous arrays can presumably be reused to store  $Cx$  itself - the space for the array that stored  $x$  could be repurposed for signed instead of unsigned data, for instance.

We now address the question of what the time complexity is in terms of. Clearly the naive algorithm would require  $2^{2k}$  partial encryption/decryptions (PEDs) to calculate  $C$ , in addition to  $O(2^{2k})$  arithmetic operations (AOs) and memory accesses (MAs) to calculate  $Cx$ . The new algorithm requires  $2^k$  PEDs to calculate the first column of  $C$ , followed by  $O(3 \cdot k \cdot 2^k)$  memory accesses and AOs to calculate  $Cx$ .

The question of how many arithmetic operations are involved in partially encrypting/decrypting a cipher varies by cipher, attack, and implementation. It is further complicated if lookup tables are used for the S-boxes and we are forced to evaluate the complexity of an encryption or decryption in terms of memory accesses as well, since the complexity of these will vary significantly by CPU. We will later on make use of approximate complexity in terms of arithmetic operations for reduced-round versions of Serpent using the optimised bitslice implementation [1, 2].

Based on the aforementioned version of the FFT [12], we estimate  $\approx (2k+3) \cdot 2^k$  MAs per transform. (This is only an estimate, since we do not have detailed knowledge of CPU register allocation.) Where  $y$  and  $z$  denote the output arrays from the first two transforms, the dot product  $y \cdot z$  must then be calculated, requiring  $3 \times 2^k$  memory accesses. Multiplying the per-transform complexity by three, and adding the complexity of the dot product and the  $2^k$  memory accesses when the first column of  $C$  was calculated and written into memory, gives us  $\approx (6k + 13) \cdot 2^k$  MAs in total. As for arithmetic operations, the calculation of the dot product requires  $2^k$  AOs, and based on the same evidence as before we estimate  $\approx (2k + 1) \cdot 2^k$  AOs per transform, giving us a total of  $\approx (6k + 4) \cdot 2^k$  AOs.

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.

We note that equating complexity in terms of memory accesses to complexity in terms of partial cipher encryptions is a difficult matter [33], 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 [6, 8, 7] 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 [65] 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  $1 \text{ MA} = 3/8$  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 [33], 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 [60] offers a potential performance improvement, trading 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  $2^k$  partial encryption/decryptions, the method need only execute  $2^{l_1}$  partial encryptions and  $2^{l_2}$  partial decryptions, in addition to est.  $(2^{l_2} \cdot (6l_1 + 4) \cdot 2^{l_1} + 2^{l_1} \cdot (6l_2 + 4) \cdot 2^{l_2}) = (6k + 8) \cdot 2^k$  arithmetic operations and est.  $(2^{l_2} \cdot (6l_1 + 13) \cdot 2^{l_1} + 2^{l_1} \cdot (6l_2 + 13) \cdot 2^{l_2}) = (6k + 26) \cdot 2^k$  memory accesses. 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.

A generalised version of this algorithm for use in multidimensional linear attacks was also developed [60], leading to the best cryptanalytic result so far against 12-round Serpent. Where  $m$  denotes 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 the experimental correlations to empirical probability distributions.

### 1.1.2 Generalising the new method to the Feistel structure - an example.

As stated, it is in some cases possible to generalise the improved analysis phase to ciphers other than substitution-permutation networks. The key fact upon which all of the new methodology relies is that  $C_{ij} = f(i \oplus j)$  for key value  $i$  and text value  $j$ . Let us note that in Matsui's attack on the full DES, twelve text bits are xored with twelve key bits prior to being input to  $f$ , but the thirteenth (an xor of eight text bits) is not. Let us therefore introduce a "dummy" key bit, the value of which we know to be zero (but which we will act as though we do not know), and assume that it is xored with one of these eight bits at the start or end of the cipher (One such bit, referred to in Matsui's notation as  $C_H[29]$ , is not in fact suitable for this; we must use one of the others.) This allows us to treat the partial encryption/decryption in Matsui's attack as  $f(i \oplus j)$ , to construct a column of  $C$  containing  $2^{13}$  entries, and indeed to carry out the rest of the attack with complexity as described above for  $k = 13$ . The fact that each row for (dummy bit = 1) will be equal to  $-1 \times$  (corresponding row for dummy bit = 0, all other key bits unchanged), and hence that the matrix will be rank-deficient, will not affect the attack.

Unfortunately, the analysis phase of Matsui's attack on the full DES has the least effect on the overall complexity, and it is this phase which would be optimised by applying the above. Furthermore, the fact that  $C_H[29]$  could not be xored with a dummy key bit (since it was already one of the bits xored with a real key bit) suggests that approximations for DES, and other non-SPN ciphers, may exist to which this method cannot be applied.

## 2 How nonlinear approximations affect the attack

### 2.1 How unbalanced nonlinear components in the approximation affect the attack.

Let us assume that we have set up a 2R linear or nonlinear approximation to the inner rounds of some cipher. (It will be straightforward to extrapolate the results of this subsection to the case of 1R attacks.) For a conventional linear approximation, this would be an equation of the form  $(x_{a_1} \oplus x_{a_2} \oplus \dots \oplus x_{a_s}) \oplus (y_{b_1} \oplus \dots \oplus y_{b_t}) = 0$ , where the  $x_a$  are the input bits to Round 2 and the  $y_b$  are the output bits of round  $r - 1$ .

Now, this equation, assuming the cipher has acted sufficiently well in randomising its outputs, should hold with bias 0 if the correct TPS has not been guessed.

We have (balanced function on one set of bits  $x$ )  $\oplus$  (balanced function on some other set of bits  $y$ ) = 0. Again, assuming the cipher's randomising effect has been adequate, the value of the first set

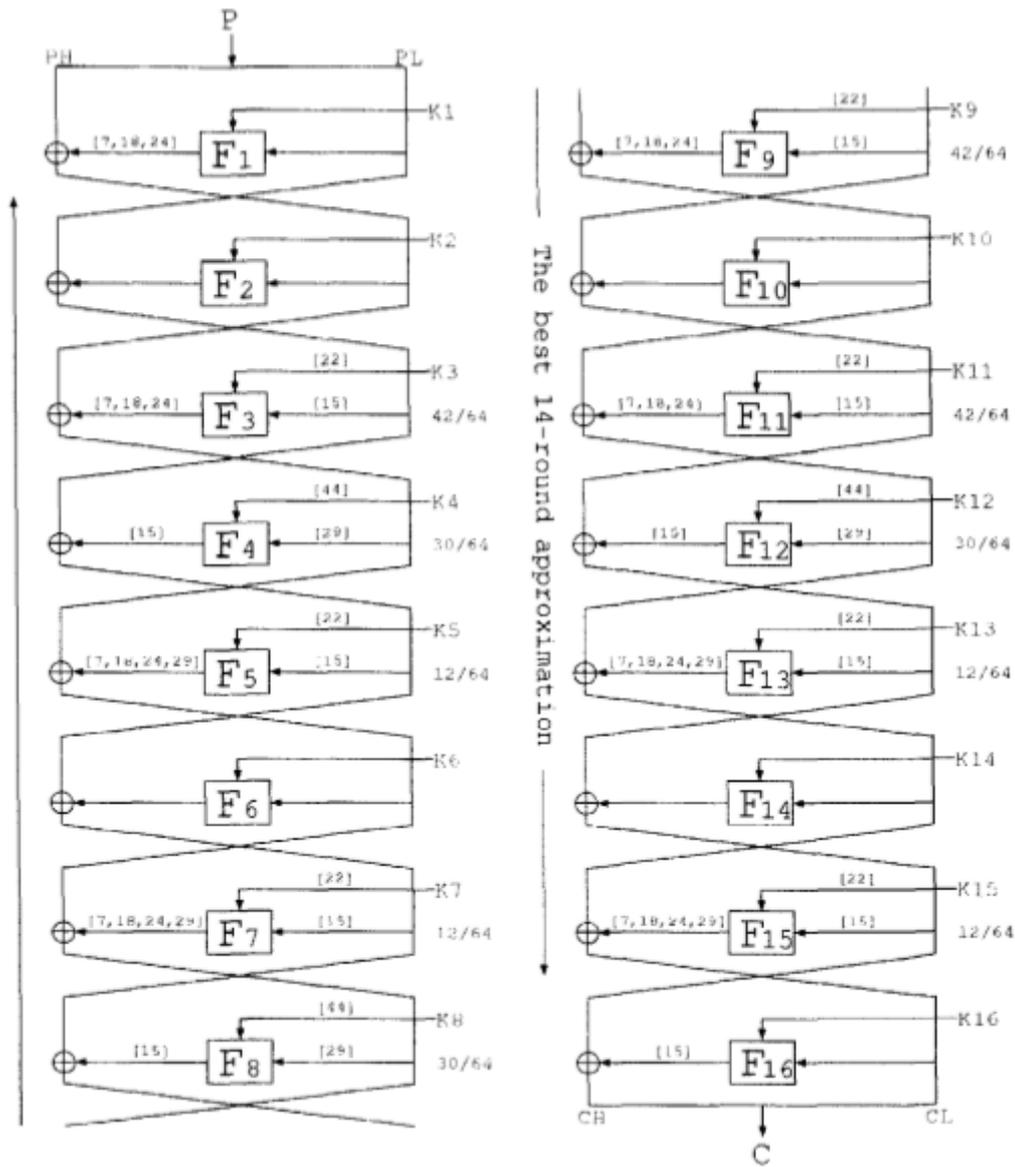


Figure 2: Diagram showing the full 16-round DES and the approximation for rounds 2 to 15 used by Matsui [53]

of bits should be viewed as independent of the second. Then  $P(\text{approximation} = 0) =$

$$\begin{aligned}
& P((x_{a_1} \oplus \dots \oplus x_{a_s} = 0) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 0)) \\
& + P((x_{a_1} \oplus \dots \oplus x_{a_s} = 1) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 1)) \\
& = (0.5 \times 0.5) + (0.5 \times 0.5) \\
& = 0.5
\end{aligned}$$

This only depends on the linear function on the approximation's input bits (the linear component at the input end) and the linear function on the approximation's output bits (the linear component at the output end) being balanced, not on their being linear. Either or both of these could be replaced with a balanced nonlinear function without affecting this.

Therefore, for the following configurations for the overall approximation, the attack works as predicted by the usual probability model:

1. First per-round approximation (Round 2 of the cipher) in overall approximation is linear. Final per-round approximation (to Round  $(r - 2)$ ) is also linear.
2. First per-round approximation in overall approximation is a nonlinear approximation with a balanced nonlinear component. Final per-round approximation is linear.
3. First per-round approximation in overall approximation is linear. Final per-round approximation is a nonlinear approximation with a balanced nonlinear component.
4. First per-round approximation in overall approximation is a nonlinear approximation with a balanced nonlinear component. Final per-round approximation is also a nonlinear approximation with a balanced nonlinear component.

Now, let us assume that either the first per-round approximation, or the final per-round approximation, is an unbalanced function, and that the other is balanced. Without loss of generality, we may assume that it is the first per-round approximation, the approximation to Round 2 of the cipher, that is balanced. Let  $P(\text{unbalanced component} = 0)$  be denoted  $\alpha$ .

Then, for an incorrect key,  $P(\text{approximation} = 0) =$

$$\begin{aligned}
& P((x_{a_1} \oplus \dots \oplus x_{a_s} = 0) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 0)) + P((x_{a_1} \oplus \dots \oplus x_{a_s} = 1) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 1)) \\
& = (0.5 \times \alpha) + (0.5 \times (1 - \alpha)) \\
& = (0.5 \times 1.0) \\
& = 0.5
\end{aligned}$$

We see that, as long as either the first or the last round of the approximation is a balanced function on the input bits to the inner rounds, or the output bits to said rounds, it does not matter whether the function acting on the bits at the other end is balanced. The question therefore arises: can we use approximations which are unbalanced at both ends?

Unfortunately, in general we cannot. Let  $\beta$  denote the probability that the nonlinear function at the input end is zero. Let  $\gamma$  be the probability that the nonlinear function at the output end equates to zero. Then  $P(\text{approximation} = 0) =$

$$\begin{aligned}
& P((x_{a_1} \oplus \dots \oplus x_{a_s} = 0) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 0)) + P((x_{a_1} \oplus \dots \oplus x_{a_s} = 1) \cap (y_{a_1} \oplus \dots \oplus y_{a_s} = 1)) \\
& = (\beta \times \gamma) + ((1 - \beta) \times (1 - \gamma))
\end{aligned}$$

This is not always equal to 0.5. For example, let  $\beta = 0.4$ ,  $\gamma = 0.6$ . Then the above is equal to 0.48, not 0.5. Let  $\beta = \gamma = 0.1$  and the probability equates to 0.82, diverging further from 0.5! Clearly, having an unbalanced function at both ends of the approximation is problematic, and it is for this reason that we limit ourselves to situations in which at least one end of the approximation is a balanced function on its respective set of bits.

The reader may, having noted that the related approximations previously referred to define several different functions at their respective ends of the approximation, be concerned that this will make it difficult to ensure that they are all balanced. Fortunately, all nonlinear components in a set of related approximations are balanced if and only if the primary approximation is balanced.

To prove this, let us assume without loss of generality that the correct key is an all-zeroes bitstring, and that the nonlinear component is in terms of the approximation's output bits. Consider that the nonlinear component of the  $\alpha_1\alpha_2\dots\alpha_l$ th related approximation,  $n_{\alpha_1\alpha_2\dots\alpha_l}$ , is equal to  $n_0((y_1 \oplus \alpha_1), (y_2 \oplus \alpha_2), \dots, (y_l \oplus \alpha_l))$ , where  $n_0$  is the nonlinear component of the correct, or primary, approximation. Clearly, each related approximation  $n_i$  must have a truth table which is a permutation of that of  $n_0$ , the permutation being determined by the fact that  $n_i(y) = n_0(y \oplus i)$ .

## 2.2 How the related approximations affect the attack.

We have already discussed the difficulty faced by the cryptanalyst in working out which of  $2^h$  functions on the partially-decrypted ciphertext bits (and partially-encrypted plaintext bits) is equivalent to the nonlinear function on the S-box output/input bits involved in the approximation. One possible approach would be to compute all possible functions, and for each guess at the key bits involved, accept the function with the highest probability bias as correct.

Knudsen and Robshaw [46] considered a very simple form of this, in which no partial decryption was involved. In effect, they carried out a "OR" attack in which the whole cipher (5-round DES) was nonlinearly approximated, using an approximation that was linear on the plaintext bits but nonlinear on the ciphertext bits. The nonlinear approximation to the final round had an absolute bias of 24, and the aim of the attack was to deduce the four key bits  $k_{\alpha_1} \dots k_{\alpha_4}$  which were xored with the final-round S-box input bits involved in the approximation.

The problem that occurred was that several of the "related" functions corresponded to alternative nonlinear approximations which also possessed high magnitude of bias. One of these possessed the same absolute bias as the original, and for those which did not, it was not clear how much data would be required to distinguish, say, the correct function and a bias 24 approximation from an incorrect function which defined a bias 16 approximation. Or, in some of the situations we encountered when devising our own approximations, a bias 24 approximation and an incorrect function defining an approximation with bias  $-22$ .

Let us try to demonstrate, using examples, why this problem does not apply in the case of linear cryptanalysis, and why attacks based on nonlinear cryptanalysis cannot disregard it in the same way as conventional linear attacks.

In a conventional 2R linear attack, key bits are guessed for S-boxes in the first and last rounds of the cipher. It is not necessary to guess the key bits affecting the S-boxes in the first and last rounds of the *approximation*. Any guess, right or wrong, at these bits simply xors a linear function with a constant value. Some key guesses will, in effect, always xor the correct function calculation with zero and leave it unaffected. Others, by always xoring with 1, will merely flip the sign of the bias.

In the context of Matsui's attack on 16-round DES [53], this means that although the first and last rounds of the approximation are, respectively, 2 and 14, it is not necessary to guess the round key bits which are xored with S-box input values in these rounds. Only round key bits from rounds 1 and 16 are needed.

For a nonlinear attack, this is not the case. If either the first or last round of the approximation involves a nonlinear component, and if the bits involved in said component are xored with key bits after leaving/before entering the active S-box, these key bits have to be guessed.

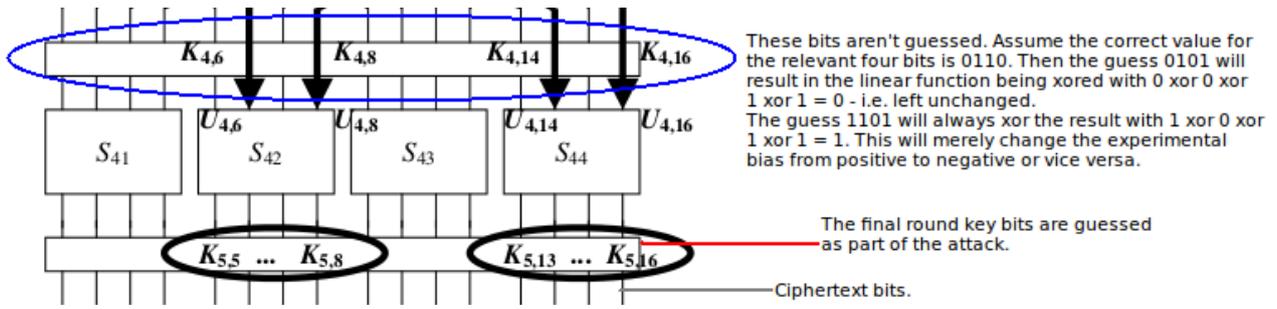


Figure 3: Diagram showing the final round and key xors of the Heys toy cipher during a conventional linear attack.

Let  $x_i$  denote the  $i$ th input bit to whichever S-box we are dealing with, and let  $y_j$  be the  $j$ th output bit. Let us compare the linear approximation  $x_4 \oplus x_5 = y_3 \oplus y_4$  to DES S5 with the nonlinear approximations

- $x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_1 y_3$
- $x_3 \oplus x_4 = 1 \oplus y_1 \oplus y_3 \oplus y_4 y_1 \oplus y_3 y_4 \oplus y_3 y_4 y_1$

to Serpent S3:

Related approximation	Linear function	Bias
0	$x_4 \oplus x_5 = y_3 \oplus y_4$	+6
1	$(x_4 \oplus 1) \oplus x_5 = y_3 \oplus y_4$	-6
2	$x_4 \oplus (x_5 \oplus 1) = y_3 \oplus y_4$	-6
3	$(x_4 \oplus 1) \oplus (x_5 \oplus 1) = y_3 \oplus y_4$	+6

Table 1: Linear approximation to DES S5. Note that all relateds are either the original approximation or  $1 \oplus$  it.

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

Table 2: Nonlinear approximation to Serpent S3. In this table, the polynomial forms of the related approximations are not expanded.

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

Table 3: Another nonlinear approximation to Serpent S3. In this table, the polynomial forms of the related approximations *are* expanded.

For the first nonlinear approximation, in a situation where  $y_1 y_3 = 1$ , any wrong guess at key bits  $(k_1, k_3)$  will result in its value being wrongly calculated as 0. If  $y_1 y_3 = 0$ , by contrast, only one of the three possible wrong guesses for  $(k_1, k_3)$  will result in its value being calculated incorrectly. 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 overall magnitude of the bias involved in the attack invariant.

It is therefore necessary to guess at 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 or reduced-round variant thereof (in a 2R-attack.), simply to be able to obtain the latter set of key bits. Since having to guess the values of these bits adds to the time complexity of the attack, we would like to obtain some information about them.

Let us look again at the approximation  $x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_1 y_3$  above. The related approximations when  $k_4$  is guessed wrongly hold with the same absolute bias as the corresponding relateds for when it is not, so unfortunately we cannot recover any information about the value of  $k_4$ . However, the related for  $(k_4$  alone wrong) is the only related with an absolute bias near to that of the correct guess, so we should be able to recover the values of bits  $k_1$  and  $k_3$ .

Now consider  $x_3 \oplus x_4 = 1 \oplus y_4 \oplus y_3 \oplus y_1 y_4 \oplus y_1 y_3 \oplus y_1 y_3 y_4$ . As seen in the table above, no related approximation has as high a bias as the correct one, so in theory it should be possible to obtain information on all three key bits involved. In practice, since the relateds for  $(k_4$  wrong) and (all three key bits wrong) both have high bias, the amount of data required to distinguish these from the correct related will be higher than that for the remainder of the attack.

For this reason, in a search for approximations to use in a straightforward generalisation of the linear attack, it would seem that the cost function should try to maximise the difference between the absolute bias of the evolved approximation, and the highest absolute bias of any of the related approximations. Since the attacker needs to obtain the key bits for the first and last rounds of the cipher, the need for the “primary” approximation to possess a high absolute bias is also important.

The above was all taken into account by Knudsen and Robshaw. However, what was not observed was that related approximations with high absolute bias may actually benefit the cryptanalyst during the search for the key bits in the cipher’s outer rounds. If the correct key is guessed in these rounds, the related approximations will be expected to hold with their predicted biases; if not they will be expected to hold with bias 0. By evaluating all possible related approximations, the cryptanalyst can track the information on the biases of  $2^l$  approximations instead of just one, and may be able to use this extra information to boost the “signal-to-noise ratio” and reduce the data requirements of the basic attack - in effect trading increased time complexity against reduced data complexity.

Moreover, it may be that the cryptanalyst will decide only to attack the key bits in the outer rounds, basing the score for each outer-round key candidate on the best experimentally obtained bias across all of the relateds. If the primary approximation is expected to hold with a particularly high magnitude of bias, the reduced data complexity resulting from this approach may be deemed a reasonable tradeoff for the increased time complexity (compared to conventional linear) in evaluating

the full set of related approximations.

### 3 New statistical frameworks and cryptanalytic techniques.

#### 3.1 Adapting the new analysis phase to nonlinear cryptanalysis of substitution-permutation networks.

Where the cipher being attacked is a substitution permutation network, we will describe an adaptation of Collard et al.'s new analysis method [25], as also the improvements due to Nguyen et al. [60] to nonlinear cryptanalysis. For other cipher structures, such as Feistel ciphers, the intention is to adapt this method as far as possible - indeed, in the next subsection we will discuss adapting this method to the Data Encryption Standard.

