

χ^2 Cryptanalysis of the SEAL Encryption Algorithm

[Published in E. Biham, Ed., *Fast Software Encryption*, vol. 1267 of *Lecture Notes in Computer Science*, pp. 1–12, Springer-Verlag, 1997.]

Helena Handschuh^{1,2} and Henri Gilbert³

¹ Gemplus, Cryptography Department
34 rue Guynemer, 92447 Issy-Les Moulineaux, France
handschuh@gemplus.com

² ENST, Computer Science Department
46 rue Barrault, 75634 Paris Cedex 13, France
handschu@inf.enst.fr

³ France Télécom-CNET, PAA-TSA-SRC
38-40 rue du Général Leclerc
92131 Issy-les-Moulineaux, France

Abstract. SEAL was first introduced in [1] by Rogaway and Coppersmith as a fast software-oriented encryption algorithm. It is a pseudorandom function which stretches a short index into a much longer pseudorandom string under control of a secret key pre-processed into internal tables. In this paper we first describe an attack of a simplified version of SEAL, which provides large parts of the secret tables from approximately 2^{24} algorithm computations. As far as the original algorithm is concerned, we construct a test capable of distinguishing SEAL from a random function using approximately 2^{30} IV values. Moreover, we describe how to derive some bits of information about the secret tables. These results were confirmed by computer experiments.

1 Description of the SEAL Algorithm

SEAL is a length-increasing "pseudorandom" function which maps a 32-bit string n to an L -bit string $\text{SEAL}(n)$ under a secret 160-bit key a . The output length L is meant to be variable, but is generally limited to 64 kbytes. In this paper, we assume it is worth exactly 64 kbytes (2^{14} 32-bit words), but all our results could be obtained with a smaller output length.

The key a is only used to define three secret tables R , S , and T . These tables respectively contain 256, 256 and 512 32-bit values which are derived from the Secure Hash Algorithm (SHA) [2] using a as the secret key and re-indexing the 160-bit output into 32-bit output words.

SEAL is the result of the two cascaded generators shown below.

The first generator uses a routine depending on the a -derived tables R and T depicted at figure 1. It maps the 32-bit string n and the 6-bit counter l to