- Let  $k$  denote the target partial subkey; i.e. the set of attacked key bits. Let  $k_1$  be the set consisting of the bits of  $k$  interacting with the S-boxes in the outer rounds of the cipher (the ones which we must partially encrypt/decrypt.) Let  $k_2$  be the set of bits of  $k$  interacting with the S-boxes in the outer rounds of the approximation.
- Let  $f(i, j)$ , where  $i$  is the value of the active text bits, and  $j$  the value of the bits of  $k_1$  with which they are xored, be a  $2^{|k_2|}$ -long string of values  $\in \{-1, 1\}$  defined as follows:
  1. Partially encrypt/decrypt  $i$  using  $j$ . This will yield a string of text bits entering/leaving the outer rounds of the approximation,  $|k_2|$  of which are involved in the nonlinear component. Note that this string of text bits is in fact only dependent on the value  $(i \oplus j)$ .
  2. For each possible value  $\mu$  of  $k_2$ , xor the  $|k_2|$  bits mentioned above with the appropriate bits of  $\mu$ , and compute the nonlinear function on these. Set the  $\mu$ th entry in the string of values to  $-1$  if the nonlinear approximation does not hold when this is done. Otherwise, set it to 1.
- The string of 1s and -1s is obtained by applying a sequence of functions to a set of bits determined entirely by the value of  $(i \oplus j)$ . This allows the matrix  $C$  such that  $C_{ij} = f(i \oplus j)$  to be defined as before, except that  $C_{ij}$  is now a string of values instead of just one.
- Where  $x$  is the vector containing the frequency with which each value for the involved text bits has occurred,  $Cx$  can also be calculated as before, although each entry in  $Cx$  is now a  $2^{|k_2|}$ -string of integers.
 

(To clarify, let us assume that we have  $2^{|k_2|}$  matrices  $C(y)$ , defined by letting  $C(y)_{i,j}$  be equal to the  $y$ th entry in  $C_{i,j}$ . We can calculate  $C(y)x$  for each  $C(y)$ , and then  $Cx$  is the vector such that  $Cx_i$  is the string of  $i$ th entries from each of the  $C(y)x$  in order:  $(C(1)x_i, C(2)x_i, \dots, C(2^{|k_2|})x_i)$ .)
- So far, the memory complexity, and the time complexities in terms of arithmetic operations and memory accesses, of the corresponding stages of the linear version of this method can simply be multiplied by  $2^{|k_2|}$  to obtain the complexity of the new method up to this point.
- The first problem we are faced with is choosing the correct value of  $j$  (i.e. of  $k_1$ ) from this. Each string of values needs to be assigned a score such that, according to some statistical theory, the more likely a given  $k_1$  candidate is to be correct, the higher the score assigned to its corresponding string of values.

In conventional linear cryptanalysis using the analysis method of Collard et al., there would be only one value in this string, the absolute value of which would be the score. The complexity of going through the values in  $Cx$  and setting them to their absolute values would be at most  $2^{|k_1|+1}$  memory accesses ( $2^{|k_1|}$  reads, and at most  $2^{|k_1|}$  writes.) and  $2^{|k_1|}$  arithmetic operations. More generally, the time complexity for this phase for nonlinear cryptanalysis is at least  $O(2^{|k_1|} + 2^{|k_1|+|k_2|})$  memory accesses, to access all values in all strings and to write the scores to an array.

- One way in which we could handle this would be by allocating each string of values a score equal to the maximum absolute value therein. This approach, which we shall refer to as the *maximum-bias* approach, is the simplest possible method, and is probably the best to use when one of the approximations has a bias of considerably higher magnitude than any of the other relateds. However, it does fail to make use of most of the information in each vector  $Cx_i$ .

The vector of scores should need at most (block size of cipher) bits per entry. Currently most block ciphers have block size  $\leq 128$ , so this usually adds  $\leq 16 \times 2^{|k_1|}$  bytes to the memory complexity. The time complexity will be dominated by the  $O(2^{|k_1|} + 2^{|k_1|+|k_2|})$  memory accesses.

- Another possible approach would be to allocate each string of values a score equal to the sum of squares of the values therein, before either accepting the value of  $k_1$  with the highest score or key ranking according to this score.

The time complexity for scoring according to this method should not differ substantially from the maximum-bias method, but the memory required for the vector of scores would be substantially higher -  $2^{|k_1|+|k_2|} \times 2 \lceil \log_2(N) \rceil$  bits,  $\leq 2^{|k_1|+|k_2|} * (BLOCK\_SIZE * 2)$ , since in theory an attack using the full codebook could result in at least one score equal to  $2^{2 \times BLOCK\_SIZE}$ . For a 128-bit block cipher, this leads to an upper bound of  $\leq 2^{|k_1|+|k_2|} * 32$  bytes.

If the truth tables of the related approximations are statistically independent, this will allow us to make use of the  $\chi^2$ -statistic in a way similar (but not identical) to its use in multidimensional linear cryptanalysis [17, 13].

If they are not, we still gain information from the sum of squares that allows them to be used as a distinguisher, but we do not gain as much as if they were independent. Since there is no known statistical framework for a variation of the  $\chi^2$ -statistic where some of the Pearson correlation coefficients of the variables are not  $\in \{0, \pm 1\}$  (i.e. where they are not independent), we will need to conduct experiments on significantly reduced-round cipher variants to obtain empirical evidence for the distinguishing advantage obtainable.

- The question arises as to whether we could exploit our knowledge of the theoretically predicted distribution of the biases of the nonlinear approximation and its relateds. For example, if we expect the related approximation for  $k_2 \oplus \alpha$  for some value  $\alpha$  to hold with bias  $\beta$ , and the related approximation for  $k_2 \oplus \gamma \oplus \alpha$  (for some value  $\gamma$ ) to hold with bias  $-\beta$ , the above approaches do not currently utilise this knowledge. However, since we do not know in advance the correct value of  $k_2$ , this would require us, for each  $k_1$ , to attempt to match the distribution to every possible value for  $k_2$  - which would result in increased time complexity.

Since the log-likelihood ratio has the optimal data complexity among all methods for distinguishing a distribution  $p$  from another distribution  $q$  [30, 17], we believe that this would be the most effective means to exploit the information referred to. Since the related approximations may not have statistically independent truth tables, though, similar problems to those described for the  $\chi^2$ -statistic may still arise.

There is also the “linear hull” effect to be borne in mind. Approximations with the same input and output bitmasks, but following different paths through the cipher - i.e. different characteristics - may result in the actual distribution being different to that predicted theoretically. Figure 4 of a recent paper by Collard and Standaert [22] shows the results of experiments conducted on reduced-round versions of the cipher SmallPRESENT [50], in which this difference is seen to increase significantly with the number of rounds. In all of these experiments, we note that as the number of rounds increases, the magnitude of the theoretical bias for a conventional linear approximation (as calculated using the Piling-Up Lemma) is seen to increasingly underestimate the magnitude of the actual bias. Furthermore, the extent of this underestimate varies significantly depending on the key value, although the extent of this variation does not appear to increase further after five rounds.

For the other methods we have suggested, this would not pose a problem; indeed it would be beneficial to the attack's performance. However, for a particular distance metric (the Kullback-Leibler distance), the LLR statistic in a linear cryptanalysis variant would reward high distance from the uniform distribution, and low distance from the theoretical distribution, equally. The linear hull effect would clearly interfere with the second part of this.

Furthermore, experiments conducted on SmallPRESENT are particularly relevant to PRESENT and Serpent - in fact, SmallPRESENT is parameterisable such that for one particular parameter value, it is exactly the same as PRESENT! All three ciphers have the following in common:

- An SPN structure, so that round-key xor, application of a layer of substitution boxes to the entire block, and then a linear diffusion layer, are applied in sequence in each round (Serpent omits the diffusion layer in the final round), followed by a final key xor at the end of the cipher.
- All the S-boxes in a given round are identical  $4 \times 4$  bijections, with differential uniformity 4, nonlinearity 4, and most/all of the S-box co-ordinate functions having algebraic degree 3. In particular, the S-box used in PRESENT and SmallPRESENT is affine-equivalent to Serpent's S2 and S6.

Serpent does have a more effective diffusion layer than the permutation used by PRESENT and SmallPRESENT, but whether the increased number of active S-boxes resulting from this exacerbates the problem observed by Collard and Standaert or not is unclear - it seems extremely unlikely that it could in any way mitigate it.

Where the theoretical prediction is known to be accurate, or where experiments have indicated that it is likely to be for the particular cipher and number of rounds being attacked, the log-likelihood ratio (LLR) has been shown in the context of multidimensional linear cryptanalysis [17] to be superior to the  $\chi^2$  statistic. Approximations to the LLR statistic also exist which can be computed much more quickly - one based on its Taylor series expansion [30], another, slightly less accurate but faster to compute, based on the convolution of probability distributions [37, 38]. In experiments, the average difference in advantage between the LLR and whichever of its approximations we are testing has been negligible and has decreased to 0 as  $N$  has increased.

- Upon accepting a given value of  $k_1$ , we next need to find  $k_2$ . Depending on the various parameters of the attack, there may be situations where the most practical approach is simply to include the bits of  $k_2$  in the exhaustive search for the non-attacked key bits. For example, it may be that the incorrect key guesses result in related approximations with too high a bias to be practically distinguishable from the correct  $k_2$  and corresponding approximation.

As an example, let us consider an attack on Serpent in which only the final round of the approximation contains a nonlinear component; this being in S-box S3 with input bitmask 11 (so  $x_1 \oplus x_3 \oplus x_4 =$  some nonlinear function of the output bits with some bias  $\epsilon$ ). We have eight nonlinear approximations to this bitmask with bias 6, four of which are of particular interest here. Each of these four has one related approximation with a bias of  $-6$ , one related approximation with bias 2, and one related approximation with bias  $-2$ .

(Approximations with these biases occur frequently for  $4 \times 4$  S-boxes. They are especially useful for various reasons:

- Both of the related approximations with absolute bias 2 are statistically independent of the approximation with bias 6.
- The approximation with bias -6 has a truth table which can be obtained from the truth table of the bias 6 approximation by flipping all of the bits therein. This means that it provides no information that the approximation with bias 6 does not, and can safely be omitted from the attack.

- The approximation with bias 2 is related to the approximation with bias -2 in the same way. This means that only one of them provides useful information, and the other can be omitted from the attack. It is up to the cryptanalyst to decide which one.
- Their nonlinear components are balanced.

We are therefore able to handle statistical dependence among the related approximations in an extremely straightforward fashion, leaving us with a set of completely independent approximations for which the  $\chi^2$  statistic is fully valid, and for which the LLR statistic is also valid (barring issues resulting from the linear hull effect).

We can use any of the approximations individually, or we can attempt a form of multiple nonlinear cryptanalysis using two or more approximations (or, equivalently, two or more sets of related approximations) simultaneously. The below pseudocode demonstrates the attack for both the one-approximation and two-approximation cases.

---

**Algorithm 1** Nonlinear cryptanalysis algorithm

---

```

l ← the number of active data bits.
h ← the length of k2.
for (i ⊕ j) ← 0, 2l − 1 do
  Partially encrypt/decrypt (i ⊕ j).
  Let m denote the result of this.
  Let  $\mu$  denote the bits of m involved in the nonlinear component(s).
  for CURRENT_K2_VAL ← 0, 2h − 1 do
     $\delta$  ←  $\mu$  ⊕ CURRENT_K2_VAL
    if Attack uses one approximation then
      Compute nonlinear function on  $\delta$ 
      if Approximation holds then
        Cij[CURRENT_K2_VAL] ← 1
      else
        Cij[CURRENT_K2_VAL] ← −1
      end if
    else if Attack uses multiple approximations then
      for CURRENT_APPROX ← 0, NO_OF_APPROXIMATIONS do
        Compute current nonlinear function on  $\delta$ 
        if Current approximation holds then
          Cij[CURRENT_APPROX][CURRENT_K2_VAL] ← 1
        else
          Cij[CURRENT_APPROX][CURRENT_K2_VAL] ← −1
        end if
      end for
    end if
  end for
end for
Compute Cx.

```

We obtain, for each value of *k*<sub>1</sub>, a vector of values.

We allocate a score to this vector depending on the statistical method in use.

---

For each value of *k*<sub>1</sub>, based on whichever statistical method is in use (whether maximum-bias,  $\chi^2$  or other) we assign a score to the distribution of values in its corresponding *Cx* entry. For the maximum-bias and  $\chi^2$  methods, the scoring system should reward high values for the distance between the experimentally obtained distribution and the uniform distribution. A randomly-chosen wrong key is expected to possess much lower distance than the correct key; however (as

noted in subsection 1.1) this does not necessarily mean that the correct key will possess the highest distance, and some form of key-ranking may be required.

If the LLR method is being used, we assign a score that rewards high LLR values. It is known [30] that the LLR of the empirical distribution  $\hat{q}$  derived from the experimental data, theoretical distribution  $p$ , and uniform distribution  $q$  (denoted  $LLR(\hat{q}, p, q)$ ) is equal to the Kullback-Leibler distance between the empirical and uniform distributions,  $D(\hat{q}||q)$ , minus the distance  $D(\hat{q}||p)$  between the empirical and theoretical distributions. Hence, such a scoring system rewards both distance from the uniform distribution and closeness to the theoretical distribution.

**Definition 3.1.** Let  $p, q$  be two probability distributions for discrete random variables, each of which has a set of  $M + 1$  possible values:

$$p = (p_0, \dots, p_M)$$

where  $p_i$  denotes  $Pr(\text{random variable with distribution } p \text{ takes the value } i)$ . Similarly:

$$q = (q_0, \dots, q_M).$$

The *Kullback-Leibler distance*, also known as the Kullback-Leibler divergence or the K-L distance, between  $p$  and  $q$  is:

$$D(p||q) = \sum_{i=0}^M p_i \log_2 \left( \frac{p_i}{q_i} \right)$$

The following lemmas [30, 17] are useful in efficiently implementing algorithms to compute the approximate K-L distance:

**Lemma 3.2.** *The first term in the Taylor series expansion of  $D(p||q)$  is:*

$$\sum_{i=0}^M \frac{(p_i - q_i)^2}{2q_i}$$

**Lemma 3.3.** *If,  $\forall(0 \leq i \leq M)$ ,  $|p_i - q_i| \ll q_i$ , then  $D(p||q)$  can be approximated by the first term in its Taylor series (as given above).*

**Definition 3.4.** Where the probability distributions  $p$  and  $q$  are defined as before, let  $\hat{q}$  denote an empirical distribution derived from experimental data. Let  $Q$  denote the true probability distribution of the data from which the  $N$  samples used to derive  $\hat{q}$  were taken.

Assume that we face a decision problem in choosing between the hypotheses  $H_0 : Q = p$  and  $H_1 : Q = q$ , and know one of these to be true. For a given test  $T$ , let  $\alpha_T$  denote the probability of  $H_0$  being accepted when  $H_1$  is correct, and let  $\beta_T$  denote the probability of  $H_1$  being accepted when  $H_0$  is correct. The Neyman-Pearson lemma [30] states that the *log-likelihood ratio* statistic is the optimal statistic for distinguishing between the two distributions, in that any test  $S$  using a different statistic in which  $\alpha_S \leq \alpha_T$  must have  $\beta_S \geq \beta_T$ .

The log-likelihood ratio (LLR) is defined thus:

$$LLR(\hat{q}, p, q) = \sum_{i=0}^M N \hat{q}_i \log_2 \left( \frac{p_i}{q_i} \right)$$

The higher the value of  $LLR(\hat{q}, p, q)$ , the more likely it is that distribution  $p$  is the correct choice.

**Lemma 3.5.** *The log-likelihood ratio and Kullback-Leibler distance are related thus:*

$$LLR(\hat{q}, p, q) = N \times D(\hat{q}||q) - N \times D(\hat{q}||p)$$



- $k_1^0$  denotes the correct value of  $k_1$ .
- $k_2^0$  denotes the correct value of  $k_2$ .
- $N$  denotes the number of known plaintext/ciphertext pairs involved in the attack.
- In nonlinear cryptanalysis,  $M$  denotes the number of related approximations. In the case of multidimensional linear cryptanalysis,  $M = 2^m - 1$  is the number of linear approximations involved in the attack; these being the nonzero linear combinations of the  $m$  base approximations.
- $P_S$  is the success probability of the attack.
- $C(p)$  is the theoretical capacity of the set of approximations used. Since  $p$  is clearly the theoretical distribution from the context in which this is referred to, we will sometimes simply denote it  $C$ .

### 3.3 Theoretical complexity with the chi-squared statistic.

Consider the complexity calculations for the  $\chi^2$ -statistic in multidimensional linear cryptanalysis [17]. Prior to Theorem 1, the authors state that  $\Phi(-b) \approx \frac{e^{-b^2/2}}{\sqrt{2\pi}}$  when  $b$  is large. This is not in fact the case - the approximation was taken from Section 4 of an earlier paper [4], but the authors of this had realised that the approximation was erroneous and published a correction [54]. The correct approximation for large  $b$  is  $\frac{e^{-b^2/2}}{b\sqrt{2\pi}}$ .

Theorem 1 of this paper relies on rearranging

$$N \approx \frac{2\sqrt{M}b + 4\Phi^{-2}(2P_S - 1)}{C(p)}$$

to obtain

$$b^2 \approx \frac{(NC(p) - 4\Phi^{-2}(2P_S - 1))^2}{4M} \quad (3.1)$$

The authors, relying upon the formula  $2^{-a} = \Phi(-b)$ , applied the approximation  $2^{-a} = \Phi(-b) \approx \frac{e^{-b^2/2}}{\sqrt{2\pi}}$ , claiming from this that  $a \approx b^2$  and so that the above equation gave an approximate formula for the advantage of the attack in terms of  $N$ . Since  $\sqrt{e} = 1.648721271 \neq 2$ , this is already incorrect. Allowing for the correction to the approximation,  $a \approx \frac{b^2}{2 \ln(2)} + \log_2(b) + \log_2(\sqrt{2\pi}) \approx 0.72b^2 + \log_2(b) + \log_2(\sqrt{2\pi}) \approx 0.72b^2 + \log_2(b) + 1.325$ .

The value  $b$  is not so large as to allow us to simplify further with  $a \approx 0.72b^2 + \log_2(\sqrt{2\pi})$ ; this would result in a 2-bit underestimate for the advantage in Cho's attack on PRESENT [14].

We note, however, that for relatively marginal attacks with low advantage, the condition of "large  $b$ " is not satisfied. Later on in this section, we will plot graphs of the estimated advantage based on this approximation, and on the direct computation of  $a = -\log_2(1 - \Phi(b))$ , and show that the approximation mistakenly predicts that low advantage attacks with data complexity below  $2^{27}$  on reduced-round Serpent cannot be mounted. We recommend calculating  $a$  from  $b$  directly if possible.

However, the problems run deeper still. Despite contacting the authors of the key paper [17], we have not been able to re-derive the approximation

$$NC(p) \approx 2\sqrt{M}b + 4\Phi^{-2}(2P_S - 1)$$

One of the authors, taking into account the incorrect  $\Phi(-b)$  approximation, has stated that there may have been a mistake, but no longer has access to the software originally used in obtaining the approximation. In particular, we believe that there is no way to obtain an approximation containing  $\Phi^{-2}(2P_S - 1)$ , and conjecture that this results from a misunderstanding of the formula  $(2P_S - 1) = \text{erf}(\frac{\Phi^{-1}(P_S)}{\sqrt{2}})$ .

Equation 9 of the same paper is:

$$\Phi^{-1}(P_s) = \left( \frac{\mu_R - \mu_a}{\sqrt{\sigma_R^2 + \sigma_a^2}} \right)$$

in which:

$$\sigma_a^2 = \frac{2M}{2^{n+a}\phi(b)^2}$$

In solving Equation 9 to obtain a formula for  $NC(p)$ , the approximation  $\sigma_a^2 \approx 0$  was originally made [17], and a quadratic equation with  $NC(p)$  as the variable is formed. In a later revision [61], the approximation  $\sigma_a^2 < M$  is used; and the formula for  $NC$  uses  $\sigma_a^2 = M$  to provide a conservative value for  $NC$ .

As  $n$  denotes the number of key bits targeted in the attack, we argue that  $n \geq 3$  (this value could in theory result during a 1R linear attack on a cipher using the CTC2 S-box [28]. A block cipher's S-boxes cannot have less than 3 input bits if they are to be nonlinear balanced functions. For an SPN the number of output bits would also have to be 3 and for a Feistel cipher it would only be the number of S-box input bits that was relevant), hence  $(n+a) \geq (3+a)$ . Moreover, it is clear that  $(n+a) \geq 2a$ . Exploiting these facts, it is possible to verify that  $M$  does represent an upper bound for advantage  $\geq 1$ . (Using the computer algebra package Mathematica, we draw graphs of  $\frac{2M}{2^{3+a}\phi(b)^2}$  and  $\frac{2M}{2^{2a}\phi(b)^2}$  to confirm this.) Using information from the graphs, we were able to obtain a tighter upper-bound of  $0.7854M$ .

Note that depending on how close the advantage is to  $n$ , the significance of the overestimate varies substantially. For example,  $\sigma_a^2 = \frac{M}{2^{5.2}}$  is attained for  $a = n = 56$ . For a smaller advantage of 32 and  $n = 56$ ,  $\sigma_a^2 \leq \frac{M}{2^{28.35}} \ll M$ . However, if  $M=2^{56} - 1$  (as used in various attacks on reduced-round Serpent [60]), this is not  $\approx 0$ .

(Note that said attacks must in fact have used the log-likelihood ratio, not the  $\chi^2$ -statistic, to succeed for so high a value of  $M$ .)

Replacing  $\sigma_a^2$  with  $0.7854M$  in the aforementioned quadratic equation, we use the quadratic formula to solve the equation in  $NC$  and obtain:

$$NC(p) \approx 2\Phi^{-2}(P_s) + \sqrt{2Mb} \pm \sqrt{\Phi^{-2}(P_s)(4\Phi^{-2}(P_s) + 4\sqrt{2Mb} + 2.7854M)} \quad (3.2)$$

We note that the expression under the square root sign cannot take on a negative value for  $P_s > 0.5$  as long as the advantage  $a$  is greater than or equal to 1 bit - and depending on  $P_s$  may still not be negative even for extraordinarily marginal attacks with lower  $a$ . So we are able to accept that the roots of this equation will be non-complex in real-world attack situations.

The question arises as to whether the larger or the smaller of the two roots should be considered the solution. For an attack obtaining a 4-bit advantage and probability of success  $\geq 0.95$ , the smaller root is a negative value, strongly indicating that it cannot be the correct solution. Furthermore, in email correspondence, we obtained from Nyberg [61] a pessimistic formula for  $NC$  relying on certain assumptions. If we consider situations in which these assumptions hold, we find that the smaller root diverges massively from the value given by this formula, while the larger root does not differ to such an extent. Based on this, we conclude that the smaller root does not match the true complexity of the attack and that the larger root is the correct value of  $NC$ :<sup>1</sup>

$$NC(p) \approx 2\Phi^{-2}(P_s) + \sqrt{2Mb} + \sqrt{\Phi^{-2}(P_s)(4\Phi^{-2}(P_s) + 4\sqrt{2Mb} + 2.7854M)} \quad (3.3)$$

---

<sup>1</sup>To avoid the use of computer algebra packages in deriving the above formula, it is stated [61] that  $NC < \frac{M}{4}$  and that this should be substituted for  $NC$  in the denominator, resulting in a pessimistic estimate for  $NC$ . Although this seems to have been the case for all multidimensional linear attacks so far,  $M$  is often much smaller in nonlinear attacks, and certainly exceeded  $4NC$  in the attack on DES below, so we were unable to make this substitution.

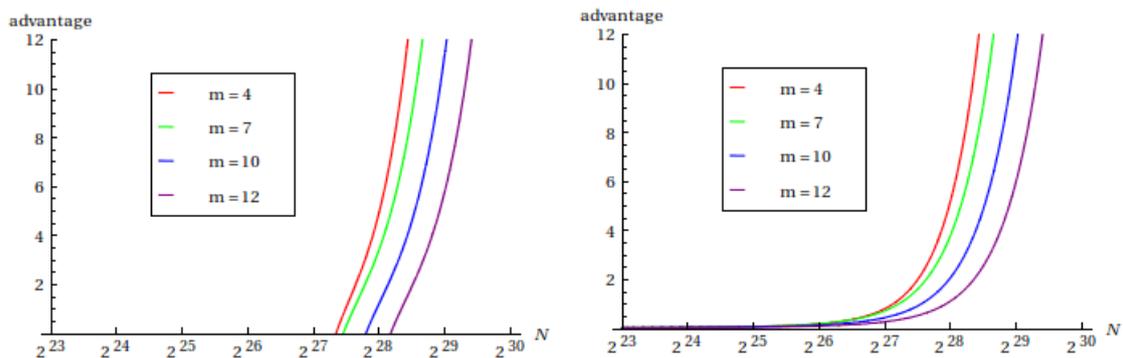


Figure 4: The graph on the left shows the result of using the approximation to calculate the advantage from  $b$ . We see that the approximation fails for marginal, low-advantage attacks. The graph on the right shows the result of computing advantage directly.

Using this new model, we find  $b$  by using Mathematica to solve the equation above, after which we can either compute the advantage directly from  $b$  or use the “large  $b$ ” approximation referred to earlier. For the 4, 7, 10 and 12-dimensional  $\chi^2$  attacks on 5-round Serpent in Hermelin et al.’s FSE 2009 paper [17], we use the corresponding capacities to plot new graphs of  $N$  against advantage (Figure 4).

Now, the y axes of these graphs show the advantage going up to 12, since this was the number  $n$  of bits in the TPS. However, if we do not use the graph plotting software to impose these restrictions, the graphs show the advantage continuing to increase indefinitely with an increasingly steep gradient, even though it should be upper-bounded by  $n$ . Furthermore, equation 3.3 does not have  $n$  as a variable after the pessimistic approximation for  $\sigma_a$  is introduced, suggesting underlying flaws in Selçuk’s statistical model for conventional linear cryptanalysis [66] that may have been exacerbated in the generalisation to multiple dimensions.

In Selçuk’s model [66], for all advantages except 0 and maximum advantage  $a = n$ , the  $r$ th-highest bias of any wrong key candidate ( $r = 2^{n-a}$ ) is assumed to have an asymptotic normal distribution. Let  $T_1$  be the lowest bias of any wrong key,  $T_2$  the second lowest,  $\dots$ ,  $T_{2^n-1}$  the highest (so that  $T_{2^n-r}$  corresponds to  $r$ ). A value  $q \approx (1 - 2^{-a})$  ( $0 < q < 1$ ) is defined, such that  $(2^n - r) = \lfloor q(2^n - 1) \rfloor + 1$ . Since there are  $2^n - 1$  wrong key candidates, we can obtain a tighter upper bound for  $q$  of  $q \leq \frac{2^n-1}{2^n-2}$ . As this would correspond to an attack with maximum advantage, for which the precise distribution of the highest bias of the wrong key candidate is known (assuming the Wrong-Key Randomization hypothesis), a formula based on this and not the asymptotic Normal approximation is used to calculate the attack’s complexity.

(A useful topic for future research would be a generalisation of this formula to the multidimensional case, so that the effect of varying the number of dimensions on the accuracy of the Normal approximations can be investigated.)

We can therefore say, when working with the non-extreme-value asymptotic Normal distribution, that  $q < \frac{2^n-1}{2^n-2}$ , and it also seems reasonable to treat the current statistical model as suspect for advantage higher than  $(n - 1)$ .

However, there is also reason to believe that the model may not be valid for some smaller advantages  $(n - x)$  either. In the textbook “Order Statistics” [32], discussing order statistic  $X_r$  (where the order statistics are  $X_1 \leq X_2 \leq \dots X_n$ ), David states (at the start of Section 9.1):

“If  $r/n \rightarrow \lambda$  as  $n \rightarrow \infty$ , fundamentally different results are obtained according as  $0 < \lambda < 1$  or  $\lambda = 0$  or 1, with  $r$  or  $(n - r)$  fixed.

“In the former case,  $X_r$  is a sample quantile and (subject to mild regularity conditions) has an asymptotic normal distribution. The latter case includes the extremes  $X_1$ ,  $X_n$  and corresponds to the  $m$ th extremes  $X_m$ ,  $X_{n-m+1}$  with  $m$  fixed. These have non-normal limiting distributions. Such a dichotomy into ‘quantile theory’ and ‘extreme value theory’ is helpful. However, there are also intermediate situations where  $r$  is a more general function of  $n$ .”

It is not clear how to deal with this when the value of  $n$  is fixed (bear in mind that David’s  $n$  corresponds to our  $2^n - 1$ ), but it indicates that the  $m$ -th most extreme order statistics for some unknown value of  $m$  (or some extreme ratio  $m/n$ ) may also fail to be described accurately by the Normal approximation, not just the single highest and lowest. Another useful topic for future research would be to conduct investigations into the value of this  $m$ , or the ratio  $m/n$ , for various values of ( $N$  and  $n$ ).

### 3.4 Theoretical complexity with the maximum-bias model.

In this model, 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 a single linear approximation is used in conventional linear cryptanalysis.

### 3.5 Theoretical complexity with the log-likelihood ratio.

The pseudocode below, headed “Algorithm 2”, describes a procedure by which the log-likelihood ratio can be used to assign a score to each candidate  $k_1$ . The higher the score, the more likely the candidate is to be the correct one,  $k_1^0$ .

In brief, we begin by converting the values in  $Cx$  from values of (number of times the approximation held) - (number of times the approximation did not hold) to fractions of known plaintexts for which the approximation held. This gives us a set of empirical probability distributions which can be compared to the theoretical probability distribution. Since we need to know the bias itself, not just its magnitude, we will have to utilise two probability distributions. One of these,  $p_1$  will be the one calculated using the Piling-Up Lemma. The other,  $p_2$ , based on the fact that the parity of the key bits involved in the linear characteristic may have been 1, will be such that all values  $p_2(x)$  are equal to  $(1 - \text{the corresponding } P_1(x))$ . We will refer to this as the “flipped” distribution.

For each possible candidate  $k_1$ , we test each possible candidate  $k_2$  in turn, by assuming that the related approximation corresponding to that value is the correct one - the “primary approximation” - and then comparing the bias with which each related approximation held to its expected bias according to the theoretical distribution. We calculate the log-likelihood ratio for each related and its corresponding theoretical distribution should the current  $k_2$  be the correct one, before summing all of these LLR values. The procedure is repeated for the flipped distribution, and the highest sum of LLR values is considered to be the “score”  $T(k_1, k_2)$  for the current candidate ( $k_1, k_2$ ) pair. The maximum of all these is the score  $T(k_1)$  for the current candidate value of  $k_1$ .

To construct a statistical framework for this method, we are again forced to assume that all related approximations are statistically independent, and to state that empirical evidence will be needed for the performance of the attack where this is not so. We build on the statistical framework for multidimensional linear cryptanalysis with the LLR statistic [17], based in turn on results from key works on order statistics and statistical frameworks in cryptanalysis [32, 66]. Let  $l$  denote the length of  $k_1$ , and  $m$  the length of  $k_2$ . Any related approximations which are expected to have zero bias can be ignored, since they do not contribute anything to the capacity of the overall approximation. Since if a given  $k_2$  is correct we know the values of  $\alpha$  such that the related approximation for  $k_2 \oplus \alpha$  has zero bias, it is practical to omit them from consideration. Let the number of relateds with bias 0 be denoted  $Z$  and let  $M = 2^m - Z$ .  $r_i$  denotes the  $i$ th related approximation with nonzero bias ( $0 \leq i < M$ ), and  $C(r_i)$  the theoretical capacity of this approximation (i.e.  $4 \times$  the square of the bias).  $C$  or  $C(p)$  denotes the overall theoretical capacity (i.e.  $4 \times$  the sum of squares of the biases, or  $\sum_{i=0}^{M-1} C(r_i)$ ).

For the correct ( $k_1, k_2$ ), and the correct choice of flipped and non-flipped theoretical distribution, each related’s LLR value is Normally distributed:

$$LLR(\text{empirical, correct choice of theoretical or flipped, uniform}) \sim \mathcal{N}(\mu_{R_i}, \sigma_{R_i}^2) \quad (3.4)$$

where  $\mu_{R_i} = \frac{NC(r_i)}{2}$ , and  $\sigma_{R_i}^2 = NC(r_i)$ .

---

**Algorithm 2** Key-ranking using the log-likelihood ratio

---

▷  $\theta$  denotes the uniform distribution.

$l \leftarrow$  the length of  $k_1$ .

$m \leftarrow$  the length of  $k_2$ .

$biased\_relateds \leftarrow$  the number of related approximations with nonzero theoretical bias.

**for**  $current\_key \leftarrow 0, 2^l - 1$  **do**

**for**  $current\_k_2 \leftarrow 0, 2^m - 1$  **do**

        ▷ This is the vector  $Cx$  as defined previously.

        ▷ We convert the values in  $Cx$  to the fraction of known

        ▷ plaintexts for which each related approximation held.

$Cx[current\_key][current\_k_2] \leftarrow (Cx[current\_key][current\_k_2] + N)/2N$

**end for**

**end for**

▷ Allocate memory for a 2D array of floating-point values

$llr\_theoretical \leftarrow$  floating-point[ $2^m$ ][ $biased\_relateds$ ]

▷ Allocate memory for another 2D array.

▷ For the reasons stated in subsection 1.1,

    ▷ the signs of the biases may be the

▷ opposite of those theoretically predicted.

$llr\_flipped \leftarrow$  floating-point[ $2^m$ ][ $biased\_relateds$ ]

**for**  $current\_k_1 \leftarrow 0, 2^l - 1$  **do**

$best\_llr\_sum \leftarrow -\infty$

**for**  $current\_k_2 \leftarrow 0, 2^m - 1$  **do**

$current\_biased \leftarrow 0$

**for**  $current\_related \leftarrow 0, 2^m - 1$  **do**

            ▷  $has\_nonzero\_bias$  is an array of bool.

            ▷  $has\_nonzero\_bias[x] = TRUE$  if the  $x$ th

            ▷ related approximation has nonzero theoretical bias.

**if**  $has\_nonzero\_bias[current\_related \oplus current\_k_2] = TRUE$  **then**

$llr\_theoretical[current\_k_2][current\_biased] \leftarrow LLR(Cx[current\_k_1][current\_related],$   
                 $theoretical\_distribution[current\_related \oplus current\_k_2], \theta)$

$llr\_flipped[current\_k_2][current\_biased] \leftarrow LLR(Cx[current\_k_1][current\_related],$   
                 $flipped\_distribution[current\_related \oplus current\_k_2], \theta)$

$current\_biased \leftarrow current\_biased + 1$

**end if**

**end for**

$llr\_sum \leftarrow \sum_{i=0}^{biased\_relateds} (llr\_theoretical[current\_k_2][current\_biased])$

$flipped\_llr\_sum \leftarrow \sum_{i=0}^{biased\_relateds} (llr\_flipped[current\_k_2][current\_biased])$

**if**  $flipped\_llr\_sum > llr\_sum$  **then**

$current\_k_2\_best\_sum = flipped\_llr\_sum$

**else**

$current\_k_2\_best\_sum = llr\_sum$

**end if**

**if**  $current\_k_2\_best\_sum > best\_llr\_sum$  **then**

$best\_llr\_sum = current\_k_2\_best\_sum$

**end if**

**end for**

$score[current\_k_1] \leftarrow best\_llr\_sum$

**end for**

---

For an incorrect value of  $k_1$ , regardless of whether the distribution is the correct choice of flipped and non-flipped or not:

$$LLR(\text{empirical, theoretical or flipped, uniform}) \sim \mathcal{N}(\mu_{W_i}, \sigma_{W_i}^2) \quad (3.5)$$

where  $\mu_{W_i} = -\frac{NC(r_i)}{2}$ , and  $\sigma_{W_i}^2 = NC(r_i)$ .

The sum of two independent Normally distributed random variables,  $v_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$  and  $v_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$  is itself Normally distributed:  $(v_1 + v_2) \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$ . It follows that the sum of  $M$  independent Normal random variables is Normally distributed with mean  $\sum_{i=0}^{M-1} \mu_i$  and variance  $\sum_{i=0}^{M-1} \sigma_i^2$ . Therefore, for an incorrect value of  $k_1$ , the sum of the LLR statistics has mean

$$\mu_W = \sum_{i=0}^{M-1} \mu_{W_i} = - \sum_{i=0}^{M-1} \frac{NC(r_i)}{2} = -\frac{NC(p)}{2}$$

and variance

$$\sigma_W^2 = \sum_{i=0}^{M-1} \sigma_{W_i}^2 = \sum_{i=0}^{M-1} NC(r_i) = NC(p)$$

Similarly, for the correct value of  $k_1$ , we have:

$$\mu_R = \sum_{i=0}^{M-1} \mu_{R_i} = \sum_{i=0}^{M-1} \frac{NC(r_i)}{2} = \frac{NC(p)}{2}$$

and

$$\sigma_R^2 = \sum_{i=0}^{M-1} \sigma_{R_i}^2 = \sum_{i=0}^{M-1} NC(r_i) = NC(p)$$

For any given incorrect value for  $k_1$ , we can deduce [32] that the CDF for the maximal sum of LLRs with the empirical, theoretical, and uniform distributions will be:

$$F_{W_{max}}(x) = \Phi_{\mu_W, \sigma_W^2}(x)^{2^m - Z} = \Phi_{\mu_W, \sigma_W^2}(x)^M$$

This is also the CDF for the maximal sum of LLRs with the empirical, flipped, and uniform distributions. We can therefore deduce that the CDF for the maximal sum of LLRs for either of the theoretical and flipped distributions, for any incorrect value of  $k_1$ , will be:

$$F_W(x) = \Phi_{\mu_W, \sigma_W^2}(x)^{M+1} \quad (3.6)$$

Differentiating this using the chain rule, we obtain the PDF:

$$f_W(x) = (M+1) \left( \Phi_{\mu_W, \sigma_W^2}(x)^M \right) \frac{1}{\sigma_W} \phi \left( \frac{x - \mu_W}{\sigma_W} \right)$$

Now, if we are aiming for advantage  $a$ , then the correct  $k_1$  must be one of the  $2^{l-a}$  highest-ranked keys. Using the notation of Hermelin et al. [17], we define  $r = 2^{l-a}$ . The correct key must have a score higher than that of the  $r$ th highest-scoring wrong key. Let this score be denoted  $T_r$ . We know [17, 66] that the distribution of  $T_r$  is approximately Normal:  $T_r \sim \mathcal{N}(\mu_a, \sigma_a^2)$ , where  $\mu_a \approx F_W^{-1}(1 - 2^{-a})$ , and:

$$\sigma_a^2 = \frac{q(1-q)}{2^l f_W(\mu_a)^2} \approx \frac{2^{-a}(1-2^{-a})}{2^l f_W(\mu_a)^2} = \frac{2^{-(a+l)} - 2^{-(2a+l)}}{f_W(\mu_a)^2}$$

(As stated in the section on the  $\chi^2$  statistic, the accuracy of the Normal approximation to  $T_r$ 's distribution breaks down for advantage near the maximum value  $l$ , but except for the special case  $a = l$ , no statistical model is known that covers this situation.)

We have:

$$\begin{aligned}\mu_a &\approx F_W^{-1}(1 - 2^{-a}) \\ &= \sigma_W \Phi^{-1}\left(\sqrt[M+1]{1 - 2^{-a}}\right) + \mu_W \\ &= \sqrt{NC} \Phi^{-1}\left(\sqrt[M+1]{1 - 2^{-a}}\right) - NC/2\end{aligned}$$

Let  $b$  denote  $\Phi^{-1}\left(\sqrt[M+1]{1 - 2^{-a}}\right)$ , and we have  $\mu_a \approx \sqrt{NC}b - NC/2$ . Next up, we calculate  $f_W(\mu_a)$ :

$$\begin{aligned}f_W(\mu_a) &= (M + 1) \left( \Phi_{\mu_W, \sigma_W^2}(\mu_a)^M \right) \frac{1}{\sigma_W} \phi\left(\frac{\mu_a - \mu_W}{\sigma_W}\right) \\ &= (M + 1) (F_W(F_W^{-1}(1 - 2^{-a}))^{M/(M+1)}) \frac{1}{\sigma_W} \phi\left(\frac{\mu_a - \mu_W}{\sigma_W}\right) \\ &= (M + 1) ((1 - 2^{-a})^{M/(M+1)}) \frac{1}{\sqrt{NC}} \phi\left(\frac{\mu_a + NC/2}{\sqrt{NC}}\right) \\ &\approx (M + 1) ((1 - 2^{-a})^{1-1/(M+1)}) \frac{1}{\sqrt{NC}} \phi\left(\frac{\sqrt{NC}b}{\sqrt{NC}}\right) \\ &= (M + 1) ((1 - 2^{-a})^{1-1/(M+1)}) \frac{1}{\sqrt{NC}} \phi(b)\end{aligned}$$

from which we derive  $\sigma_a^2$ :

$$\begin{aligned}\sigma_a^2 &\approx \frac{2^{-(a+l)} - 2^{-(2a+l)}}{f_W(\mu_a)^2} \\ &\approx \frac{2^{-(a+l)} - 2^{-(2a+l)}}{(M + 1)^2 ((1 - 2^{-a})^{2(1-1/(M+1))}) \frac{1}{NC} \phi(b)^2} \\ &= \frac{(2^{-(a+l)} - 2^{-(2a+l)}) NC}{(M + 1)^2 ((1 - 2^{-a})^{2(1-1/(M+1))}) \phi(b)^2}\end{aligned}$$

It is stated by Hermelin et al. [17], that  $\sigma_a^2 \ll \sigma_R^2$ . For clarity, we restate this as  $\sigma_a^2 \ll NC$ . However, due to the presence of  $\phi(b)^2$  in the denominator, it is not intuitively clear that this is the case. As  $M$  and  $a$  increase,  $\sqrt[M+1]{1 - 2^{-a}} \rightarrow 1$ , hence  $\Phi^{-1}\left(\sqrt[M+1]{1 - 2^{-a}}\right) \rightarrow \infty$ . Since  $b \rightarrow \infty$ ,  $\phi(b) \rightarrow 0$ .

We attempt to obtain a lower bound for the value  $\sigma_R^2/\sigma_a^2$ . For any fixed pair  $(a, M)$ , it should be clear that the lowest value of  $l$  such that  $l \geq a$  maximises  $\sigma_a^2$ . Therefore, for any fixed value of  $a \geq 3$ ,  $\sigma_a^2$  is maximised by setting  $l = a$ . For the same reasons as before, we assume that  $l \geq 3$ , and hence for any  $a < 3$  we set  $l = 3$ . Using Mathematica to plot graphs based on these assumptions, we find that the value  $\sigma_R^2/\sigma_a^2$  is minimised for  $M = 1$ ,  $a = 3$ , and is always greater than 4.

(This particular fraction can take values much higher than 4 - in the case of the multidimensional linear attack on 12-round Serpent by Nguyen et al. [60], with  $a = 172$  and  $m = 56$ ,  $\sigma_R^2/\sigma_a^2 \approx 310.5$ . For the multidimensional attacks on 11-round Serpent in the same paper with  $m = 56$ , we calculate  $\sigma_R^2/\sigma_a^2 \approx 133.87$  for advantage 44 and  $\sigma_R^2/\sigma_a^2 \approx 139.38$  for advantage 48. So there are clearly situations where we can reasonably assume  $\sigma_a^2 \ll \sigma_R^2$ .)