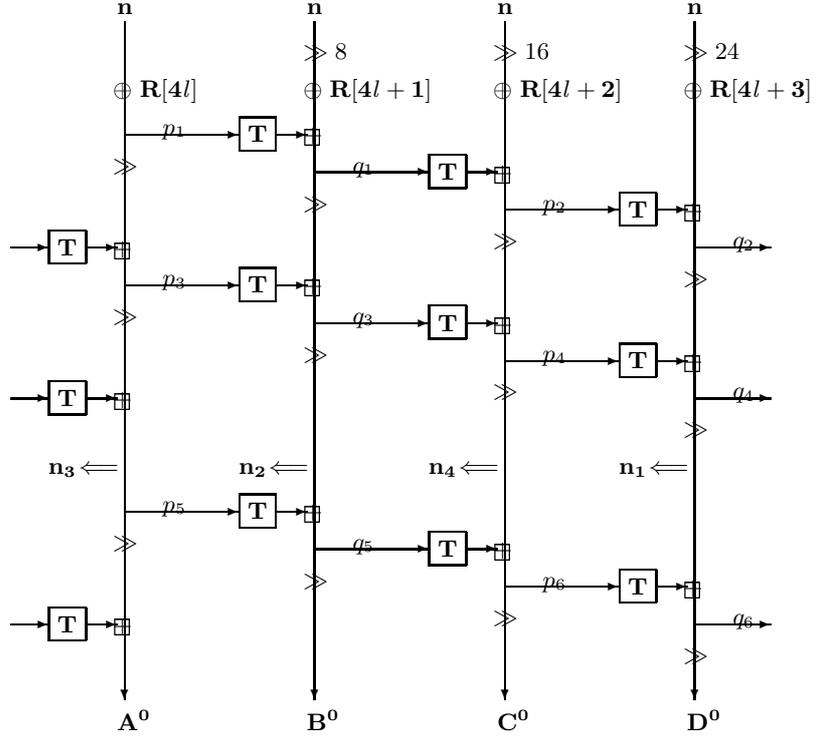


Fig. 1. The first generator of SEAL

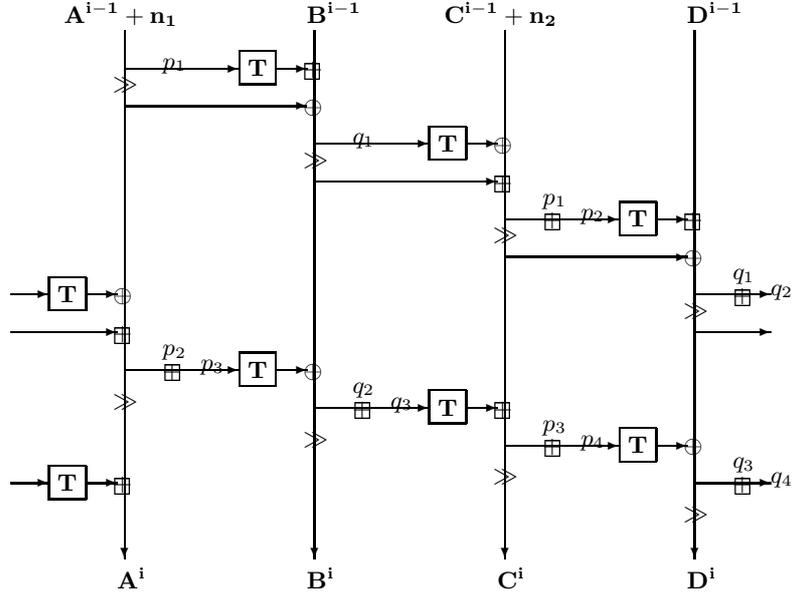


Fig. 2. The second generator of SEAL (i^{th} iteration)

four 32-bit words A^0, B^0, C^0, D^0 and another four 32-bit words n_1, n_2, n_3, n_4 . These eight words are to be used by the second generator.

The second generator uses a routine depending on the a -derived tables depicted at figure 2. There are 64 iterations of this routine, indexed by $i = 1$ to 64. $A^0B^0C^0D^0$ serves as an input to the first iteration, producing an $A^1B^1C^1D^1$ block. For the next iterations, the input block is alternately $(A^{i-1} + n_1, B^{i-1}, C^{i-1} + n_2, D^{i-1})$ for even i values and $(A^{i-1} + n_3, B^{i-1}, C^{i-1} + n_4, D^{i-1})$ for odd i values. At iteration i , the output block $(Y_1^i, Y_2^i, Y_3^i, Y_4^i)$ is deduced from the intermediate block (A^i, B^i, C^i, D^i) using the a -derived table S as shown below in figure 3.

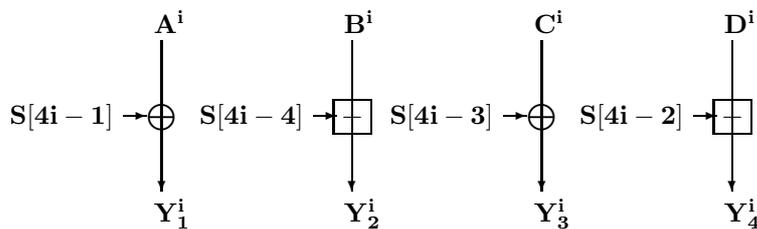


Fig. 3. Deriving the generator output

In the above figures:

- \oplus stands for the XOR function;
- \square stands for the sum $(\text{mod } 2^{32})$;
- \gg stands for a right rotation of 9 bits;
- $\gg N$ stands for a right rotation of N bits;
- p_1 through p_4 and q_1 through q_4 stand for the inputs of table T obtained from the 9 bits 2 to 11 of values A, B, C and D ; for instance in figure 1, $p1 = A \& 0x7fc$.

Concerning the definition of SEAL, more details can be found in [1] and in [2].

The algorithm is divided into three steps.

- First we compute the internal tables under the secret key a . The security of this step relies on SHA which is assumed to be highly secure. Therefore, R, S and T are pseudorandom tables.
- Second we compute $A^0, B^0, C^0, D^0, n_1, n_2, n_3$ and n_4 from n, l and table R . This is what we already called the first generator. Let us assume the output is pseudorandom as well.
- Finally, the second generator computes iteratively the $A^iB^iC^iD^i$ blocks, from which the $Y_1^i, Y_2^i, Y_3^i, Y_4^i$ values are derived. We change the original notations as follows:

- $Y_1^i = A^i \oplus S_1^i$
- $Y_2^i = B^i + S_2^i$

- $Y_3^i = C^i \oplus S_3^i$
- $Y_4^i = D^i + S_4^i$.