Let  $T_{2^n-r}$  denote the  $r$ th-highest LLR value computed using any of the wrong-key candidates. Let  $T(k_1^0)$  be the LLR-based score computed using the correct value of  $k_1$ . Hermelin et al. define the following three probabilities:

- $P_1$  denotes the probability that  $k_2^0$  resulted in the highest sum of LLRs when computing  $T(k_1^0)$ ;  $Pr(T(k_1^0, k_2^0) > \max_{k_2 \neq k_2^0} T(k_1^0, k_2))$ . (Equivalently,  $Pr(T(k_1^0, k_2^0) = T(k_1^0))$ ).
- $P_2$  denotes the probability of success for Algorithm 2 - that is, the probability that  $k_1^0$  is one of the  $2^{l-a}$  highest-ranked keys,  $Pr(T(k_1^0) > T_{2^{n-r}})$ .
- $P_{12}$  is initially used to denote  $Pr(T(k_1^0) > T_{2^{n-r}} | T(k_1^0, k_2^0) = T(k_1^0))$ .

However, it is later defined as  $Pr(T(k_1^0, k_2^0) > T_{2^{n-r}})$ , which is a lower bound for  $P_2$ . This is used as a conservative estimate for the success probability, since the distribution for an *LLR* value corresponding to an incorrect  $k_2$  value in conjunction with  $k_1^0$  is unknown and therefore  $P_2$  cannot be calculated.

We will do the same here, for the same reason, but will refer to  $Pr(T(k_1^0, k_2^0) > T_{2^{n-r}})$  as  $P_3$  to avoid ambiguity.

$$\begin{aligned} P_3 &= \Phi \left( \frac{\mu_R - \mu_a}{\sqrt{(\sigma_R^2 + \sigma_a^2)}} \right) \\ &= \Phi \left( \frac{NC/2 - \sqrt{NC}b + NC/2}{\sqrt{(NC + \sigma_a^2)}} \right) \end{aligned}$$

As noted, the ratio of  $\sigma_a^2$  to  $\sigma_R^2 = NC$  can vary significantly depending on the cipher being attacked and the nature of the linear approximation. For the cryptanalyst, the worst-case scenario is that calculated above, in which  $\sigma_a^2 \approx NC/4$ . In this case:

$$\begin{aligned} P_3 &\approx \Phi \left( \frac{NC/2 - \sqrt{NC}b + NC/2}{\sqrt{(NC + NC/4)}} \right) \\ &= \Phi \left( \frac{NC - \sqrt{NC}b}{\sqrt{5NC/4}} \right) \\ &= \Phi \left( \frac{\sqrt{NC} - b}{\sqrt{1.25}} \right) \end{aligned}$$

This gives us a conservative (over)estimate for data complexity:

$$N = \frac{(1.118\Phi^{-1}(P_3) + b)^2}{C} \quad (3.7)$$

In the best-case scenario, where  $\sigma_a^2 \ll NC$ , we have instead:

$$N = \frac{(\Phi^{-1}(P_3) + b)^2}{C} \quad (3.8)$$

Clearly, the cryptanalyst should analyse the value of  $\sigma_a^2$  in relation to  $NC$  before calculating  $N$ .

We know that  $\Phi(b) = \sqrt[M+1]{1 - 2^{-a}}$ . Hermelin et al. state that this is approximately equal to  $(1 - 2^{-a - \log_2(M+1)})$ . By plotting graphs in Mathematica, we see that the accuracy of this approximation improves as  $a$  and  $M$  increase. When  $a$  and  $M$  are both particularly small, the approximation may not be adequate. It appears, regardless of  $M$ , to be extremely close for  $a \geq 3$ . Whether it is adequate for  $a = 2$  and small  $M$  is not clear, however as  $M$  increases the approximation becomes very close for  $1 \leq a \leq 2$ .

The real question, however, is whether  $\Phi^{-1}(\sqrt[M+1]{1 - 2^{-a}})$  is adequately approximated by  $\Phi^{-1}(1 - 2^{-a - \log_2(M+1)})$ . Plotting graphs of

$$\frac{\Phi^{-1}(\sqrt[M+1]{1-2^{-a}})}{\Phi^{-1}(1-2^{-a-\log_2(M+1)})}$$

we see that the accuracy of the approximation increases rapidly with  $a$ , but increases relatively slowly as  $M$  increases. The approximation is always an overestimate for  $a \geq 1$ , so can always be used to calculate an upper bound for  $N$  even for  $a$  this low. The worst-case overestimate for  $a \geq 1$  (for  $a = 1, M = 1$ ) is  $\approx 1.25$ -fold. If the effect of  $\Phi^{-1}(P_3)$  on the value of  $N$  is assumed to be negligible compared to the effect of  $b$ , then this leads to a worst-case overestimate of  $1.5625 \times$  the correct value of  $N$ . If the effect of  $\Phi^{-1}(P_3)$  is *not* assumed to be negligible, the overestimate is less extreme. For  $a \geq 2$ , this leads to an overestimated value of at most  $1.08 \times$  the correct  $N$  - which is extremely close. We therefore accept this approximation as valid and sufficiently accurate for  $a \geq 2$ , and useful for calculating upper-bounds on  $N$  for  $a \geq 1$ . We have not investigated lower values of  $a$  in much detail, however, but do note that the approximation is not always an overestimate for these.

Accepting  $b \approx \Phi^{-1}(1-2^{-a-\log_2(M+1)})$ , we obtain by the reasoning in subsection 3.3 that:

$$\begin{aligned} (\log_2(M+1) + a) &\approx \frac{b^2}{2\ln(2)} + \log_2(b) + \log_2(2\pi) \\ &\approx 0.72b^2 + \log_2(b) + 1.325 \\ \therefore a &\approx 0.72b^2 + \log_2(b) + 1.325 - \log_2(M+1) \end{aligned} \quad (3.9)$$

To compute  $a$  from  $N, C, M$  and  $P_s$ , the cryptanalyst rearranges Equation 3.8 (or, in some circumstances, some more pessimistic equation such as Equation 3.7) to compute  $b$ , and then inputs it into the above equation. Similarly, for some desired value of  $a$  and known  $M, C, P_s$ , the cryptanalyst solves Equation 3.9 to obtain  $b$ , and uses Equation 3.8 to compute  $N$ .

We can rewrite Equation 3.9 as:

$$a \approx 0.72(\sqrt{NC} - \Phi^{-1}(P_3))^2 + \log_2(\sqrt{NC} - \Phi^{-1}(P_3)) + 1.325 - \log_2(M+1) \quad (3.10)$$

Since this equation applies equally to both nonlinear and multidimensional linear cryptanalysis, we compare it to Hermelin et al.'s Theorem 2 [17]:

$$a \approx (\sqrt{NC} - \Phi^{-1}(P_3))^2/2 - \log_2(M+1) \approx NC - \log_2(M+1) \quad (3.11)$$

Clearly the two equations are very different. We argue that ours is the more accurate for the following reasons:

- $(\sqrt{NC} - \Phi^{-1}(P_3))^2/2 - \log_2(M+1)$  is clearly  $< (NC/2) - \log_2(M+1)$ , and not  $\approx NC - \log_2(M+1)$ . Moreover, if this approximated form were to be used, it would imply that the probability of success did not affect the data complexity of the attack in any way.
- Equation 3.11 is derived using the incorrect approximation  $a \approx (b^2/2) - \log_2(M+1)$ .

### 3.5.1 Using the LLR without key ranking.

What, then, of the special case  $a = l$ ? We shall derive a statistical framework for this based on the maximum-advantage statistical framework for conventional linear cryptanalysis.

Matsui's Table 3 [52], containing the success probabilities for various multiples of  $|p - 1/2|^{-2}$ , is calculated from the double integral in his Lemma 5 based on the assumption that the TPS length  $l$  is equal to 6. Selçuk [66] states that there is a tendency to base complexity calculations for linear cryptanalytic attacks on results from Matsui's work that specifically applied to his attacks on DES. He presents an alternative double integral for use in calculating the advantage when the cryptanalyst aims for maximum advantage  $a = l$ .

Since the two double integrals are not identical, the question of how each was derived, and which is the better choice, arises. Neither is limited to the case of  $l = 6$ , and different values may easily be input.

Examining Matsui's original integral, we observe first of all a typographical error. The limits of the internal integral include the term  $(p - 1/2)$ . This is clearly meant to be  $|p - 1/2|$ , since if not, two attacks using approximations with identical magnitude but different sign of bias would have significantly different complexity.

The intent behind the double integral would be as follows: for each possible value  $x$  for the empirical absolute bias when the correct key is used, calculate the probability that the empirical bias  $y$  for all other  $k_i$  satisfies  $(-x < y < x)$ . By integrating this over all  $x > 0$ , the probability of success of the attack is obtained.

$$\begin{aligned} & \int_0^\infty \left( \prod_{k_i \neq k_0} Pr(-x < y < x) \right) f_R(x) dx \\ &= \int_0^\infty \left( \prod_{k_i \neq k_0} \int_{-x}^x f_{W_{k_i}}(y) dy \right) f_R(x) dx \end{aligned}$$

However, examining the integrals in more depth makes it clear that a Normal distribution is assumed for  $x$ , whereas if  $x$  represents absolute bias, it would have a Folded Normal distribution. We therefore restate the intent as follows: for each possible value  $x$  for the bias when the correct key is used, calculate the probability that the empirical bias  $y$  for all other  $k_i$  satisfies  $(\min(-x, x) < y < \max(-x, x))$ . By integrating this over  $(\infty > x > -\infty)$ , the probability of success of the attack is obtained.

$$\begin{aligned} & \int_{-\infty}^0 \left( \prod_{k_i \neq k_0} Pr(x < y < -x) \right) f_R(x) dx + \int_0^\infty \left( \prod_{k_i \neq k_0} Pr(-x < y < x) \right) f_R(x) dx \\ &= \int_{-\infty}^0 \left( \prod_{k_i \neq k_0} \int_x^{-x} f_{W_{k_i}}(y) dy \right) f_R(x) dx + \int_0^\infty \left( \prod_{k_i \neq k_0} \int_{-x}^x f_{W_{k_i}}(y) dy \right) f_R(x) dx \\ &= \int_{-\infty}^0 \left( \prod_{k_i \neq k_0} \int_x^{-x} \frac{1}{\sigma_W} \phi \left( \frac{y - \mu_W}{\sigma_W} \right) dy \right) \frac{1}{\sigma_R} \phi \left( \frac{x - \mu_R}{\sigma_R} \right) dx \\ &+ \int_0^\infty \left( \prod_{k_i \neq k_0} \int_{-x}^x \frac{1}{\sigma_W} \phi \left( \frac{y - \mu_W}{\sigma_W} \right) dy \right) \frac{1}{\sigma_R} \phi \left( \frac{x - \mu_R}{\sigma_R} \right) dx \end{aligned}$$

Selçuk simplifies the calculation by assuming that the Wrong-Key Randomization Hypothesis (WKRH) applies. That is, for all incorrect candidate values  $k_i \neq k_0$  for the TPS  $k_1$ , he assumes that the biases have identical underlying probability distributions with mean 0. We will also do so here:

$$\begin{aligned} & \int_{-\infty}^0 \left( \int_x^{-x} \frac{1}{\sigma_W} \phi \left( \frac{y - \mu_W}{\sigma_W} \right) dy \right)^{2^l - 1} \frac{1}{\sigma_R} \phi \left( \frac{x - \mu_R}{\sigma_R} \right) dx \\ &+ \int_0^\infty \left( \int_{-x}^x \frac{1}{\sigma_W} \phi \left( \frac{y - \mu_W}{\sigma_W} \right) dy \right)^{2^l - 1} \frac{1}{\sigma_R} \phi \left( \frac{x - \mu_R}{\sigma_R} \right) dx \end{aligned}$$

Let us integrate by substitution. Firstly, let  $u = ((x - \mu_R)/\sigma_R)$ :

$$\begin{aligned}
& \int_{-\infty}^{\frac{0-\mu_R}{\sigma_R}} \left( \int_x^{-x} \frac{1}{\sigma_W} \phi \left( \frac{y-\mu_W}{\sigma_W} \right) dy \right)^{2^l-1} \phi(u) du \\
& + \int_{\frac{0-\mu_R}{\sigma_R}}^{\infty} \left( \int_{-x}^x \frac{1}{\sigma_W} \phi \left( \frac{y-\mu_W}{\sigma_W} \right) dy \right)^{2^l-1} \phi(u) du \\
& = \int_{-\infty}^{\frac{0-\mu_R}{\sigma_R}} \left( \int_{u\sigma_R+\mu_R}^{-u\sigma_R-\mu_R} \frac{1}{\sigma_W} \phi \left( \frac{y-\mu_W}{\sigma_W} \right) dy \right)^{2^l-1} \phi(u) du \\
& + \int_{\frac{0-\mu_R}{\sigma_R}}^{\infty} \left( \int_{-u\sigma_R-\mu_R}^{u\sigma_R+\mu_R} \frac{1}{\sigma_W} \phi \left( \frac{y-\mu_W}{\sigma_W} \right) dy \right)^{2^l-1} \phi(u) du
\end{aligned}$$

Selçuk states that the bias of the correct key has a Normal distribution, with mean  $\mu_R = (p-1/2)$  and variance  $\sigma_R^2 = 1/4N$ . The figure for the variance appears to be derived from Junod [43]; we have no reason to doubt it.

If  $p > 1/2$ , the mean  $(p-1/2)$  is equal to  $|p-1/2|$ . Let  $N = a \times |p-1/2|^{-2}$ , and we have:

$$\frac{0-\mu_R}{\sigma_R} = 2\sqrt{N}(0-|p-1/2|) = 2\sqrt{a}|p-1/2|^{-1}(0-|p-1/2|) = -2\sqrt{a}$$

Assuming  $a > 2$ ,  $(0-\mu_R)/(\sigma_R) < -2\sqrt{2} \approx -2.828$ .  $\Phi(-2\sqrt{2}) \approx 1/427$ , implying that the contribution of the first integral will be negligible and that we can approximate the whole expression with:

$$\int_{\frac{0-\mu_R}{\sigma_R}}^{\infty} \left( \int_{-u\sigma_R-\mu_R}^{u\sigma_R+\mu_R} \frac{1}{\sigma_W} \phi \left( \frac{y-\mu_W}{\sigma_W} \right) dy \right)^{2^l-1} \phi(u) du$$

If  $p < 1/2$ , the mean  $(p-1/2)$  is equal to  $-|p-1/2|$ . Again, let  $N = a \times |p-1/2|^{-2}$ , and we have:

$$\frac{0-\mu_R}{\sigma_R} = 2\sqrt{N}(0+|p-1/2|) = 2\sqrt{a}|p-1/2|^{-1}(0+|p-1/2|) = 2\sqrt{a}$$

Still assuming  $a > 2$ ,  $(0-\mu_R)/(\sigma_R) > 2\sqrt{2} \approx 2.828$ .  $P(u > 2\sqrt{2}) = 1 - \Phi(2\sqrt{2}) \approx 1/427$ , implying that the contribution from the second integral will be negligible and that we can approximate the whole with:

$$\int_{-\infty}^{\frac{0-\mu_R}{\sigma_R}} \left( \int_{u\sigma_R+\mu_R}^{-u\sigma_R-\mu_R} \frac{1}{\sigma_W} \phi \left( \frac{y-\mu_W}{\sigma_W} \right) dy \right)^{2^l-1} \phi(u) du$$

On the other hand, for  $p < 1/2$ , we could have defined  $x$  as  $-1 \times$  the bias at the start, and obtained the same approximation as for the case  $p > 1/2$ . This means that the probability of success can in both cases be approximated by:

$$\int_{\frac{0-\mu_R}{\sigma_R}}^{\infty} \left( \int_{-u\sigma_R-\mu_R}^{u\sigma_R+\mu_R} \frac{1}{\sigma_W} \phi \left( \frac{y-\mu_W}{\sigma_W} \right) dy \right)^{2^l-1} \phi(u) du$$

with  $\mu_R = |p-1/2|$  and  $\sigma_R = 1/2\sqrt{N}$ . (If this were not the case, we would have been faced with a situation where the sign of the approximation's bias affected the performance of the attack despite the cryptanalyst discarding this information and taking the absolute bias in Algorithm 2.)

Let us now complete the substitution of  $|p-1/2|$  for  $\mu_R$  and  $1/2\sqrt{N}$  for  $\sigma_R$  in the equation above:

$$\int_{-2\sqrt{N}|p-1/2|}^{\infty} \left( \int_{-u/2\sqrt{N}-|p-1/2|}^{u/2\sqrt{N}+|p-1/2|} \frac{1}{\sigma_W} \phi \left( \frac{y-\mu_W}{\sigma_W} \right) dy \right)^{2^l-1} \phi(u) du$$

Then, we integrate by substitution again. Let  $v = (y - \mu_W)/\sigma_W$ , with  $\mu_W = 0$  and  $\sigma_W = 1/2\sqrt{N}$  [43], and we have:

$$\begin{aligned} & \int_{-2\sqrt{N}|p-1/2|}^{\infty} \left( \int_{\frac{-u/2\sqrt{N}-|p-1/2|-\mu_W}{\sigma_W}}^{\frac{u/2\sqrt{N}+|p-1/2|-\mu_W}{\sigma_W}} \phi(v)dv \right)^{2^l-1} \phi(u)du \\ &= \int_{-2\sqrt{N}|p-1/2|}^{\infty} \left( \int_{\frac{-u/2\sqrt{N}-|p-1/2|-0}{1/2\sqrt{N}}}^{\frac{u/2\sqrt{N}+|p-1/2|-0}{1/2\sqrt{N}}} \phi(v)dv \right)^{2^l-1} \phi(u)du \\ &= \int_{-2\sqrt{N}|p-1/2|}^{\infty} \left( \int_{-u-2\sqrt{N}|p-1/2|}^{u+2\sqrt{N}|p-1/2|} \phi(v)dv \right)^{2^l-1} \phi(u)du \end{aligned}$$

- precisely Selçuk's equation. We therefore accept this double integral as correct unless there is reason to believe that the WKRH does not apply, and even then we would use a modified version of Selçuk's equation in preference to Matsui's.

Let us compare the predicted values for the probability of success (denoted  $P_s$ ) in Matsui's attack on 8-round DES with  $l = 6$ :

$N$	$2 p-1/2 ^{-2}$	$4 p-1/2 ^{-2}$	$8 p-1/2 ^{-2}$	$16 p-1/2 ^{-2}$
$P_s$ ( $l = 6$ , Matsui)	0.486	0.785	0.967	0.999
$P_s$ ( $l = 6$ , Selçuk)	0.589331	0.902745	0.997249	0.999999

Table 4: Comparison of success rates (calculated numerically using Wolfram Mathematica) for  $l = 6$  according to Matsui [52] and Selçuk [66]

Clearly, unless Matsui had reason to believe that the wrong-key randomization hypothesis did not hold, his original equation gave pessimistic estimates for the success probability of Algorithm 2 without key ranking.

So far, our reasoning has applied only to conventional linear cryptanalysis with  $a = l$ . We are now, however, in a position to carry out the generalisation to the  $a = l$  situation when using nonlinear and multidimensional linear cryptanalysis with the LLR statistic.

The intention behind the new double integral for these cases is as follows: For each possible value  $x$  for the empirical LLR (or sum thereof - we use one LLR as the statistic in multidimensional linear and a sum of LLRs in nonlinear) when the correct outer key value  $k_0$  is used, calculate the probability that the empirical LLR/sum of LLRs  $y$  for all other  $k_i$  is less than  $x$ . By integrating this over  $(-\infty > x > \infty)$ , the probability of success is obtained. Assuming that the WKRH holds:

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

For large  $M$  and  $l$ , this is not easy to calculate numerically. For example, with  $M = 2^{56} - 1$  as per the attacks of Nguyen et al. [60], Wolfram Mathematica displays error messages relating to "underflows" when calculating:

$$\left(\Phi_{\mu_W, \sigma_W^2}(x)^{M+1}\right)$$

due to the high precision needed to handle numbers extremely close to 0. Until some means can be found to overcome this, we are forced to continue relying on Equation 3.10 for our data complexity estimates.

### 3.5.2 Time complexity.

The time complexity of the LLR scoring phase is dominated by the  $2^{|k_1|+|k_2|+1} \times \text{biased\_relateds}$  calls to the `LLR()` function to compute the log-likelihood ratio. This makes the complexity difficult to calculate, since computing the log-likelihood ratio involves the calculation of logarithms (in our implementation, two calls to a `log()` function are needed for each call to an `LLR()` function.). Various different algorithms for this exist, using numerical algorithms to compute a logarithm accurate to some implementer-defined precision, and it may not even be possible to convert complexity calculations in terms of “speed of convergence” to expressions in Big-Oh notation. In any case, calculations based on one compiler’s implementation of the function to compute logarithms to a particular base may not apply to other compilers, or even other versions of the same compiler.

Approximations to the LLR do exist which do not require the computation of logarithms, and which are thus considerably faster to compute and easier to analyse:

### 3.5.3 Taylor series approximation.

Where  $D(g||h)$  denotes the Kullback-Leibler distance between discrete probability distributions  $g$  and  $h$ , each with  $M$  possible values,  $LLR(\hat{q}, p, q) = N \times D(\hat{q}||q) - N \times D(\hat{q}||p)$  [30]. If  $g$  and  $h$  are close,  $D(g||h)$  can be approximated by the first term of its Taylor series [17] [30]:

$$D(g||h) \approx \sum_{i=0}^M \frac{(g_i - h_i)^2}{2h_i}$$

We therefore compute the approximated  $D(\hat{q}||q)$  and  $D(\hat{q}||p)$  - optimising based on the fact that  $q$  is the uniform distribution - and then subtract the second from the first and multiply by  $N$ .

### 3.5.4 Approximation using the convolution method.

Previous research in multidimensional linear cryptanalysis [38] argued that the data complexity of an attack using the log-likelihood ratio was of the same order of magnitude as the complexity of an attack using the convolution of distributions  $p$  and  $\hat{q}$ , and presented experimental evidence that, for an attack based on Matsui’s Algorithm 1, the data complexity was in practice near-identical. Since this method has a much lower time complexity than the LLR method, it is currently the favoured ranking statistic for multidimensional linear cryptanalysis.