In this part we found certain weaknesses which are investigated in Sections 3 and 4.

2 Preliminary

2.1 Role of mod 2^{32} additions

Although the combined use of the $+$ and \oplus operations probably strengthens SEAL as compared with a situation where only one of these operations would be used, we do not believe that this represents the main ingredient of the security of SEAL, which is essentially a table-driven algorithm.

As a matter of fact, any $x + y$ sum can be written:

$$x + y = x \oplus y \oplus c(x, y)$$

where the carry word $c(x, y)$ is far from being uniformly distributed thus $+$ just introduces an additional, unbalanced term, as compared with \oplus .

This remark led us to assume that replacing in SEAL (more precisely in the second generator of figure 2) all $+$ operators by xors would not fundamentally modify the nature of the algorithm, and that cryptanalytic results obtained with such a simplified version could at least partially be transposed to the real cipher. The results of our analysis of this simplified version of SEAL are summarised in Section 3 hereafter.

2.2 The three words D^{i-1} , C^i and D^i are correlated

Let us consider the function depicted at figure 2. Given a fixed value of the iteration index i (say $i = 3$), the input and output to this function are known from the generator outputs $(Y_1^{i-1}, Y_2^{i-1}, Y_3^{i-1}, Y_4^{i-1})$ and $(Y_1^i, Y_2^i, Y_3^i, Y_4^i)$ up to the following unknown words:

- the 8 words $(S_1^{i-1}, S_2^{i-1}, S_3^{i-1}, S_4^{i-1})$ and $(S_1^i, S_2^i, S_3^i, S_4^i)$, which value does not depend upon the considered initial value (n, l) .
- the 2 words n_1 and n_2 , which value depends upon (n, l) .

The involvement of the IV-dependent words n_1 and n_2 in the function considerably complicates the analysis of the i^{th} iteration because of the randomisation effect upon the input to output dependency.

In order to find statistics applicable to any IV value, we investigate how to "get rid" of any dependency in n_1 and n_2 in some relations induced by the equation of iteration i .

Let us consider the D^{i-1} input word and the C^i and D^i input words. Denote the output tables involved in the right part of figure 2 by: $T_1 = T[p_2]$, $T_2 = T[q_3]$ and $T_3 = T[p_4]$. It is easy to establish the relation:

$$(D^{i-1} + T_1) \oplus (C^i \lll 9 + T_2) = (D^i \lll 18) \oplus (T_3 \lll 9) \quad (1)$$

This relation does not involve n_1 and n_2 . The T_1, T_2, T_3 terms in this relation can be seen as three random values selected from the T table. Since there are only 2^9 values in the T table, given any two words out of the (D^{i-1}, C^i, D^i) triplets, there are at most 2^{27} possible values for the third word of the triplet instead of 2^{32} if D^{i-1}, C^i and D^i were statistically independent. This gives some evidence that the D^{i-1} input and the C^i and D^i output are statistically correlated, in a way which does not depend upon n_1 and n_2 . In other words, the SEAL generator derives from an (n, l) initial value three slightly correlated output words Y_4^{i-1}, Y_3^i and Y_4^i .

Relation (1) above represents the starting point for the various attacks reported in Sections 3 and 4 hereafter.

3 An Attack of a Simplified Version of SEAL

In this Section we present an attack of the simplified version of SEAL obtained by replacing in figure 2 all mod 2^{32} additions by xors. The attack is divided into four steps.

3.1 Step 1

We derive the unordered set of values of the T table, up to an unknown 32-bit constant D^i . Relation (1) above represents the starting point for this derivation. After replacing $+$ by \oplus in (1) and D^{i-1}, C^i, D^i by $X_4 = Y_4^i, Y_3 = Y_3^i$ and $Y_4 = Y_4^i$ respectively, we obtain the relation:

$$Y_4 \oplus Y_3 \ggg 9 \oplus X_4 \ggg 18 = T_3 \ggg 9 \oplus (T_1 \oplus T_2) \ggg 18 \oplus \Delta^i \ggg 9 \quad (2)$$

where the Δ^i constant depends upon the S table. T_1 and T_2 are 2 among 512 values of table T . Statistically speaking, once in 2^9 , $T_1 = T_2$, and $T_1 \oplus T_2 = 0$. If we compute 2^{18} samples, each of the 512 values of table $T \oplus \Delta^i$ should appear once in average.