We can use the convolution of  $p$  and  $\hat{q}$  to approximate the LLR in nonlinear cryptanalysis. The approximation is not quite so close as that based on Taylor series, but the data complexities are still almost identical and this method has the lowest time complexity. For any related approximation used in the nonlinear attack, let  $\rho(a)$  denote the empirical correlation; that is,  $2 \times$  the empirical bias. Let  $c(a)$  denote the theoretical correlation as predicted by the Piling-Up Lemma - hence,  $-c(a)$  is the theoretical correlation for the “flipped” distribution.  $-\rho(a)$  is what the empirical correlation would be had we used the “flipped” nonlinear approximation,  $1 \oplus$  that which we actually used.

$$(c(a) - \rho(a))^2 = (\rho(a)^2 + c(a)^2) - 2c(a)\rho(a)$$

Since  $c(a)^2$  is constant regardless of the current key candidate (and the value of  $a$ , which is in fact always equal to 1 due to the fact that there is only one dimension and  $c(0)$  contains no information in a multidimensional attack), we have:

$$\begin{aligned}
2c(a)\rho(a) + \text{const} &= (\rho(a)^2) - (c(a) - \rho(a))^2 \\
&\equiv 2c(1)\rho(1) + \text{const} = (\rho(1)^2) - (c(1) - \rho(1))^2
\end{aligned}$$

Correlation =  $2 \times$  bias, and hence  $\rho(a)^2$  is  $4 \times$  the Euclidean distance between the empirical bias and that predicted by the uniform distribution. Similarly,  $(c(a) - \rho(a))^2$  is  $4 \times$  the Euclidean distance between the empirical bias and that predicted by the theoretical distribution. In other words,  $2 \cdot c(a)\rho(a) + \text{a fixed constant}$  gives us the distance between the empirical and uniform distributions, minus the distance between empirical and theoretical, albeit using Euclidean distance instead of Kullback-Leibler. Clearly  $2 \cdot c(a)\rho(a) + c$  contains the same information as  $c(a)\rho(a)$ , and the higher it is, the more likely that the theoretical distribution (not the flipped) is correct and the correct key candidate has been chosen.

Now, let  $k$  equal 1 if the flipped distribution is correct, 0 if the original “unflipped” theoretical distribution is.  $\sum_{a=0}^1 (-1)^{a \cdot k} c(a)\rho(a) = c(0)\rho(0) + c(1)\rho(1)$  if  $k = 0$ , else  $c(0)\rho(0) - c(1)\rho(1)$ . In a multidimensional attack,  $c(0)$  and  $\rho(0)$  would be correlations for a linear sum of no approximations; i.e.  $c(0)\rho(0)$  is a fixed constant containing no information relevant to the attack.

It is shown [38] that  $\sum_{a=0}^1 (-1)^{a \cdot k} c(a)\rho(a)$  is equal to the  $k$ th component of the convolution  $p * \hat{q}$  of the probability distributions  $p$  and  $\hat{q}$ . Hence, whichever value of  $k$  maximises  $\sum_{a=0}^1 (-1)^{a \cdot k} c(a)\rho(a)$  also corresponds to the maximum component of the convolution. Since the convolution of two probability distributions is itself a probability distribution, we need only check whether component 0 is greater than 0.5 to know which is the maximum component. Since the Convolution Theorem shows that the convolution of two probability distributions can be calculated in  $O(M \log(M))$  time using the FFT, this leads to a significant reduction in time complexity for the high values of  $M$  used in multidimensional linear cryptanalysis, and even for the lower  $M = 2$  for each related approximation in nonlinear cryptanalysis, allows us to carry out an efficient calculation using only arithmetic operators instead of having to call an expensive and hard to analyse logarithm function.

---

**Algorithm 3** Key-ranking using the convolution method

---

$\triangleright \theta$  denotes the uniform distribution.

```
l ← the length of k1.
m ← the length of k2.
biased_relateds ← the number of related approximations with nonzero theoretical bias.
for current_key ← 0, 2l − 1 do
  for current_k2 ← 0, 2m − 1 do
    Cx[current_key][current_k2] ← (Cx[current_key][current_k2] + N)/2N
  end for
end for
 $\triangleright$  Allocate memory for a 2D array of floating-point values
convols ← floating-point[2m][biased_relateds]
for current_k1 ← 0, 2l − 1 do
  best_convolution_sum ← −∞
  for current_k2 ← 0, 2m − 1 do
    current_biased ← 0
    for current_related ← 0, 2m − 1 do
      if has_nonzero_bias[current_related ⊕ current_k2] = TRUE then
        convols[current_k2][current_biased] ← convolution(Cx[current_k1][current_related],
          theoretical_distribution[current_related ⊕ current_k2])
        current_biased ← current_biased + 1
      end if
    end for
    convolution_sum ←  $\sum_{i=0}^{biased\_relateds} (convols[current\_k_2][current\_biased])$ 
    if convolution_sum > 0.5 × biased_relateds then
      current_k2_best_sum = convolution_sum
    else
      current_k2_best_sum = biased_relateds − convolution_sum
    end if
    if current_k2_best_sum > best_convolution_sum then
      best_convolution_sum = current_k2_best_sum
    end if
  end for
  score[current_k1] ← best_convolution_sum
end for
```

---

What is the theoretical complexity of Algorithm 3? Although considerably faster than the original LLR algorithm, it is still much slower than the key-ranking algorithms for the maximum bias and  $\chi^2$  frameworks, and its contribution to the overall attack complexity is nontrivial. Moreover, even slight optimisations or complexity improvements in the algorithm to compute the convolution may significantly affect complexity in terms of memory accesses and arithmetic operations. We need only seven arithmetic operations to compute each convolution due to the small size of the probability distributions, but this can be reduced to six by computing ( $2 \times$ convolution) instead, and altering the rest of Algorithm 3 accordingly. The extremely small complexity of the convolution calculations makes it hard to dismiss any part of the complexity as negligible, and both  $|k_2|$  and *biased\_relateds* may take many different values in relation to  $|k_1|$  and each other.

We have:

- $2^{|k_1|+2 \times |k_2|}$  memory accesses to check *has\_nonzero\_bias*.
- (Assuming the aforementioned optimisation)  $2^{|k_1|+|k_2|+biased\_relateds} \times 6$  AOs for the convolutions.
- Approximately  $2^{|k_1|+|k_2|+biased\_relateds}$  AOs and MAs in summing the modified convolutions.

- At most  $(8 \times 2^{|k_1|+|k_2|}) + (2 \times 2^{|k_1|})$  MAs after this.

This gives us

- $2^{|k_1|+|k_2|+biased\_relateds+2.8}$  AOs.
- $2^{|k_1|+2 \times |k_2|} + 2^{|k_1|+|k_2|+biased\_relateds} + 2^{|k_1|+|k_2|+3} + 2^{|k_1|+1}$  MAs.

The below graphs were obtained by averaging the results of fifty trials, in which we used the same nonlinear approximations from our cryptanalysis of Serpent (in the later section) to attack the Heys toy cipher [40] with fifty different keys:

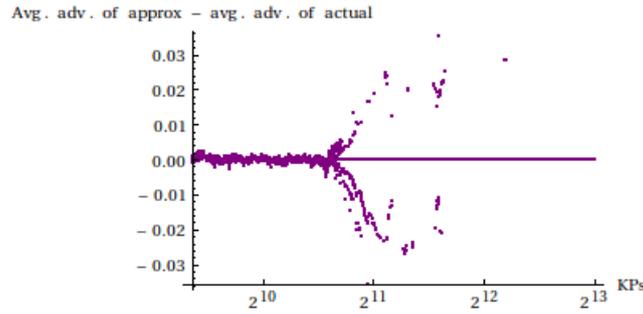


Figure 5: Average difference in advantage (where mean rank for each statistic is input to formula for advantage) between cryptanalysis with LLR statistic and Taylor-approximated LLR statistic.

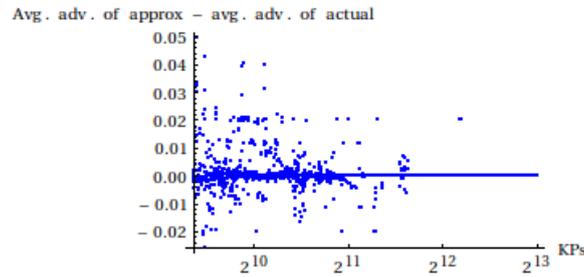


Figure 6: Average difference in advantage (based on difference in mean advantage) between cryptanalysis with LLR and with Taylor approximation.

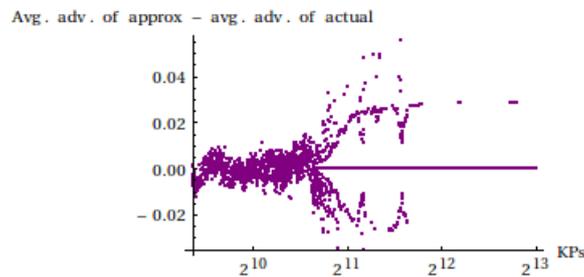


Figure 7: Average difference in advantage (where mean rank for each statistic is input to formula for advantage) between cryptanalysis using LLR statistic and cryptanalysis using convolution.

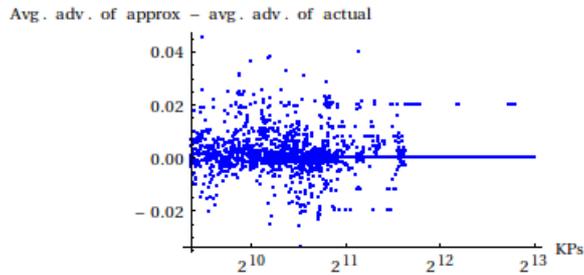


Figure 8: Average difference in advantage (based on difference in mean advantage) for LLR and convolution.

### 3.6 When the cipher is not a substitution-permutation network.

For a cipher such as DES, the procedure is not so straightforward to adapt as it is for the SPN structure, and it may not be possible to do so in all cases. Let us consider a situation in which we have incorporated nonlinear approximations into Matsui's linear attack on the full DES [53]. Let us start by adapting the part of the attack based on his Equation 4:

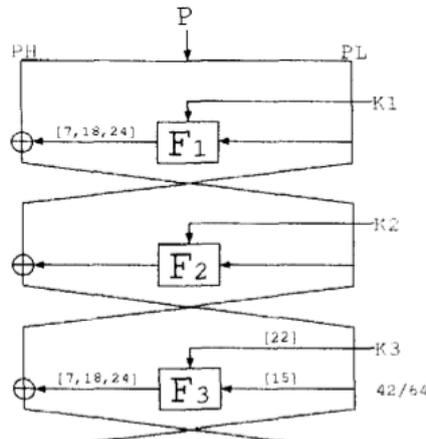


Figure 9: Diagram showing the first three rounds of DES in Matsui's attack. The numbers in square brackets indicate the active bits using Matsui's indexing system [53], and the fraction on the right shows the probability  $p$  of the first round's linear approximation.

We cannot replace the third-round xor of bits [7, 18, 24] with a nonlinear term due to the xors which are applied to it; a nonlinear term in variables  $z[i]$  is not equal to the xor of (the same nonlinear term in variables  $x[i]$ ) with (the same nonlinear term in variables  $y[i]$ ). Therefore, we cannot incorporate nonlinear components into the first round of the approximation.

The final round is another matter. We can replace the linear approximation to DES S5 with any nonlinear approximation with output bitmask 15 and nonlinear input component. There are several of these such that at least one of the set of relateds has bias  $\pm 24$ ; at present our metaheuristic algorithm has found sixty-two. Of these (numbering the S-box input bits from 1 to 6, with the MSB being 1):

- One approximation uses S-box input bits 1, 2, 5, 6. Although we number the S-box input bits  $x_i$  differently, this is the approximation found by Knudsen and Robshaw [46].
- Thirty-one approximations use input bits 1, 2, 4, 5, 6.
- Thirty approximations use input bits 1, 2, 3, 5, 6.

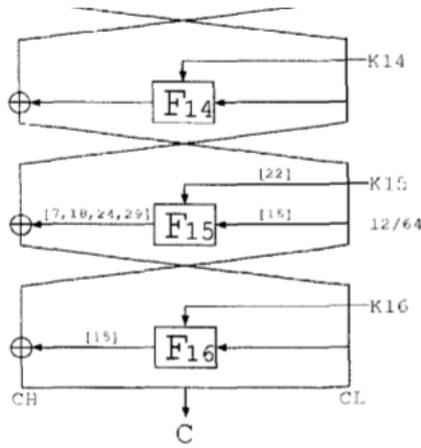


Figure 10: Diagram showing the last three rounds of DES in Matsui's attack. As in Figure 9, the numbers in square brackets indicate the active bits using Matsui's indexing system [53], and the fraction on the right shows the probability  $p$  of the penultimate round's linear approximation.

Let us look at how we can adapt the new procedure to the first of these cases.

First of all, we will need to decrypt S-boxes in Round 16 to expose the data bits relevant to S5 in Round 15:

Round 15 data bit	Round 16 data bit
S5 input bit 1	S3 output bit 2
S5 input bit 2	S1 output bit 2
S5 input bit 3	S2 output bit 4
S5 input bit 4	S6 output bit 4
S5 input bit 5	S4 output bit 2
S5 input bit 6	S8 output bit 4

Table 5: S-box output bits in Round 16 corresponding to the S5 input bits in Round 15.

In addition, we will need to guess key bits for S5 in round 15. Some of these will already have been guessed:

Round 15 key bit	Round 16 key bit
S5 input bit 1	S6 input bit 2
S5 input bit 2	S6 input bit 3
S5 input bit 3	S6 input bit 1
S5 input bit 4	S8 input bit 4
S5 input bit 5	S8 input bit 1
S5 input bit 6	N/A (main key bit 53, numbering from 0 to 55)

Table 6: Key bits corresponding to S5 input bits in Round 15, and their Round 16 counterparts where applicable.

Finally, we still have to guess key bits for S5 in round 1.

Round 1 key bit	Round 16 key bit
S5 input bit 1	S8 input bit 2
S5 input bit 2	Main key bit 52 - S7 bit 6
S5 input bit 3	S8 input bit 6
S5 input bit 4	N/A (main key bit 37)
S5 input bit 5	S6 input bit 6
S5 input bit 6	Main key bit 55 - S7 bit 4

Table 7: Key bits corresponding to S5 input bits in Round 1, and their Round 16 counterparts where applicable.

Let us consider the approximation on bits 1, 2, 5 and 6, since this provides the simplest example. We need to guess key bits for four S-boxes (S1, S3, S4, S8) in Round 16; 24 key bits corresponding to 22 text bits. To allow  $C$  to have the property that  $C_{ij} = f(i \oplus j)$ , we will work with the 24 bits resulting from applying the expansion to the text bits. We also need to guess four key bits in Round 1 - we will need to introduce two dummy key bits to correspond to the six input bits of S5 - despite knowing that they should share the same values as key bits 2 and 6 of S8. So far, we have  $|k_1| = 30$ . We also have four active text bits in the left-hand ciphertext block, which we cannot now simply xor together and treat as part of a larger xor of bits. These require us to introduce four more dummy key bits with the value zero, in addition to the dummy key bit for the xored bits in the left block of the plaintext and right block of the ciphertext. We have  $|k_1| = 35$ . Since one of the four key bits at the input to DES S5 in round 15 is active in round 16, we also have  $|k_2| = 3$ .

The question of estimating the complexity of a partial encryption/decryption in terms of DES encryptions also arises. Matsui encrypts one S-box, decrypts another, and xors various bits; as the full DES involves  $(8 \times 16) = 128$  S-boxes, we will estimate the complexity of each partial encryption/decryption in Matsui's analysis phase to be  $2/128 = 1/64$  of a full DES encryption.

In our case, this is more complicated. We encrypt one S-box and decrypt four, after which we need to compute the following algebraic expression on S5's input bits  $2^{|k_2|} = 8$  times:

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

(Note that this was the expression used as an example before.)

We therefore estimate the time required for each partial encryption/decryption to be  $5/128 + 8/128 = 13/128$  of the time required for a full DES encryption.

This specific attack, although it breaks the DES and in spite of its improved bias, turns out to have poorer data complexity than that of Matsui [53]. We will explain later on how Matsui's attack - in effect a combination of two separate attacks - is able despite its lower bias to perform more effectively. For now, though, with the method defined, we have enough information to design cost functions and run experiments, and it is therefore time to discuss the metaheuristic algorithm.

## 4 The use of simulated annealing to evolve nonlinear approximations.

(A description of the simulated annealing algorithm is provided in Appendix B, for those unfamiliar with it.)

In the previous application of simulated annealing to this problem by Clark et al. [21, 19], each nonlinear approximation was represented as follows:

- A global constant,  $k$ , determined the maximum number of S-box input bits that could be involved in the nonlinear component of the approximation. The number  $n$  of input bits was 9, and values of  $k$  between 2 and 8 were used in experiments.

- The nonlinear equation, on  $k$  of the  $n$  input bits, was represented by its truth table (an array of 1s and 0s). As stated, this framework did not take into account the related approximations.
- The linear equation on the output bits was represented by an  $m$ -bit bitmask ( $m$  being 32 in this case), with 1s corresponding to the positions of the bits involved. Most C/C++ compilers could easily accommodate this using an unsigned long integer.
- A “projection” containing the information on which of the input bits were involved in the approximation was represented using an array of size  $k$ .

The cost function multiplied the absolute bias of the approximation by -1, and returned the result. The initial acceptance rate was set at 0.6. The move function was somewhat unusual for a simulated annealing algorithm, in that it chose one of four move types at random. Three user-supplied parameters dictated the relative probabilities of changes to the nonlinear component’s truth table, the linear component’s bitmask, and the projection as follows:

---

$0 \leq P\_NLTT \leq P\_BITMASK \leq P\_SWAP\_USED\_UNUSED \leq 1.0$

$u \leftarrow RAND(0, 1)$

**if**  $u < P\_NLTT$  **then**

A randomly chosen bit in the nonlinear component’s truth table is flipped.

**else if**  $P\_NLTT \leq u < P\_BITMASK$  **then**

The linear component’s bitmask is changed.

A new bitmask is chosen uniformly at random from the set of  $m$ -bit integers.

(This causes  $2^{m-1}$  bits in the linear component’s truth table to change.)

**else if**  $P\_BITMASK \leq u < P\_SWAP\_USED\_UNUSED$  **then**

The projection is altered.

An unused input bit replaces one of those involved in the nonlinear function.

**else**

The ordering of the bits in the projection is changed

**end if**

---

Clark et al. experimented with  $(0.25 \leq P\_NLTT \leq 0.45)$ ,  $(0.25 \leq P\_BITMASK \leq 0.45)$ , and  $P\_SWAP\_USED\_UNUSED \in \{0.5, 1.0\}$ .

For changes in the truth table of the nonlinear component, we have reason to believe a smooth search landscape is defined for the move function as described.

Let there be  $k$  bits involved in the nonlinear approximation of a single S-box  $S$ . Then, if the nonlinear component of the approximation acts on the input bits, there are  $(n - k)$  input bits not involved in it (and  $(m - k)$  if the nonlinear component acts on the output bits).

The truth table of the nonlinear component of the approximation will contain  $2^k$  entries. Let us consider a “padded” truth table for the approximation, containing the value of the nonlinear expression for every possible value of the bits at the same “end” as the nonlinear component. This truth table will contain  $2^{n-k}$  (or  $2^{m-k}$ ) copies of the truth table entry for any choice of the  $k$  involved bits. Changing one bit of the nonlinear approximation’s truth table will change the values of these copies in the padded truth table, and no other bits.

Now, let us consider the full truth table of the nonlinear approximation, containing the value of the nonlinear expression for every possible value of the S-box’s input bits. Clearly if the nonlinear expression is in terms of the input bits, this will be identical to the padded truth table. If not, we compute the full truth table using the equation  $FULL\_TRUTH\_TABLE[i] = PADDED\_TRUTH\_TABLE[S(i)]$ . For a bijective S-box, changing one bit in the basic truth table of the nonlinear component will change precisely  $2^{m-k} = 2^{n-k}$  bits in this table. For a balanced S-box with more input than output bits (such as a DES S-box), changing one bit in the basic truth table of the nonlinear component will change  $2^{n-m} \cdot 2^{m-k} = 2^{n-k}$  bits in the full truth table.

Since we can upper bound the number of changes to the truth table of the approximation's nonlinear component by  $2^{n-k}$ , which can be as low as 2 in the circumstances described if the nonlinear approximation uses  $(n-1)$  input bits, and since the truth table of the linear component does not change, none of the  $2^k$  related approximations can change bias by more than  $2^{n-k}$  when a move of this sort is made - and this acts in turn as an upper bound on changes in the absolute values of their biases.

However, for the other possible moves, experiments have shown that a smooth search landscape is not defined:

- A change to the bitmask changes the value of precisely half the bits in the linear component's truth table, meaning that for an S-box mapping  $GF(2)^n$  to  $GF(2)^m$ , the change in an approximation's bias from such a move is upper bounded by  $2^{n-1}$  - an upper bound so high as to be almost meaningless.

As an example, consider the following nonlinear approximation to DES S5. The nonlinear component of the approximation involves input bits 0, 1, 3, 4, 5 and has full truth table

```
11111111111111111111000110010001100111111111111111111111111111111111110000001000000010
```

This approximation holds with bias +22 when the linear component has bitmask 1111. However, changing the bitmask to 1101 results in its holding with bias 0, and 0 is 12 less than the smallest absolute bias of any of the original related approximations.

- Changing the order of the bits involved in the nonlinear approximation (for example, where approximation input bit  $x_{a_0}$  is  $x_0$  and  $x_{a_1}$  is  $x_1$ , exchanging their positions so that  $x_{a_0} = x_1$  and  $x_{a_1} = x_0$ ) will have no effect on the basic truth table of the approximation, but can affect several bits in the full truth table (in this case, as many as  $2^{n-1}$ ). In the case of our previous example, this causes the bias to drop from 22 to 2. Moreover, the highest absolute bias of any of the related approximations drops to 4, when the lowest had previously been 12.
- Finally, we consider swapping one of the bits involved in the nonlinear approximation with an uninvolved bit. One of the approximations related to our previous example held with bias -20. Replacing  $x_{a_1} = x_1$  with  $x_{a_1} = x_2$  changed its full truth table in 26 places and reduced this bias to 0. The largest absolute bias among any of the related approximations was now 8, 4 lower than the previous minimum.