We collect the combination of the generator output words given by the left term of (2) for about 2^{21} (n, l) samples. Whenever one value appears more than 4 times, we assume this is a value of table $T \oplus \Delta^i$. All the other values have a probability of about $\frac{2^{21}}{2^{32}}$ to appear. This way, we find about 490 out of 512 values of table T up to one constant value.

3.2 Step 2

The purpose of the second step is to compute a constant α^i which is needed in the third step in order to find out statistics involving B^{i-1}, D^i, A^i and B^i (see Fig. 2.). Consider the following equation (3) established in a similar way to (2) from the relation between B^{i-1} and the output words:

$$\begin{aligned} Y_4 \ggg 9 \oplus Y_2 \oplus Y_1 \ggg 9 \oplus X_2 \ggg 18 \oplus T_3 \ggg 18 \\ = (T'_1 \oplus T'_2) \ggg 18 \oplus (T'_3 \oplus T'_4) \ggg 9 \oplus (S_4^i \ggg 9 \oplus S_2^{i-1} \ggg 18 \oplus S_2^i \oplus S_1^i \ggg 9) \end{aligned} \quad (3)$$

where

$$\alpha^i = S_4^i \gg 9 \oplus S_2^{i-1} \gg 18 \oplus S_2^i \oplus S_1^i \gg 9 \oplus \Delta^i \gg 18$$

For each sample, we can find out $T_3 \oplus \Delta^i$ by searching exhaustively the right combination (T_1, T_2, T_3) in equation (2). In order to save time, we compute once and for all a table with the 2^{18} values of $(T_1 \oplus T_2) \gg 18$ and search for the right third value. We perform this search as well as the computation of the left term of (3) for 2^{21} samples. Once in 2^{18} , $T'_1 = T'_2$ and $T'_3 = T'_4$. This way the constant value α^i we are looking for appears at least 4 times.

3.3 Step 3

The purpose of this step is to find out various values of n_1 . Once we have these values, we can make out the relation between the in and outputs of table T up to one constant value. Let us consider equation (4) established from the relation between A^{i-1} and the output words:

(4)

$$X_1 \gg 18 \oplus Y_1 \oplus Y_4 \oplus T'_2 \gg 9 \oplus T'_4 \oplus T_3 \gg 9 = n_1 \gg 18 \oplus S_1^{i-1} \gg 18 \oplus S_4^i \oplus S_1^i$$

We can find out $T'_2 \oplus \Delta^i$ and $T'_4 \oplus \Delta^i$ by searching the right combination of (T'_1, T'_2, T'_3, T'_4) in equation (2) using the value α^i we made out in step 2. For each sample we compute, we get about 16 possibilities, as (T'_1, T'_2, T'_3, T'_4) gives 2^{36} possible values for a 32-bit word.

In order to find the right combination, let us consider two distinct iteration indexes i and j : we know that for a given l value, if i is even (or odd), we always xor the same n_1 (or n_3) to the input A . Let us therefore take two rounds i and j that are both odd (or even). We need to know table $T \oplus \Delta^i$, table $T \oplus \Delta^j$, α^i and α^j .

We collect samples of:

- $n_1 \gg 18 \oplus \beta^i$
where $\beta^i = S_1^{i-1} \gg 18 \oplus S_4^i \oplus S_1^i \oplus \Delta^i$;
- $n_1 \gg 18 \oplus \beta^j \oplus \Delta^j \gg 9 \oplus \Delta^i \gg 9$
as value T_3 is found through table $T \oplus \Delta^i$ and values T'_2 and T'_4 through table $T \oplus \Delta^j$.

We xor all the samples of round i with all the samples of round j . One of these values is the right combination of $\beta^i \oplus \beta^j \oplus (\Delta^i \oplus \Delta^j) \gg 9$

Then we find all the samples for rounds i and j of another value n_1 (i.e. of round l). We compare these two sets of samples and make out the right value of $n_1 \gg 18 \oplus \beta^i$.

This step can be repeated various times to collect values of n_1 while computing only once the tables $T \oplus \Delta$ and the constants α .

3.4 Step 4

In this step we finally derive the in and outputs of table T from equation (5):

$$p_1 = ((X_1 \oplus n_1 \oplus S_1^{i-1}) \& 0x7fc) / 4$$

In this equation p_1 is the input of table T . We have seen in the first three steps that we can derive the value of T_1 from in and output samples of SEAL.

So we finally derive several values of:

$$T[p \oplus \delta^i] \oplus \Delta^i$$

where $\delta^i = ((S_1^{i-1} \oplus \beta^i \ll 18) \& 0x7fc) / 4$.

3.5 Summary

Summing up the four steps we have just described, we can break the T table up to one constant value using about 2×2^{21} samples of (n, l) for step 1, 2×2^{21} samples of (n, l) for step 2 and about 2^9 values of (n, l) for steps 3 and 4. This means, the T table can be broken using about 2^{24} samples of (n, l) .

We could probably go on breaking the simplified version of SEAL by finding out sets of values (n_1, n_2, n_3, n_4) , then trying to break the first generator and find table R , but this is not our purpose here.

4 A Test of the Real Version of SEAL

In this Section we use some of the ideas of Vaudenay's Statistical Cryptanalysis of Block Ciphers [3] to distinguish SEAL from a truly random function.

4.1 χ^2 Cryptanalysis

The purpose of Vaudenay's paper is to prove that statistical analysis on ciphers such as DES may provide as efficient attacks as linear or differential cryptanalysis. Statistical analysis enables to recover very low biases and a simple χ^2 test can get very good results even without knowing exactly what happens inside the inner loops of the algorithm or the S-boxes.

We intend to use this property to detect low biases of a certain combination of the output words of SEAL suggested by the analysis made in Section 3 in order to prove SEAL is far from being undistinguishable from a pseudo-random function. This provides a first test of the SEAL algorithm.

4.2 Number of samples needed for the χ^2 test to distinguish a biased distribution from an unbiased one

We denote by N the number of samples computed. We assume the samples are drawn from a set of r values. We call n_i the number of occurrences of the i^{th} of the r values among the N samples and S_{χ^2} the associated indicator:

$$S_{\chi^2} = \frac{\sum_{i=1}^r (n_i - \frac{N}{r})^2}{\frac{N}{r}}$$

The χ^2 test compares the value of this indicator to the one an unbiased distribution would be likely to provide. If the n_i were drawn according to an unbiased multinomial distribution of parameters $(\frac{1}{r}, \frac{1}{r}, \dots, \frac{1}{r})$, the expectation and the standard deviation of the S_{χ^2} estimator would be given by:

$$\begin{aligned} - E(S_{\chi^2}) &\rightarrow \mu = r - 1 \\ - \sigma(S_{\chi^2}) &\rightarrow \sigma = \sqrt{2(r - 1)} \end{aligned}$$

If the distribution of the n_i is still multinomial but biased, say with probabilities p_1, \dots, p_r , then we can compute the new expected value μ' of the S_{χ^2} :

$$\mu' = E\left(\frac{\sum_{i=1}^r (n_i - \frac{N}{r})^2}{\frac{N}{r}}\right) = \frac{N}{r} \sum_{i=1}^r E\left((n_i - p_i N + p_i N - \frac{N}{r})^2\right)$$

It can be easily shown that:

$$\mu' \rightarrow \mu + r(N - 1) \sum_{i=1}^r (p_i - \frac{1}{r})^2$$

An order of magnitude of the number N of samples needed by the χ^2 test to distinguish a biased distribution from an unbiased one with substantial probability is given by the condition:

$$\mu' - \mu \gg \sigma$$

which gives us the following order of magnitude for N :

$$N \gg \frac{\sqrt{2(r - 1)}}{r \sum_{i=1}^r (p_i - \frac{1}{r})^2}$$

4.3 Model of the test

Let us consider equation (2) with the real scheme (including the sums). We can rewrite it:

$$Y_4 \oplus Y_3 \gg 9 \oplus X_4 \gg 18 = T_3 \gg 9 \oplus (T_1 \oplus T_2) \gg 18 \oplus (r_1 \oplus r_2 \oplus r_3) \gg 18 \oplus \Delta^i \gg 9 \oplus r_4$$

where r_1 and r_2 are the carry bits created by the addition of T_1 and T_2 , and r_3 and r_4 the ones of the addition of S_4^{i-1} and S_4^i .

We apply the χ^2 test to the four leftmost bits of $Y_4 \oplus Y_3 \gg 9 \oplus X_4 \gg 18$ suspecting a slight bias in this expression. Without having carefully analysed the exact distribution of the sum of the four carries, we intend to prove that its convolution with the biased distribution of $T_3 \gg 9 \oplus (T_1 \oplus T_2) \gg 18 \oplus \Delta^i \gg 9$ does result in a still slightly unbalanced distribution.