There is evidence that this affected the behaviour of the search in Clark et al.'s experiments. In their paper, it is stated that for "almost all the executions tried" the search began with an initial period in which there was little improvement in the quality of the best approximation found, lasting approximately 500,000 moves. After this, a period of rapid and almost uninterrupted improvement began, lasting for approximately 500,000 to 700,000 moves, before the level of improvement tailed off.

We believe that the period of improvement began when the temperature of the annealing algorithm had dropped to a point where non-improving moves were very unlikely to be accepted, and that early on in this period, a sequence of moves, all acting on the truth table of the nonlinear component, were accepted. These moves increased the absolute bias of the approximation to the extent that any other sort of move was most unlikely to be accepted due to the unpredictable (but increasingly likely to be negative) effects of such moves on said value. In other words, the algorithm spent this period hill-climbing, with a move function that was limited to changing bits in the truth table of the nonlinear component, before slowing down as it approached a local optimum.

To exploit this, we significantly increase the probability of truth table changes being chosen as the move; we chose to increase probability to 0.9. Since we were attempting to find nonlinear approximations to replace the first and last round components of existing linear approximations to ciphers, the bitmask was assigned a value at the start of the search and remained static thereafter. We still allowed the search to make moves of the other two types (with probability 0.05 each) to see if it would

“home in on” particular choices of projection which would result in biases of higher magnitude than others; this was indeed the case.

We focused primarily on S-boxes from ciphers which were

1. such that linear cryptanalysis or a variant/derivative thereof has been used in a significant attack on the cipher or a reduced-round variant thereof.
2. significant, due to being or having been widely used, or being considered a viable alternative to AES, or being a promising new lightweight cipher...

The three ciphers which best satisfied both of these criteria were DES [59], PRESENT [10] and Serpent [1].

Our original experiments utilised various cost functions.

1. Let the number of related approximations be denoted  $R$ , and let  $\epsilon_i$  denote the bias of the  $i$ th related approximation ( $0 \leq i < R$ ).

We initially rewarded high values for the sums-of-squares of the biases; with the cost being

$$2^{3n-3} - \sum_{i=0}^{R-1} \epsilon_i^2$$

2. As it became apparent that the related approximations were not always statistically independent of each other; and furthermore that the sample biases in the vector  $v$  at the end of the cryptanalysis would not be either, we attempted to refine the first cost function to address this issue. The next cost function was identical to the first, but did not count the biases of related approximations with truth tables that were identical to, or bit-flips of, the truth tables of previous relateds. This was not sufficient to address the issue of statistical dependency among the linear algorithms, and we began to focus our cost functions more on the maximum bias model.
3. We rewarded the highest absolute bias:

$$cost = \left( 2^{n-1} - \max_{(0 \leq i < R)} \epsilon_i \right)$$

In situations where the maximum-bias model is used, and obtaining the  $k_1$  bits is prioritised over obtaining  $k_2$  bits, this strategy makes the most sense. In the attacks on Serpent described below, for instance, the number of  $k_2$  bits compared to the number of  $k_1$  bits was extremely small. This was also the case with the attack on DES, and no cost function used identified a nonlinear approximation such that we could be sure the correct value of  $k_2$  could be identified.

4. We attempted to obtain related approximations such that all biases were high in magnitude by rewarding high values for the smallest biases:

$$cost = \left( 2^{n-1} - \min_{(0 \leq i < R)} \epsilon_i \right)$$

This cost function, when tried, merely returned the highest bias linear approximation for the specified bitmask in all cases, suggesting (though this is not a matter of certainty) that nonlinear approximations with all relateds having bias in excess of the best linear approximation may not exist.

5. In an attempt to find cost functions suited for obtaining both  $k_1$  and  $k_2$  bits, we then tried  $cost = 2^{2n-2} - (\text{max. bias} - 2\text{nd highest bias})$ . In the case of the  $4 \times 4$  S-boxes, this did not find any nonlinear approximations that cost function 3 above had not. In the case of DES S5, which we were targeting due to its presence in the final round of Matsui’s linear attack, this

found approximations such that the maximum bias among the relateds had magnitude ranging from 18 to 22; and such that the second-highest bias was 12 lower. The bias 24 approximations found by cost function 3 were, however, considered more effective in recovering the  $k_1$  bits due to the extent to which they reduced the data complexity. It should be noted that we considered the recovery of the  $k_1$  bits to be a much higher priority than the recovery of the  $k_2$  bits; since achieving the first objective was necessary to achieve the second.

## 5 Experiments on various ciphers, and application to their crypt-analyses.

### 5.1 AES

We ran several experiments with the AES S-box and a cost function seeking to reward the maximum absolute bias of all related approximations. Using 600,000 inner loops, 500 outer loops, cooling factor 0.97 and initial acceptance rate 0.95, these yielded several approximations with absolute biases ranging from 64 to 72 (albeit with much lower-bias relateds). Some of these approximations were linear on the S-box's input bits, and some were linear on the output bits - in both cases, several approximations with bias  $\pm 72$  were obtained. For all of the input and output bitmasks involved, the bias of the best linear approximation was  $\pm 16$ .

Since all AES S-box co-ordinate functions are affine-equivalent [35], the question arises as to whether the absolute bias of the best nonlinear approximation to a Boolean function is affine invariant. While more experimentation would be needed to gain evidence for this, if true it would mean that for all input and output bitmasks, the AES S-box would have at least one nonlinear approximation with bias  $\pm 72$  (and for some bitmasks, we have already found more than one nonlinear approximation with such a bias.)

While some of the approximations with bias  $\pm 72$  were balanced, most were not. All balanced approximations found so far with absolute bias 72 have been linear on the input bits and nonlinear on the output bits, although further experimentation may yield balanced approximations with this absolute bias that are linear on the output bits.

### 5.2 Serpent.

In the literature, various linear, differential-linear, multiple linear and multidimensional linear attacks on reduced-round Serpent are described. These fall into two categories, those based on Collard et al.'s approximations [23, 24, 25, 60] and those based on the approximation of Dunkelman, Keller et al. [6, 8, 34]. In Appendix A, we point out a few errors in the descriptions of Dunkelman et al.'s approximation as found in the literature.

Of the existing attacks on reduced-round Serpent to utilise linear approximations, those described by Collard et al. included a 2R attack on 11 rounds of Serpent, utilising a 9-round approximation, and later using the new analysis phase to improve time complexity.

The same 9-round approximation was later used by Nguyen et al. [60]. In their paper, a 56-dimensional approximation to the preceding round using several input bitmasks was connected to the first round of the original approximation; yielding a 10-round multidimensional approximation which resulted in the best attacks so far against reduced-round Serpent (up to 12 rounds).

However, we believe that the data complexities of these attacks have been underestimated.

Let  $C$  denote capacity, and  $p$  the probability that the linear approximation holds (so that  $(p-1/2)$  is the bias). In Nguyen et al.'s work [60], the data complexity  $N$  specified in each case is  $4C^{-1}$ . This figure appears to have been chosen to match the values for  $N$  used by Collard et al., in which key ranking was not used and values of  $N$  equal to  $4C^{-1}$  were used in the multiple linear attacks. However, Collard et al. also used  $N$  equal to  $4 \cdot |p-1/2|^{-2}$  in the conventional linear attacks (apparently to obtain a success probability of 0.785 as predicted by Matsui [52]), and  $4C^{-1}$  is equal to  $1/(\text{sum of squares of biases})$ . To obtain internal consistency,  $N = 16C^{-1}$  would have been needed. Moreover, we have already

critiqued Table 3 of [52] in Section 3.5. For the reasons given in that section, we recalculate the expected success rates using Selçuk’s double integral [66].

Tables 8, 9, and 10 give the expected success rates for other values of  $l$ , in particular  $l = 108$ , since this is the value of  $l$  used in Collard et al.’s maximum advantage attack on 11-round Serpent [25]. We can clearly see from these that  $N = 4|p - 1/2|^{-2} = 2^{118}$  is not enough in Collard et al.’s attack on 11-round Serpent to achieve  $P_s = 0.785$ . A reasonably close probability to 0.785 may be achieved with  $N = 41.5|p - 1/2|^{-2} \approx 2^{121.375}$ , or an extremely high probability with  $N = 2^{122}$ .

For the same reasons, in Biham et al.’s linear attack on 11-round Serpent [6],  $N = 53|p - 1/2|^{-2} \approx 2^{121.728}$  is needed instead of  $N = 4|p - 1/2|^{-2} = 2^{118}$  to achieve success probability 0.785.

$N$	$8 p - 1/2 ^{-2}$	$16 p - 1/2 ^{-2}$	$17.6 p - 1/2 ^{-2}$	$32 p - 1/2 ^{-2}$
$P_s (l = 44, \text{Selçuk})$	0.028194	0.657866	0.785718	0.999875

Table 8: Success rates (calculated numerically) for  $a = l = 44$ .

$N$	$16 p - 1/2 ^{-2}$	$32 p - 1/2 ^{-2}$	$41.5 p - 1/2 ^{-2}$	$64 p - 1/2 ^{-2}$
$P_s (l = 108, \text{Selçuk})$	0.000027	0.229319	0.794093	0.999955

Table 9: Success rates (calculated numerically) for  $a = l = 108$ . We were unable to solve for a precise success rate of 0.785.

$N$	$32 p - 1/2 ^{-2}$	$53 p - 1/2 ^{-2}$	$64 p - 1/2 ^{-2}$
$P_s (l = 140, \text{Selçuk})$	0.007278	0.785316	0.986902

Table 10: Success rates (calculated numerically) for  $a = l = 140$ .

Nguyen et al.’s multidimensional “Method 2” attack on 12-round Serpent [60] aims for 172-bit maximum advantage  $a = l$  with  $M = (2^{56} - 1)$  and capacity  $C = 2^{-116}$ . Based on the discussion above, we assume that the intended probability of success  $P_s$  is 0.785. Using this information to solve Equation 3.10 and to compute  $N$  from  $b$ , we obtain a data complexity of  $N \approx 2^{124.39}$ .

The “Method 1”-based attack from the same paper is not so easy to estimate data complexity for, since it consists of  $2^{128}$  separate 1R attacks with key guessing on 48 bits in the final round. We can assume that the data complexity for one such 1R attack must lower-bound the value of  $N$  in this case, but we believe that it must be an underestimate. Using the same methodology as before, for a capacity of  $2^{-114}$ , we obtain  $N \geq \approx 2^{121.275}$ .

If differences between the actual and theoretical distributions resulting from the linear hull effect are not too significant, this is still the best attack on reduced-round Serpent to date. However, as the effectiveness of LLR-based nonlinear and multidimensional linear attacks has not to our knowledge been experimentally tested for as many as 12 rounds - or indeed as many as 11 - we are forced to express some doubt as to whether the attack can succeed with the data complexity claimed. This would be a matter for future research.

If we attempt to address this issue by carrying out the attack using the  $\chi^2$  statistic instead, then according to the formula given in subsection 3.3, if we use the entire codebook of  $2^{128}$  known plaintexts, we obtain an advantage of  $\approx 0.279$ . Since the attack is against 256-bit Serpent, this gives the search phase a time complexity of  $\approx 2^{255.721}$ . This dominates the complexity of the attack, giving us an approximate overall time complexity of  $\approx 2^{255.721}$ . Since the success probability is 0.785, and since an exhaustive search of 78.5% of the keyspace would have slightly lower time complexity  $\approx 2^{255.651}$ , we are not sure that the  $\chi^2$  attack could reasonably be viewed as an attack under these circumstances.

The same paper’s attacks on 11-round Serpent also underestimate the data complexity, and rely on the LLR working as predicted. In the case of the attack with twelve active S-boxes in the final round, 48-bit advantage is the implied aim since key ranking is not used. We solve Equation 3.10 for  $M = (2^{56} - 1)$ , capacity  $2^{-114}$  and  $P_s = 0.785$ , and obtain  $N \approx 2^{121.275}$ .

In the case of the attack with eleven active final-round S-boxes, we use the same methodology and obtain  $N \approx 2^{123.219}$ .

(Both of these figures depend on the LLR-statistic remaining usable in spite of the linear hull effect after 11 rounds. If this is not the case, since the same linear characteristic is used as in the case of the 12-round attacks, we still obtain advantage of only  $\approx 0.279$  and resultant time complexity  $\approx 2^{255.721}$  when using the  $\chi^2$  statistic instead - and for the same reasons as before, this probably cannot be considered to constitute an attack.)

If the LLR statistic is used, we assume that the convolution method [38] is used to minimise the complexity of converting the empirical distributions into scores for the various key candidates. The time complexity is still non-negligible compared to the remainder of the attack, being equal to  $2^k((6m+13)\cdot 2^m)$  MAs +  $2^k((6m+4)\cdot 2^m)$  AOs.

As we see from tables 11, 12 and 13 below, the best *existing* attacks on eleven-round Serpent in terms of data and memory complexity are those of Nguyen et al. Time complexity depends on which of the Serpent key lengths is in use; for the 192 and 256-bit keys, Collard et al. have less key bits remaining to search for and achieve the best time complexity of the existing methods; for the 128-bit key length the faster analysis phase, and reduced time required to encrypt the known plaintexts, of Nguyen et al.'s method dominates the time complexity and makes it the superior attack. The complexity of the search phase gives the nonlinear attack in this paper the best overall time complexity for the case of 11-round Serpent with 256-bit keys, and it may also be seen that nonlinear cryptanalysis achieves better data complexity than any other known-plaintext - or indeed chosen plaintext - attack on 11-round Serpent with 192 or 256-bit keys.

Rounds	Type of attack	Data	Time (analysis)
11	Linear [6]	$2^{121.728}$ KP	$2^{188.1}$ E
11	Linear [6]	$2^{121.728}$ KP	$2^{96}$ PE + $2^{44}$ PD + $2^{149.73}$ AO + $2^{149.76}$ MA
11	Linear [25]	$2^{121.375}$ KP	$2^{60}$ PE + $2^{48}$ PD + $2^{117.36}$ AO + $2^{117.4}$ MA
11	<b>Multidim. lin.</b> [60]	$2^{121.275}$ KP	$2^{48}$ PD + $2^{114.087}$ AO + $2^{114.134}$ MA
11	<b>Multidim. lin.</b> [60]	$2^{123.219}$ KP	$2^{44}$ PD + $2^{110.055}$ AO + $2^{110.103}$ MA
11	Differential-linear [34]	$2^{121.8}$ CP	$2^{135.7}$ MA
11	Nonlinear (this paper)	$2^{120.467}$ KP	$2^{80}$ PE + $2^{48}$ PD + $2^{139.6}$ AO + $2^{139.63}$ MA
11	Nonlinear (this paper)	$2^{117.401}$ KP	$2^{60}$ PE + $2^{76}$ PD + $2^{149.69}$ AO + $2^{149.72}$ MA
11	Nonlinear (this paper)	$2^{115.44}$ KP	$2^{60}$ PE + $2^{80}$ PD + $2^{153.73}$ AO + $2^{153.76}$ MA
11	Differential-linear [34]	$2^{113.7}$ CC	$2^{137.7}$ MA
12	Differential-linear [34]	$2^{123.5}$ CP	$2^{249.4}$ E
12	<b>Multidim. lin.</b> [60]	$2^{124.39}$ KP	$2^{128}$ PE + $2^{44}$ PD + $2^{238.744}$ AO + $2^{238.769}$ MA
12	<b>Multidim. lin.</b> [60]	$\geq 2^{121.275}$ KP	$2^{128}$ PE + $2^{48}$ PD + $2^{242.087}$ AO + $2^{242.134}$ MA

Table 11: 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 analysis method [6] is. Table entries in bold signify that the method may not work as claimed depending on the linear hull effect and how it affects the LLR statistic. E = full encryptions of the reduced round cipher. PE = partial encryptions. PD = partial decryptions. AO = arithmetic operations. KP = known plaintexts. CP = chosen plaintexts. CC = chosen ciphertexts.

Rounds	Type of attack	Time (analysis) summary	Mem	Bits recovered
11	Linear [6]	$2^{188.1}$ E	*	140
11	Linear [6]	$2^{137.08}$ E + $2^{149.76}$ MA	$2^{144.087}$	140
11	Linear [25]	$2^{104.71}$ E + $2^{117.4}$ MA	$2^{112.087}$	108
11	<b>Multidim. linear</b> [60]	$2^{101.437}$ E + $2^{114.134}$ MA	$2^{108}$	48
11	<b>Multidim. linear</b> [60]	$2^{97.405}$ E + $2^{110.103}$ MA	$2^{104}$	44
11	Differential-linear [34]	$2^{135.7}$ MA	$2^{76}$	48
11	Nonlinear (this paper)	$2^{126.95}$ E + $2^{139.63}$ MA	$2^{134.087}$	128 $k_1$ , 2 $k_2$
11	Nonlinear (this paper)	$2^{137.04}$ E + $2^{149.72}$ MA	$2^{144.087}$	136 $k_1$ , 4 $k_2$
11	Nonlinear (this paper)	$2^{141.08}$ E + $2^{153.76}$ MA	$2^{148.087}$	140 $k_1$ , 4 $k_2$
11	Differential-linear [34]	$2^{137.7}$ MA	$2^{99}$	60
12	Differential-linear [34]	$2^{249.4}$ E	$2^{128.5}$	160
12	<b>Multidim. linear</b> [60]	$2^{225.964}$ E + $2^{238.769}$ MA	$2^{232}$	172
12	<b>Multidim. linear</b> [60]	$2^{229.437}$ E + $2^{242.134}$ MA	$2^{108}$	176

Table 12: Attack complexities cont. All memory complexities are measured in bytes. The time and memory complexities for Biham et al.’s linear cryptanalysis vary depending on whether the new analysis method of Collard et al. is used, or whether an earlier analysis method [6] is. In the latter case, the relevant sources [6, 25] disagree as to the memory complexity. Based on the bitsliced Serpent implementation and Osvik’s new implementation of S6 [1, 64] we estimate  $2^{12.65}$  AOs are needed for an 11-round Serpent encryption, ignoring the key schedule as this is only done once, and  $2^{12.78}$  AOs for 12-round Serpent. E = full encryptions of the reduced round cipher. PE = partial encryptions. PD = partial decryptions. KP = known plaintexts. CP = chosen plaintexts. CC = chosen ciphertexts.

		Bits remaining		
Rounds	Type of attack	(128-bit key)	(192-bit key)	(256-bit key)
11	Linear [6]	N/A	52	116
11	Linear [6]	N/A	52	116
11	Linear [25]	20	84	148
11	<b>Multidim. linear</b> [60]	80	144	208
11	<b>Multidim. linear</b> [60]	84	148	212
11	Differential-linear [34]	80	144	208
11	Nonlinear (this paper)	N/A	62	126
11	Nonlinear (this paper)	N/A	52	116
11	Nonlinear (this paper)	N/A	48	112
11	Differential-linear [34]	68	132	196
12	Differential-linear [34]	N/A	32	66
12	<b>Multidim. linear</b> [60] (Method 2)	N/A	20	84
12	<b>Multidim. linear</b> [60] (Method 1)	N/A	16	80

Table 13: Complexities for attack when  $P_s = 0.785$  (or slightly higher) cont.

### 5.2.1 Using nonlinear attacks to reduce the data complexity of attacking 11-round Serpent-192 and Serpent-256.

It is not clear how, if it is possible at all, to combine nonlinear and multidimensional linear approximations. We therefore focus on modifying Collard et al.’s “Approximation D2”, and focus on the version with 12 active S-boxes in the final round.)

The simplest change possible is to replace the (input bitmask 12, output bitmask 10) bias 4 approximation in the first round (affecting bits 16, 17, 18, 19) with the following approximation:

$$x_2 \oplus x_1 \oplus x_1x_4 = y_1 \oplus y_3$$

The primary approximation has bias 6, and after we eliminate related approximations which are bit-flips of others, we obtain sum-of-squares-of-biases 40. Fortunately, the related approximations are uncorrelated and we obtain the full corresponding increase in capacity should we choose to use the  $\chi^2$  model.

20 instead of 15 S-boxes are now activated in the plaintext, increasing the number of  $k_1$  key bits attacked to 128. The memory requirements are increased to  $2^{(128+2)+4.087} = 2^{134.087}$  bytes, due both to the extra  $k_1$  bits and the four related approximations. The time complexity of the analysis phase also increases, and is dominated by the  $4 \cdot (6 \times 128 + 8) \cdot 2^{128} = 2^{139.6}$  arithmetic operations and  $4 \cdot (6 \times 128 + 26) \cdot 2^{128} = 2^{139.63}$  memory accesses (The multiplication by 4 results from there being four “relateds”).

Despite the aforementioned difficulty in comparing memory access complexity to complexity in terms of encryptions, we are able to calculate an estimate for the number of arithmetic operations per reduced-round encryption, by counting the number of operations involved in the optimised “bitslice” implementation of Serpent [1]. In particular, this implementation does not use lookup tables for the S-boxes, but instead uses arithmetic operations to calculate the output values extremely quickly.

If we obtain an optimistic estimate for the number of arithmetic operations per reduced-round Serpent encryption, dividing the attack’s AO complexity by this figure will give us a conservative estimate for its time complexity. For this reason, we base our estimate on a version of Serpent in which Osvik’s implementation of S6 [64] has replaced the original implementation, allowing one less AO per calculation of S6, and assume that the performance gains of the bitslice implementation are not compromised by this. If future research should provide evidence that this is not in fact possible, we can easily base new estimates on the original version.

(Note that operations such as bitwise xor, which might more often be described as logical operations, are included under the banner of “arithmetic operations” in this case.)

This gives us  $2^{12.65}$  AOs per 11-round Serpent encryption, and  $2^{12.78}$  per 12-round encryption. Dividing the appropriate figure by  $2^{12.65}$ , we obtain time complexity of  $2^{126.95}$  encryptions +  $2^{139.63}$  MAs.