As we take 4-bit samples, we apply the χ^2 test with $r - 1 = 15$ degrees of freedom. Detailed information about this test can be found in [4].

4.4 Results

Whenever we analyse at least 2^{33} samples, the test proves with probability $\frac{1}{1000}$ to be wrong, that SEAL has a biased distribution. We have made several tests, and each time the value of the S_{χ^2} estimator we obtained for this order of magnitude of N was greater than 45 for the 4 leftmost bits and greater than 320 for the 8 leftmost bits. In other words, the test proves that the distribution is a biased one.

If the two leftmost bits of S_4^i can be found, the test works with 2^{30} samples or less. Similar results are found when testing the eight leftmost bits of the samples. Figure 4 shows the value of the S_{χ^2} estimator for tests made with $a = 0x67452301$.

S_{χ^2}	2^{23}	2^{24}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}
4 bits	14.27	25	13.97	9.41	26.96	16.5	16.78	29.65	21.05	30.15	45.74	44.69	55.96
8 bits	261	293	274	238	229	227	246	225	278	313	331	378	453

Fig. 4. Results of the tests with up to 2^{35} samples of (n, l) .

These and many more figures have been computed using various tests and trying out different cases.

4.5 Deriving first information on SEAL

It appears in equation (2), only r_4 has some influence in making the number of samples increase by a factor 4. Whenever the two leftmost bits of S_4^i are found, we know part of r_4 and the χ^2 test gives much better results than with wrong bits of S_4^i . Therefore we can derive at least two bits of information on table S . If the test is applied to more than the four leftmost bits of the samples, more than 2 bits can be derived from secret table T . Whenever these bits are right, the χ^2 rises much faster than for wrong values.

Compute about 2^{32} samples of the four leftmost bits of $Y_3 \gg 9 \oplus X_4 \gg 18$ and Y_4 and store the frequencies of every value of these 8 bits. For every possible value of the four leftmost bits of S_4^i , compute the new frequencies of $(Y_4 - S_4^i) \oplus Y_3 \gg 9 \oplus X_4 \gg 18$. The two leftmost bits of S_4^i correspond to the ones used for the two highest values of the χ^2 of the samples.

As the evolution of the χ^2 is quite close to a straight line when the divergence starts, check the results by applying the test to 2^{20} through 2^{32} samples. Divergence becomes obvious when about 2^{30} samples have been computed.

5 Conclusion

We have shown in some detail in Section 3 that the simpler scheme with xors instead of sums can be broken with about:

- 2^{21} samples of (n, l) for each one of the two tables $T \oplus \Delta$ of step 1;

- 2^{21} samples of (n, l) for each one of the two values of α of step 2;
- 512 samples of n for n_1 and β in steps 3 and 4, depending mostly upon how many elements of table $T \oplus \Delta$ you find in step 1. For each n , we find l and l' such that the corresponding values n_1 and n'_1 can be found when comparing all the results between rounds i and j .

The attack of Section 3 can therefore be realised with about 2^{24} samples of (n, l) .

The test of Section 4 can be applied with about 2^{32} samples of (n, l) and first information about table S can be derived from this test with 2^{32} samples of (n, l) as well. A good idea of secret table S can therefore be obtained with the same amount of samples of (n, l) but slightly more operations.

We believe the attack we showed can be adapted to the real scheme of SEAL, but we did not investigate whether the complexity is within reasonable range.

6 Acknowledgements

We thank Thierry Baritaud and Pascal Chauvaud for the elaboration of the L^AT_EX version of the figures and the paper. We would like to address special thanks to François Allègre who gave us access to a quite powerful computer that made the tests reasonably quick and to Alain Scheiwe who helped us write the C source code of the tests.

References

1. P. Rogaway and D. Coppersmith, "A Software-Optimized Encryption Algorithm", Proceedings of the 1993 Cambridge Security Workshop, Springer-Verlag, 1994.
2. B. Schneier, Applied Cryptography, Second Edition, John Wiley & Sons, 1996.
3. S. Vaudenay, "Statistical Cryptanalysis of Block Ciphers - χ^2 Cryptanalysis", 1995.
4. J. Bass, Eléments de Calcul des Probabilités, 3^e édition, Masson Et Cie, 1974.