In the  $\chi^2$  model, the capacity of the new approximation is equal to  $2.5 \times$  what it was before, however the increased number of degrees of freedom (4 instead of 1) means that we cannot reduce the data requirements a full 2.5-fold. The number of degrees of freedom is too low for us to use Equation 3.3, which in any case heavily underestimates the advantage of the original attack; however if it can be taken as a guide, it indicates that we achieve the same advantage with  $2^{120.875}$  known plaintexts instead of  $2^{121.375}$ .

If we use the maximum-bias approach instead, the capacity is multiplied by  $(6/4)^2 = 2.25$ . However, we cannot decrease the known-plaintext requirements 2.25-fold, since the increased number of  $k_1$  bits, and the need to deal with  $2^2$  relateds per outer key guess, effectively raises  $l$  to 130. To obtain success probability close to the 0.794 of the original attack, a higher value of  $N|p - 1/2|^{-2}$  is needed. 49.75 instead of the previous 41.5 gives us success probability 0.8, and means that  $N$  is in fact reduced by a factor of 1.877, to  $2^{120.467}$ . This is clearly a better option than using the  $\chi^2$  statistic.

This approximation involves three  $k_2$  bits. Due to the bit-flipped relateds, we can only recover two of these; the bits corresponding to  $x_1$  and  $x_4$ .

There is another bias 4 linear approximation in the first round, and several approximations in the final round with bias  $\pm 4$ , that can be replaced with nonlinear approximations possessing similar properties to the one above. Let us consider a situation in which:

- the entire first round approximation remains linear,
- we replace the final-round S-box approximation  $x_3 \oplus x_4 = y_4$  (bias 4; affecting state bits 76 to 79) with  $x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_1 y_3$ . This approximation has bias 6, and a statistically independent related with absolute bias 2. All other relateds either have zero bias or are bit-flips of these, so we have sum of squares of biases 40.

This increases the number of active final-round S-boxes from 12 to 17.

- we also replace one of the final-round  $x_1 \oplus x_3 \oplus x_4 = y_2$  approximations (bias 2; the one affecting state bits 96 to 99) with  $x_1 \oplus x_3 \oplus x_4 = y_2 \oplus y_1 \oplus y_2 y_4$ . The number of active final-round S-boxes increases again, from 17 to 19.

The total number of active S-boxes increases from 27 to 34.

We have replaced a bias 4 ( $\text{bias}^2 = 16$ ) approximation and a bias 2 ( $\text{bias}^2 = 4$ ) approximation with two nonlinear approximations, each being such that the primary approximation has bias 6, and such that the sum of squares of statistically independent biases is equal to 40.

Let us first consider the  $\chi^2$  model. In this model, the capacity is multiplied by  $(2.5 \times 10) = 25$ . Evidence from experiments on an SPN-based cipher in which final-round linear approximations with bias  $\pm 4$  were replaced with nonlinear approximations with identical properties to the ones above suggests that a 6.25-fold increase in capacity, mitigated by an increase in the number of degrees of freedom from 1 to 16, results in a reduction in data complexity by a factor of approximately  $2^1$ . This would lead to an estimated  $2^{120.375}$  known-plaintext requirement. Since we have a further  $2^2$ -fold increase in capacity on top of this, we estimate that  $2^{118.375}$  known plaintexts are required, and that the data requirements for the same advantage as Collard et al.’s original attack are very unlikely to be  $\geq 2^{119.375}$ . However, these experiments used a smaller value of  $l$ , and due to the low number of degrees of freedom, it is not clear how much confidence we can place in these figures.

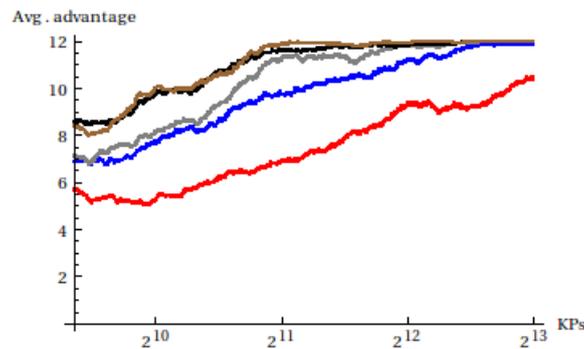


Figure 11: Graph showing mean advantages for attack on four round SPN with  $4 \times 4$  S-boxes using: linear approximation (red), nonlinear approximation (Two final-round S-boxes are approximated with “6, 2, bit-flips” approximations of the type used in this section) in  $\chi^2$  model (blue), same nonlinear approximation in maximum-bias model (grey), multiple nonlinear in  $\chi^2$  model with two sets of approximations of this type (black), and multiple nonlinear with same two approximations in maximum-bias model (brown).

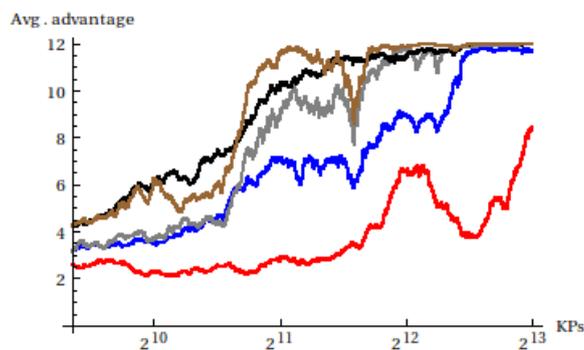


Figure 12: Graph showing alternate calculation for average advantage in which the mean rank obtained was input to the formula for advantage.

If, by contrast, we utilise the maximum-bias model, we replace one bias 4 approximation and

one bias 2 approximation with two bias 6 approximations, multiplying capacity by  $\left(\frac{6 \times 6}{4 \times 2}\right)^2 = 20.25$ . Since the value of  $l$  is effectively increased to 140, this does not simply reduce the known-plaintext requirements to  $2^{117.035}$ , since we have to increase  $N|p - 1/2|^2$  to compensate.  $N = 53.5|p - 1/2|^{-2}$  gives success probability 0.8, and  $N = 2^{117.401}$ . The memory requirements are increased to  $2^{144.087}$ . 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  $\approx 2^{137.04}$  11-round encryptions.

To reduce the number of known plaintexts further, we could replace another of the  $x_1 \oplus x_3 \oplus x_4 = y_2$  approximations with a nonlinear approximation instead of replacing  $x_3 \oplus x_4 = y_4$ . If we choose the approximation affecting state bits 116-119, we can do this with a total of 35 S-boxes activated, and 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  $\approx 2^{141.08}$  eleven-round encryptions with memory complexity  $2^{148.087}$ . Estimated data complexity in the  $\chi^2$  model is  $2^{116.375}$ , but for the reasons given above we view complexity calculations as more reliable in the maximum-bias model.  $l$  is in effect increased to 144, resulting in  $N$  having to equal  $55|p - 1/2|^{-2}$  to obtain success probability 0.8 with  $N = 2^{115.44}$ .

(In all three of the above attacks, the improvements in data complexity that could potentially arise from using the LLR were extremely slight, and due to our doubts regarding this statistic and the linear hull effect, we did not consider it worth including results for it in this section's tables.)

### 5.2.2 Improving the capacity of the highest-bias approximation of nine rounds of Serpent.

The description of Collard et al.'s approximations [24] includes one of several nine-round linear approximations discovered with bias  $2^{-50}$  (capacity  $2^{-98}$ ); the highest bias achieved for a linear approximation of that many rounds. However, none of these approximations are used in attacks, since the high number of active S-boxes in each would result in attacks with far higher complexity than the then-current state of the art.

Our algorithm found several higher-capacity replacements for linear S-box approximations in the outer rounds (both of which used Serpent S3). These included various approximations in which the various "relateds" were all either:

1. statistically independent, or
2. bit-flips of other relateds, which could safely be ignored

allowing us to calculate the new capacity precisely:

- The first round - nonlinear components in the input bits.
  - We can replace the bias  $-4$  (capacity 64) linear approximation  $x_1 \oplus x_4 = y_2 \oplus y_4$  (input bitmask 9, output bitmask 5) with one of the following nonlinear approximations:
    1.  $x_2 \oplus x_1 x_4 \oplus x_1 x_2 = y_2 \oplus y_4$  and relateds. (Primary approximation has bias  $+6$ , we can choose a statistically-independent related with bias either 2 or  $-2$ , other relateds either have bias 0 or are bit-flips of these.)
    2.  $x_4 \oplus x_1 \oplus x_2 x_4 = y_2 \oplus y_4$  and relateds. (Primary approximation has bias  $-6$ . Again, we can choose a statistically-independent related with bias either 2 or  $-2$ , and the other relateds either have bias 0 or are bit-flips of these.)
    3. Other nonlinear approximations such that one related has bias  $\pm 6$  exist, but the truth tables of the related approximations are not statistically independent, so we are unable to calculate their capacity when working in the  $\chi^2$  model. In experiments on a toy cipher, these appear to have approximately the same capacity, but since optimisations to omit bit-flips and zero-bias relateds cannot be made, they are also much slower to work with.

Table 14 summarises the above:

Nonlinear component	Bias	(2, 4) wrong	(1) wrong	(1, 2, 4) wrong
$x_2 \oplus x_1x_4 \oplus x_1x_2$	+6	-6	-2	+2
$x_4 \oplus x_1 \oplus x_2x_4$	-6	+2	+6	-2

Table 14: Nonlinear approximations to S3 with output bitmask 0101.

- We can also replace the bias 4 (capacity 64) linear approximation  $x_2 \oplus x_3 = y_1 \oplus y_2$  (input bitmask 6, output bitmask 12) with one of various nonlinear approximations with very similar properties to those found in the above case:

1.  $x_2 \oplus x_3x_4 = y_1 \oplus y_2$  and relateds. (Primary approximation has bias +6, we choose an independent related with bias either 2 or -2, all other relateds have either bias 0 or are bit-flips of the preceding.)
2.  $x_3 \oplus x_2x_4 = y_1 \oplus y_2$  and relateds. (Primary approximation has bias +6, again we choose an independent related with bias either 2 or -2, and all others have zero bias or are bit-flips of the preceding two.)

As before, other nonlinear approximations with bias  $\pm 6$  primary approximations but statistically dependent relateds also exist.

Nonlinear component	Bias	(2) wrong	(3) wrong	(2, 3) wrong
$x_2 \oplus x_3x_4$	+6	-6	-2	+2
$x_3 \oplus x_2x_4$	+6	-2	-6	+2

Table 15: Nonlinear approximations to S3 with output bitmask 1100.

- The final round - nonlinear components in the output bits.
  - We can replace the bias 4 linear approximation  $x_3 \oplus x_4 = y_4$  with one of two nonlinear approximations with similar properties to those presented above. These are:  $x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_1y_3$  and  $x_3 \oplus x_4 = y_4 \oplus y_3 \oplus y_3y_4 \oplus y_1y_4 \oplus y_1y_3$ .

Nonlinear component	Bias	(4) wrong	(1, 3, 4) wrong	(1, 3) wrong
$y_4 \oplus y_3 \oplus y_1y_3$	+6	-6	-2	+2
$y_4 \oplus y_3 \oplus y_3y_4 \oplus y_1y_4 \oplus y_1y_3$	+6	-2	-6	+2

Table 16: Nonlinear approximations to S3 with input bitmask 0011.

This pattern occurs fairly frequently.

- The bias 4 approximation  $x_1 \oplus x_3 \oplus x_4 = y_1$  (which occurs three times) can be replaced with one of the *four* approximations in Tables 17 and 18:

Nonlinear component	Bias	(1) wrong	(1, 2) wrong	(2) wrong
$y_2 \oplus y_1 \oplus y_2y_4$	+6	-6	-2	+2
$y_2 \oplus y_2y_4 \oplus y_1y_4$	+6	-2	-6	+2

Table 17: First set of nonlinear approximations to S3 with input bitmask 1011.

Nonlinear component	Bias	(1) wrong	(1, 3) wrong	(3) wrong
$y_4 \oplus y_3 \oplus y_1 \oplus y_3y_4$	-6	+6	+2	-2
$y_3 \oplus y_3y_4 \oplus y_1y_4$	+6	-2	-6	+2

Table 18: Second set of nonlinear approximations to S3 with input bitmask 1011.

- The bias  $-4$  approximation  $x_1 \oplus x_2 \oplus x_3 = y_3$  can be replaced with either of the two approximations in Table 19 (both capacity 160):

Nonlinear component	Bias	(1, 3, 4) wrong	(1, 3) wrong	(4) wrong
$y_4 \oplus y_1 \oplus y_3y_4 \oplus y_1y_4 \oplus y_1y_3$	+6	-6	-2	+2
$y_4 \oplus y_1 \oplus y_3y_4 \oplus y_1y_4$	+6	-2	-6	+2

Table 19: Nonlinear approximations to S3 with input bitmask 1110.

Although we are likely to encounter the same issues with increased TPS size and time complexity of handling the relateds as before, in the maximum-bias model this gives us several nonlinear approximations to nine-round Serpent with bias  $\pm 2^{-45.9}$  instead of  $2^{-50}$ . In the  $\chi^2$  model, we can replace the highest-capacity approximation to 9-round Serpent known so far (capacity  $4 * (2^{-50})^2 = 2^{-98}$ ) with several different approximations with capacity  $\approx 2^{-88.75}$ .

This is unlikely to be of use in practice - the original nine-round linear approximation had too many active S-boxes in the plaintext and ciphertext to be used in a feasible attack, and this approximation only exacerbates the same problem. We include it here merely to demonstrate the potential nonlinear approximations to cipher rounds have to increase bias and capacity.

### 5.3 DES

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2 bits in NL component	4	8	8	8	6	8	10	6	8	8	8	6	10	12	18
3 bits in NL component	10	10	12	8	14	10	14	10	10	12	12	10	12	16	24
4 bits in NL component	14	14	14	14	14	14	20	16	14	14	18	14	18	18	24
5 bits in NL component	14	14	18	16	16	18	22	16	16	16	20	16	22	22	28
Best linear approx.	14	12	8	10	10	12	12	14	8	12	12	12	10	12	18

Table 20: DES S1. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds.

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2 bits in NL component	4	6	8	4	8	8	10	4	6	8	16	8	12	10	14
3 bits in NL component	10	10	12	14	12	10	12	8	10	10	16	10	16	14	22
4 bits in NL component	14	12	14	16	14	12	20	10	12	14	18	16	20	18	22
5 bits in NL component	16	14	16	18	20	16	24	14	18	18	22	20	22	22	24
Best linear approx.	10	12	10	14	10	8	10	14	12	10	16	10	12	10	12

Table 21: DES S2. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds.

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2 bits in NL component	4	8	8	4	10	8	8	4	8	8	12	8	14	10	16
3 bits in NL component	8	10	12	8	14	12	14	6	10	12	16	12	18	14	18
4 bits in NL component	12	12	16	12	16	16	16	10	14	16	18	14	18	20	20
5 bits in NL component	14	16	18	14	22	22	20	14	18	20	22	20	24	22	22
Best linear approx.	14	10	12	12	10	12	12	14	12	10	12	10	14	10	16

Table 22: DES S3. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds.

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2 bits in NL component	4	4	8	4	8	8	12	4	8	8	10	8	10	12	16
3 bits in NL component	10	8	8	8	12	8	16	10	8	12	16	8	16	16	16
4 bits in NL component	12	12	16	12	12	16	18	12	16	12	18	16	18	18	24
5 bits in NL component	16	16	20	16	16	20	22	16	20	16	22	20	22	22	32
Best linear approx.	10	10	12	10	12	16	10	10	16	12	10	12	10	10	16

Table 23: DES S4. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds. Note in particular that, for bitmask 15, maximum bias of 32 was achieved - the xor of the four output bits of DES S4 is independent of the sixth input bit.

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2 bits in NL component	6	6	6	4	10	6	10	6	6	10	12	8	14	16	20
3 bits in NL component	6	12	8	6	12	10	16	8	10	10	14	10	14	16	20
4 bits in NL component	10	12	14	10	14	14	18	12	14	14	18	14	18	18	24
5 bits in NL component	16	18	16	12	16	16	20	14	16	18	20	18	22	22	24
Best linear approx.	10	12	10	14	10	8	10	12	10	12	12	10	14	16	20

Table 24: DES S5. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds. Note that for bitmask 4 the best linear approximation has higher magnitude of bias than any of our nonlinears. The linear function on the input bits involves all six  $x_i$ , whereas our nonlinear approximations were limited to five to reflect the fact that if all six  $x_i$  were exposed to the cryptanalyst, there would be no need to use an approximation.

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2 bits in NL component	4	6	6	4	6	8	14	6	8	8	14	8	12	12	12
3 bits in NL component	12	8	10	8	10	12	18	6	12	10	14	12	14	18	14
4 bits in NL component	12	10	12	10	14	14	20	12	16	16	16	16	18	18	18
5 bits in NL component	16	14	20	16	16	16	24	14	18	20	22	22	22	22	20
Best linear approx.	12	12	10	12	10	10	14	12	8	10	14	12	12	12	12

Table 25: DES S6. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds.

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2 bits in NL component	4	8	10	8	10	6	10	6	8	10	14	10	14	16	14
3 bits in NL component	6	8	12	12	14	10	12	8	12	12	16	10	14	16	16
4 bits in NL component	12	12	16	14	16	14	18	12	16	14	20	14	18	20	20
5 bits in NL component	14	18	20	18	20	20	20	14	18	18	22	20	22	22	24
Best linear approx.	14	10	10	18	10	10	12	12	8	12	14	12	14	16	14

Table 26: DES S7. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds.

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2 bits in NL component	4	6	8	4	8	6	8	6	12	8	12	6	12	16	16
3 bits in NL component	8	10	12	12	10	10	16	10	14	12	14	10	16	16	24
4 bits in NL component	14	12	14	12	12	14	18	10	14	16	18	16	18	22	28
5 bits in NL component	16	18	18	20	18	16	20	16	18	18	20	18	22	24	28
Best linear approx.	10	12	12	12	10	10	14	10	12	10	12	10	12	16	16

Table 27: DES S8. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds.

### 5.3.1 The approximation on bits 1, 2, 5, 6

As stated, since one of the four key bits at the input to DES S5 in round 15 is active in round 16, we have  $|k_2| = 3$ . The main approximation has bias 24, and the relateds corresponding to wrong key guesses for the three undetermined bits have bias 16 (in three cases) and 12 (four cases).

- If the  $\chi^2$ -statistic is used, then the number  $M$  of degrees of freedom for the new attack is equal to  $2^{|k_2|}$ , which for this attack is 8. We cannot re-guess the key bit that was active in Round 16 to take advantage of the biases of the incorrect relateds, since each of these has a truth table obtained by flipping all the bits in the truth table of one of the relateds for the correct value of this key bit; and hence they provide us with no additional information.
- The complexity of the analysis stage would be dominated by:
  - $2^6$  PEs, each with complexity 1/128 that of a full encryption.
  - $2^{24}$  PDs, each with complexity 12/128 that of a full encryption. (The total complexity of the PEs and PDs so far equates to  $2^{20.585}$  DES encryptions.)
  - $(6 \times 35 + 8) \times 2^{35} = 2^{42.77}$  arithmetic operations.
  - $(6 \times 35 + 26) \times 2^{35} = 2^{42.88}$  memory accesses.

Equating complexity in terms of arithmetic operations to an estimated complexity in terms of DES encryptions is much more difficult than in the case of Serpent. If we treat the number of gate operations as equivalent to the number of AOs, Kwan’s best figures for bitsliced DES [48] give us a total of 6528 AOs for the S-boxes. Biham [5] claims that we need not treat the DES expansion and permutation as requiring any operations in a bitsliced implementation, that the key xor requires 48 operations per round, and the xor of the round function outputs with the left block requires 32.  $(32 + 48) \times 16 = 1280$ . Reference is also made to 160 CPU load/store instructions per round; due to the small amount of data involved it may be possible to keep these in cache memory, but they clearly complicate the issue. Kwan also notes [49] improved bitslice S-boxes by the developers of the “John the Ripper” password cracking software, which depending on the CPU architecture may be able to use as few as 4208 AOs instead of 6528.

- An additional  $2^{35}$  time would then be required to go through the set of results and eliminate all values corresponding to incorrect values of the dummy key bits. If we count this as part of the analysis phase, its complexity is expected to be negligible compared to the above.
- Since seven of the bits of  $k_1$  were dummies, there would be 28 key bits remaining to handle during the search phase. There are also 28 non-dummy bits in  $k_1$ . If we seek to obtain the same advantage as Matsui's linear attack ( $a = 13$ ), then we would need to use key-ranking with the  $X = 2^{28-13} = 2^{15}$  highest-scoring keys, and the search phase would have complexity  $O(28 \cdot 2^{28})$  to sort the results, plus  $(2^{15} \cdot 2^{28}) \approx 2^{43}$  DES encryptions.

The complexity of the distillation phase is dependent on the change in data complexity. The various related approximations involved are all statistically dependent, with pairwise correlation coefficients of 0.5, and we do not currently have a statistical model or empirical evidence for the effect this would have on the capacity when using the  $\chi^2$ -statistic. We therefore use the maximum-bias model, noting that we may not have sufficient data to deduce bits of  $k_2$  due to the high bias of the relateds involved. Although it is not clear precisely how the time complexity so far compares with Matsui's original attack, we will now see that despite the improved bias of the approximation, the data complexity has *worsened!*

It seems as though the data complexity should be reduced by a factor of  $24^2/20^2 = 1.44 = 2^{0.526}$ , giving overall data complexity, and time complexity for this phase, of  $2^{43-0.526} = 2^{42.474}$ . Unfortunately, this is not the case. Matsui's attack does in fact consist of two separate attacks with  $a = 6.5$ , combined to produce one attack with  $a = 13$ . This low advantage allows Matsui to use less data than would be the case for a direct advantage 13. We can obtain  $P_s = 0.85$  for  $a = 6.5, N = 2^{42.5}$ , but not with  $a = 13$  [66]. The reduced advantage would massively increase the attack's time complexity; still breaking DES but much more slowly than Matsui.

### 5.3.2 The approximations on bits 1, 2, 4, 5, 6

To use these approximations, it is necessary to guess thirty key bits in the final round (for (S1, S3, S4, S6, S8)). The six active key bits in Round 1 now include three guessed bits and three dummy bits, as one of them is input to S6 in round 16. We need five instead of four dummy key bits for the left-hand ciphertext block now, raising the value of  $|k_1|$  to 41. The complexity of a partial encryption/decryption increases to 6/128 that of a full DES encryption, *plus* the time required to compute the truth tables of the nonlinear functions on bits 1, 2, 4, 5 and 6.  $|k_2|$  is reduced to 1, as only one of the five active key bits at the input to S5 in Round 15 is now not active in Round 16. However, we can increase it by re-guessing Round 16 key bits. The minimum possible complexity of a partial encryption/decryption is, therefore, equal to  $((6 + 2^1)/128) = (1/16)$  of a full DES encryption

The complexity of the analysis phase is, as a result of this, now at least

- $2^{30}$  PDs, each with complexity 7/128 that of a full encryption.
- $2^6$  PEs, each with complexity 1/128 that of a full encryption. (The total complexity of the PEs and PDs so far equates to  $2^{25.8}$  DES encryptions.)
- $(6 \times 41 + 8) \times 2^{41} = 2^{49}$  arithmetic operations.
- $(6 \times 41 + 26) \times 2^{41} = 2^{49.09}$  memory accesses.

In terms of time complexity, this attack is clearly inferior to its predecessor. As for data complexity, the maximum-bias model only equals the attack described above, and the level of statistical dependence among the related approximations means that we do not currently know what its data complexity in the  $\chi^2$  model would be.

## 5.4 PRESENT

Currently, the largest number of rounds of PRESENT attacked is 26 [13, 14]. This (slightly controversial [39]) attack utilises a new form of multidimensional linear cryptanalysis that relies heavily on the existence of multiple linear paths with the same bias.

We note that this paper’s formula for the data complexity is almost identical to Hermelin et al.’s Theorem 1 [17], except that it incorporates the assumption that  $a \approx b^2$ , and replaces the value  $4M$  with  $8M$ .

We have already criticised the original formula and the  $a \approx b^2$  assumption; however the replacement of  $4M$  with  $8M$  in the (denominator) of 3.1, resulting in

$$a \approx \frac{(NC(p) - 4\Phi^{-2}(2P_S - 1))^2}{8M} \quad (5.1)$$

is a new development, which the author does not explain. Although this is referred to elsewhere [39] as a typographical error; in emails the author has stated that he believes the  $8M$  version to be correct, and has introduced the change to bring the theoretical formula more closely into line with empirical evidence for the behaviour of the attack.

Unfortunately, for the 1-bit input/output bitmasks of the S-boxes in the outer rounds of the approximation, we have not been able to find any nonlinear approximations with sufficient capacity and lack of dependence among the related approximations to improve on this attack, nor on its (less controversial) predecessor [63]. Nonlinear approximations for the PRESENT S-box with higher magnitude of bias than linear approximations do exist, but not for the bitmasks required to attach them to the linear approximation involved in this attack.

	Bitmask for linear function of input bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Max bias (NL approx.)	4	4	6	4	4	4	6	4	6	4	8	4	6	4	6
Best linear approx.	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Table 28: PRESENT S-box. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds (3 bits in nonlinear component).

	Bitmask for linear function of output bits.														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Max bias (NL approx.)	4	4	6	4	6	4	4	4	6	4	6	4	6	6	6
Best linear approx.	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Table 29: PRESENT S-box. Maximum (absolute) bias found for cost function rewarding maximum bias among relateds (3 bits in nonlinear component).

## 5.5 Heys

Treating the four  $4 \times 4$  S-boxes per round of the Heys toy cipher as two  $8 \times 8$  S-boxes (each mapping the concatenation of two 4-bit inputs to the concatenation of two 4-bit outputs), we tried to find out if there existed nonlinear approximations to these conjoined boxes which could not be decomposed into expressions of the form (nonlinear approximation for first box)  $\oplus$  (nonlinear approximation for second box), and which had nonzero bias despite the independence of the two  $4 \times 4$  S-boxes.

We did indeed find such approximations, some of which also had biases which could not have resulted from applying the Piling-Up Lemma to two separate nonlinear approximations to the  $4 \times 4$  boxes. For example,  $1 + x_1 + x_3 + x_5 + x_1x_2x_3x_4x_5 = y_2 + y_3 + y_4 + y_5 + y_8$  had a bias of 68. If this bias could occur from applying the Piling-Up Lemma to two separate approximations with biases

$(a - 8)/16$ ,  $(b - 8)/16$ , then  $68/256$  would be expressible as  $(ab + (16 - a)(16 - b) - 128)/256$ , and there exists no pair of positive integers  $(a, b)$  such that this is possible.

## 6 Conclusions, and directions for future research

In this chapter, we have evolved nonlinear approximations for block cipher S-boxes with higher absolute bias than the best-known linear approximations for said boxes. Prior to doing this, we have designed new algorithms which would be able to use these new forms of approximation in attacks, and devised the statistical frameworks allowing us to calculate the attack complexities, before designing the cost functions with these facts in mind.

We have also built on existing work in evolving nonlinear approximations not merely by incorporating a more detailed knowledge of the problem domain, but by studying the various possible move functions and by establishing the existence of a smooth search landscape for one type of move function when evolving nonlinear approximations.

We have also incorporated the newly evolved approximations into attacks on DES and Serpent, and although we have not improved on the performance of the best attack on DES, we have succeeded in devising an attack on 11-round Serpent with better data complexity than any other known-plaintext attack, and have also achieved the best time complexity of any attack so far on 11-round Serpent-256.

We now consider directions in which this research might proceed further.

Instead of trying to modify the approximations from existing linear attacks - with the resulting increase in the number of active plaintext/ciphertext bits - we would like to develop new algorithms to search for approximations which achieve a better bias/active-outer-round-S-box tradeoff. We would also like to develop statistical frameworks for, and working prototypes of, differential-nonlinear attacks.

We mentioned earlier that we did not know how to combine multidimensional linear approximations and nonlinear approximations in the same attack. This may prove a promising research avenue, as might attempts to move from what is basically a multiple-approximation attack using the multiple “related” nonlinear approximations to a multidimensional nonlinear attack utilising all linear combinations of the relateds, and indeed to explore further generalisations of multiple nonlinear cryptanalysis to forms of multidimensional cryptanalysis.

Although it is not in general possible to link together nonlinear approximations to the inner rounds of a cipher, for certain weak key classes it may be possible to do so in certain cases, leading to increasingly powerful attacks for such keys. Search algorithms to find such approximations may also prove useful in identifying weak key classes of this form.

We have already mentioned that use of the log-likelihood ratio statistic - or indeed approximations thereof - to achieve attacks with even lower data complexity is impaired for approximations more than a certain number of rounds in length by the linear hull effect. Collard et al.’s work [22] shows the true distribution for the bias of a single linear approximation becoming increasingly key-dependent, and diverging increasingly from the theoretical distribution calculated beforehand, as the number of approximated rounds increases. Research into means by which the scale of the effect might be estimated, and partial information about the theoretical distribution incorporated into a modified LLR-like statistic could prove fruitful. Possibly, for individual attacks, nonlinear approximations to inner rounds might be utilised to obtain information about the various key-dependent distributions.

There has also been research, as mentioned earlier [14, 63] into a variation of linear cryptanalysis *exploiting* the linear hull effect [42, 31, 62]. Although we have not yet found a way to exploit nonlinear approximations in the best-known use of this method, to whit the attack on PRESENT mentioned previously, research into combining nonlinear approximations with linear hull cryptanalysis could prove promising.

## 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] R. Anderson, E. Biham, L. Knudsen, and F. Stajano. Serpent optimised (bitslice) reference implementation. 1998. C source code. Can be downloaded from <http://www.cl.cam.ac.uk/~rja14/Papers/serpent.tar.gz>; is contained in the archive's "floppy2" folder.
- [3] R. Anderson, E. Biham, L. Knudsen, and F. Stajano. Serpent reference implementation. 1998. C source code. Can be downloaded from <http://www.cl.cam.ac.uk/~rja14/Papers/serpent.tar.gz>; is contained in the archive's "floppy1" folder.
- [4] T. Baignères, P. Junod, and S. Vaudenay. How far can we go beyond linear cryptanalysis? In Pil Joong Lee, editor, *Advances in Cryptology - Asiacrypt 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450. IACR, Springer, December 2004.
- [5] E. Biham. A fast new DES implementation in software. In E. Biham, editor, *Proceedings of the Fourth International Workshop on Fast Software Encryption (FSE 1997)*, volume 1267 of *Lecture Notes in Computer Science*, pages 260–272. IACR, Springer, January 1997.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] A. Biryukov, C. De Cannière, and M. Quisquater. On multiple linear approximations. In M. Franklin, editor, *Advances in Cryptology - Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 1–22. IACR, Springer, 2004.
- [10] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, *Proceedings of the Ninth International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007)*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. IACR, Springer, September 2007.
- [11] 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>.
- [12] 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>.
- [13] Joo Yeon Cho. Linear cryptanalysis of reduced-round PRESENT. Cryptology ePrint Archive, Report 2009/397. 2009. <http://eprint.iacr.org/2009/397>.

- [14] Joo Yeon Cho. Linear cryptanalysis of reduced-round PRESENT. 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 302–317. Springer, March 2010.
- [15] J.Y. Cho, M. Hermelin, and K. Nyberg. Multidimensional linear cryptanalysis of reduced round Serpent. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *Proceedings of the Thirteenth Australasian Conference on Information Security and Privacy (ACISP 2008)*, volume 5107 of *Lecture Notes in Computer Science*, pages 203–215. Springer, July 2008.
- [16] J.Y. Cho, M. Hermelin, and K. Nyberg. A new technique for multidimensional linear cryptanalysis with applications on reduced round Serpent. In Pil Joong Lee and Jung Hee Cheon, editors, *Proceedings of the 11th International Conference on Information Security and Cryptology (ICISC 2008)*, volume 5461 of *Lecture Notes in Computer Science*, pages 383–398. Springer, December 2008.
- [17] 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.
- [18] J.Y. Cho, M. Hermelin, and K. Nyberg. Statistical tests for key recovery using multidimensional extension of Matsui's algorithm 1. In: Eurocrypt 2009 - poster session. April 2009. <http://research.ics.tkk.fi/publications/mhermeli/dags-unif-alg1.pdf>.
- [19] J.A. Clark, J.C Hernández-Castro, and J.M.E. Tapiador. Private communication. C source code used in “Non-linear Cryptanalysis Revisited: Heuristic Search for Approximations to S-Boxes”.
- [20] J.A. Clark, J.C. Hernández-Castro, and J.M.E. Tapiador. Heuristic search for non-linear cryptanalytic approximations. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC2007)*, pages 3561–3568. IEEE, September 2007. doi:10.1109/CEC.2007.4424934.
- [21] 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.
- [22] B. Collard and F.-X. Standaert. Experimenting linear cryptanalysis. 2011. <http://perso.uclouvain.be/fstandae/PUBLIS/90.pdf>.
- [23] 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.
- [24] 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>.
- [25] 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.
- [26] 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.

- [27] N.T. Courtois. Feistel schemes and bi-linear cryptanalysis. Cryptology ePrint Archive, Report 2005/251. August 2005. <http://eprint.iacr.org/2005/251>.
- [28] N.T. Courtois. CTC2 and fast algebraic attacks on block ciphers revisited. Cryptology ePrint Archive, Report 2007/152. May 2007. <http://eprint.iacr.org/2007/152>.
- [29] 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>.
- [30] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley-Interscience, second edition, 2006.
- [31] Zhenli Dai, Meiqin Wang, and Yue Sun. Effect of the dependent paths in linear hull. Cryptology ePrint Archive, Report 2010/325. 2010. <http://eprint.iacr.org/2010/325>.
- [32] H.A. David. *Order Statistics*. Wiley, second edition, 1981.
- [33] O. Dunkelman. Private communication.
- [34] O. Dunkelman, S. Indestege, 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.
- [35] J. Fuller and W. Millan. On linear redundancy in the AES S-box. Cryptology ePrint Archive, Report 2002/111. August 2002. <http://eprint.iacr.org/2008/385>. Full version published as “Linear Redundancy in S-Boxes” in proceedings of FSE 2008 (LNCS 2887).
- [36] 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.
- [37] M. Hermelin. *Multidimensional Linear Cryptanalysis*. PhD thesis, Aalto University, June 2010. <https://aaltodoc.aalto.fi/handle/123456789/4802>.
- [38] 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.
- [39] M. Hermelin and K. Nyberg. Linear cryptanalysis using multiple linear approximations. Cryptology ePrint Archive, Report 2011/093. February 2011. <http://eprint.iacr.org/2011/093>.
- [40] H.M. Heys. A tutorial on linear and differential cryptanalysis. Technical Report CORR 2001-17, University of Waterloo, March 2001. Available online, with errata, at <http://www.engr.mun.ca/~howard/Research/Papers/index.html>.
- [41] B.S. Kaliski Jr. and M.J.B. Robshaw. Linear cryptanalysis using multiple approximations. In Y.G. Desmedt, editor, *Advances in Cryptology - Crypto ’94*, volume 839 of *Lecture Notes in Computer Science*, pages 26–39. IACR, Springer, 1994.
- [42] J. Nakahara Jr., P. Sepehrdad, M. Wang, and B. Zhang. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In J.A. Garay, A. Miyaji, and A. Otsuka, editors, *Proceedings of the 8th International Conference on Cryptology and Network Security (CANS 2009)*, volume 5888 of *Lecture Notes in Computer Science*, pages 58–75. Springer, December 2009.

- [43] P. Junod. On the complexity of Matsui's attack. In S. Vaudenay and A.M. Youssef, editors, *Proceedings of the 8th Annual International Workshop on Selected Areas in Cryptography (SAC 2001)*, volume 2259 of *Lecture Notes in Computer Science*, pages 199–211. Springer, August 2001.
- [44] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986, March 1984.
- [45] L.R. Knudsen. October 2008. Private communication.
- [46] 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.
- [47] 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.
- [48] M. Kwan. Reducing the gate count of bitslice DES. Cryptology ePrint Archive, Report 2000/051. October 2000. <http://eprint.iacr.org/2000/051>.
- [49] M. Kwan. Post-postscript. Kwan's homepage, <http://www.darkside.com.au/bitslice/>. June 2011.
- [50] G. Leander. Small scale variants of the block cipher PRESENT. Cryptology ePrint Archive, Report 2010/143. March 2010. <http://eprint.iacr.org/2010/143>.
- [51] 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.
- [52] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseht, editor, *Advances in Cryptology - Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. IACR, Springer, 1993.
- [53] 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.
- [54] J. Monnerat and S. Vaudenay. Generic homomorphic undeniable signatures - erratum. February 2005. [http://infoscience.epfl.ch/record/99428/files/MV04d\\_note.pdf](http://infoscience.epfl.ch/record/99428/files/MV04d_note.pdf).
- [55] S. Murphy. The independence of linear approximations in symmetric cryptanalysis. *IEEE Transactions on Information Theory*, 52(12):5510–5518, December 2006. [http://www.isg.rhul.ac.uk/~sean/Mult\\_lin\\_chi\\_IEEE\\_FinalA.pdf](http://www.isg.rhul.ac.uk/~sean/Mult_lin_chi_IEEE_FinalA.pdf).
- [56] 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-Rev2-11nsc.pdf](http://www.isg.rhul.ac.uk/~sean/Linear_Hull_JMC-Rev2-11nsc.pdf).
- [57] S. Murphy. Overestimates for the gain of multiple linear approximations. Technical Report RHUL-MA-2009-21, Royal Holloway, University of London, October 2009. [http://www.isg.rhul.ac.uk/~sean/MLA\\_Gain.pdf](http://www.isg.rhul.ac.uk/~sean/MLA_Gain.pdf).
- [58] S. Murphy. Overestimates for the gain of multiple linear approximations in symmetric cryptology. *IEEE Transactions on Information Theory*, 57(7):4794–4797, July 2011.
- [59] National Institute for Science and Technology (NIST). Data Encryption Standard (DES). October 1999. <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>.

- [60] Phuong Ha Nguyen, Hongjun Wu, and Huaxiong Wang. Improving the algorithm 2 in multi-dimensional 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.
- [61] K. Nyberg. Multidimensional extension of Matsui’s algorithm 2 - a revision of subsection 5.1. Private email communication, 2011.
- [62] K. Nyberg and A. Röck. Exploiting linear hull in Matsui’s algorithm 1 (extended version). Cryptology ePrint Archive, Report 2011/285. May 2011. <http://eprint.iacr.org/2011/285>.
- [63] K. Ohkuma. Weak keys of reduced-round PRESENT for linear cryptanalysis. In M.J. Jacobson Jr, V. Rijmen, and R. Safavi-Naini, editors, *Proceedings of the 16th Annual International Workshop on Selected Areas in Cryptography (SAC 2009)*, volume 5867 of *Lecture Notes in Computer Science*, pages 249–265. Springer, August 2009.
- [64] D.A. Osvik. Speeding up Serpent. In *Proceedings of the 3rd Advanced Encryption Standard Candidate Conference (AES 2000)*, April 2000.
- [65] 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>.
- [66] 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 [6], on page 20, after S6 is applied the only active bit in the state is bit 30. In the descriptions given in later papers [8, 34], 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 linear diffusion layer is then applied, after which the active bits according to the diagram are 80, 101 and 103. However, by examining the description of Serpent’s diffusion layer in its AES proposal [1], and the C reference implementation [3], we see that 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 another typographical error - the active bits at this point should in fact have been 81, 83 and 100.

### B The simulated annealing algorithm

Simulated annealing is a local-search based algorithm, akin in many ways to a more complex form of hill-climbing. It is inspired by a technique used in metallurgy to eliminate defects in the crystalline structures in samples of metal.

In simulated annealing, we are attempting to create an entity of some particular type - in this paper, an object of the class *nonlinear\_approximation*. The “search space” is the set of all such

entities, and any member of this set is referred to as a “candidate solution”. It must also be possible to define:

- A “cost function”, which takes a candidate solution as input and outputs a scalar value. The better the candidate solution is, according to the criteria which the user wishes to optimise, the lower this value should be.
- A “move function”, which takes a candidate solution and makes some small change to it before outputting the result. Each such “move” should have only a small effect on the cost, preferably with a known upper bound. The set of all solution candidates which can be obtained by making one move from the current candidate  $C$  is referred to as the “1-move neighbourhood” of  $C$ .

The smaller the effect of the move function on the cost, the “smoother” we refer to the search space - or “search landscape” - as being.

At the start of the algorithm, some initial candidate solution,  $S_0$ , usually chosen at random, is input to the SA algorithm, along with the following parameters:

- The cost function  $C$ .
- The initial value  $T_0$  for the “temperature”. The higher the temperature in the current iteration, the more likely the search algorithm is to accept a move which results in a candidate solution with higher cost than the current candidate (that is, to store said candidate solution as the “current candidate”). The temperature drops over time, causing the algorithm to accept fewer non-improving moves and hence to shift away from exploration and towards optimisation. Towards the end of the search, it is extremely rare for the algorithm to accept a non-improving move, and its behaviour is very close to that of a hill-climbing algorithm.
- In choosing the value of  $T_0$ , various sources state that it should be chosen so that a particular proportion of moves are accepted at temperature  $T_0$ . There is very little information or advice available as to what this proportion should be. In one of the earliest papers on simulated annealing [44] it is stated that any temperature leading to an initial acceptance rate of 80% or more will do; however our initial experiments indicated that this was far too high for most of the experiments in this thesis. We usually settled on an initial acceptance rate of 0.5 or 0.6 instead of 0.8.

Having chosen the initial acceptance rate, the experimenter executes the annealing algorithm with various  $T_0$  until a temperature is found that achieves a fraction close enough to this. We started with the temperature at 0.1, and repeatedly ran the algorithm, doubled the temperature, and re-ran the algorithm until an acceptance rate at least as high as that specified was obtained. Where  $T_a$  was the temperature at which this had been achieved, and  $T_b = T_a/2$ , we then used a binary-search-like algorithm to obtain a temperature between  $T_a$  and  $T_b$  that would result in an acceptance rate  $\approx 50\%$ .

- A value  $\alpha$ ; the “cooling factor”, determining how far the temperature decreases at each iteration of the algorithm.
- An integer value: *MAX\_INNER\_LOOPS*, determining the number of moves that the local search algorithm can make at each temperature.
- The stopping criterion must also be specified. We used a *MAX\_OUTER\_LOOPS* value, indicating how many times the algorithm was to be allowed to reduce the temperature and continue searching before it stopped.
- We also specified a *MAX\_FROZEN\_OUTER\_LOOPS* parameter. If the algorithm had, at any stage, executed this many outer loops without accepting a single move, it would be considered extremely unlikely to do anything other than remain completely stationary from then on, and would be instructed to terminate early.

---

**Algorithm 4** Pseudocode for simulated annealing algorithm

---

```
S ← S0
bestsol ← S0
T ← T0
ZERO_ACCEPT_LOOPS ← 0
for x ← 0, MAX_OUTER_LOOPS − 1 do
  ACCEPTS_IN_THIS_LOOP ← false
  for y ← 0, MAX_INNER_LOOPS − 1 do
    Choose some Sn in the 1-move neighbourhood of S.
    cost_diff ← C(Sn) − C(S)
    if cost_diff < 0 then
      S ← Sn
      ACCEPTS_IN_THIS_LOOP ← true
      if C(Sn) < C(bestsol) then
        bestsol ← Sn
      end if
    else
      u ← Rnd(0,1)
      if u < exp(−cost_diff/T) then
        S ← Sn
        ACCEPTS_IN_THIS_LOOP ← true
      end if
    end if
  end for
  if ACCEPTS_IN_THIS_LOOP = false then
    ZERO_ACCEPT_LOOPS ← ZERO_ACCEPT_LOOPS + 1
    if ZERO_ACCEPT_LOOPS = MAX_FROZEN_OUTER_LOOPS then
      return bestsol
    end if
  end if
  T ← T × α
end for
return bestsol
```

---

▷ Algorithm terminates early.