χ^2 Cryptanalysis of the SEAL Encryption Algorithm

Helena Handschuh *
Gemplus PSI
1, Place de la Méditerranée
95200 Sarcelles
France

Henri Gilbert
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} computations. 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 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 study reported in this paper was performed while Helena Handschuh was working at France Télécom-CNET.

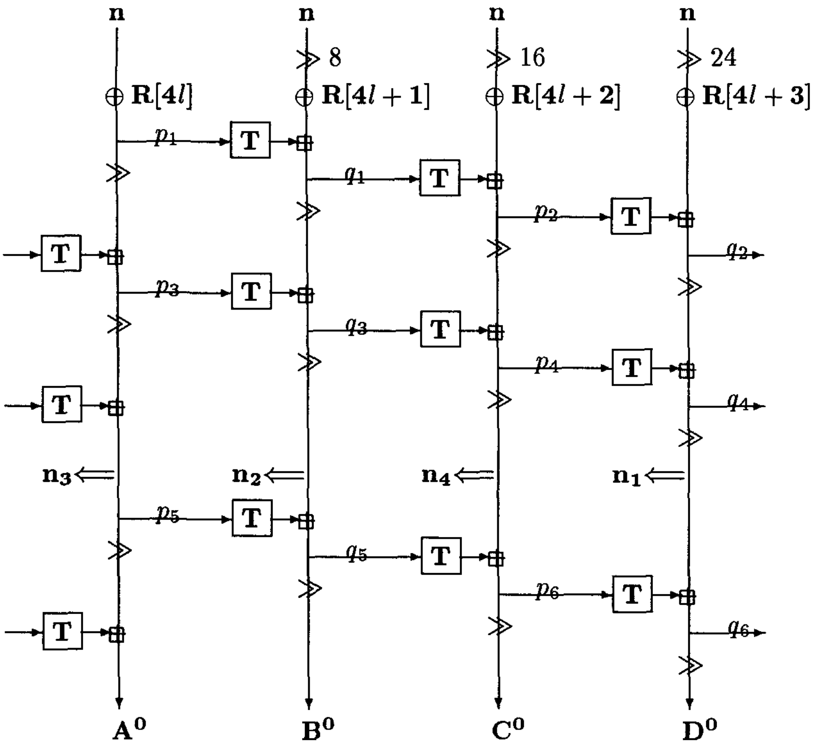

Fig. 1. The first generator of SEAL

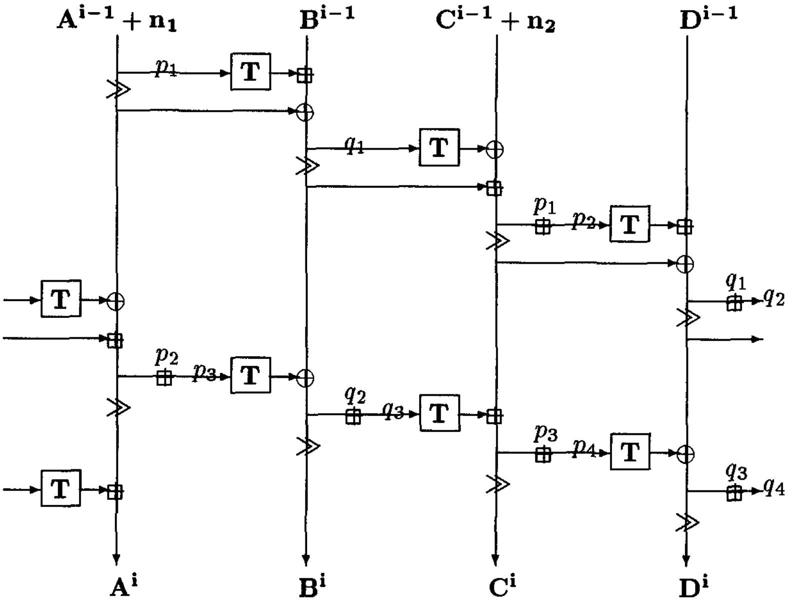

Fig. 2. The second generator of SEAL (i^{th} iteration)

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

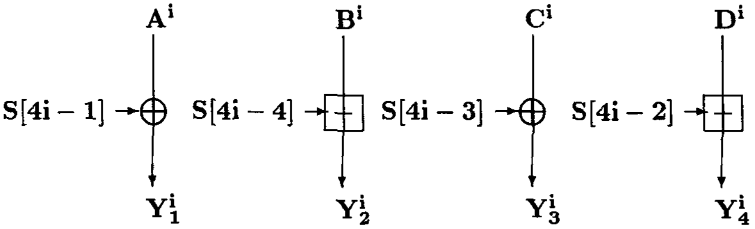

Fig. 3. Deriving the generator output

In the above figures :

- $\oplus$ stands for the XOR function;
- $+$ stands for the sum $\pmod{2^{32}}$;
- $\gg$ stands for a right rotation of 9 bits ($\gg$ has precedence over $+$ and $\oplus$);
- $\gg N$ 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 the 32-bit intermediate 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 notation 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 Preliminaries

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 in 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), the value of which does not depend on the considered initial value (n, l).
- the 2 words n_1 and n_2, the value of which depends on (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 on the input to output dependency.

In order to find statistics applicable to any IV value, we investigate how to "get rid" of any dependency on 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 :

(1)

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

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 to 5 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 Δ^i. Relation (1) above represents the starting point for this derivation. After replacing + by $\oplus$ in (1) and introducing the notation $X_4 = Y_4^{i-1}$, $Y_3 = Y_3^i$ and $Y_4 = Y_4^i$, we obtain the relation :

(2)

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

where the Δ^i constant depends on the S table. T_1 and T_2 are 2 among 512 values of the table T. Statistically speaking, once in 2^9, $T_1 = T_2$, thus $T_1 \oplus T_2 = 0$. If we compute 2^{18} samples, each of the 512 values of the table $T \oplus \Delta^i$ should appear once on 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 the table $(T \oplus \Delta^i) \gg 9$. 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 the table T up to a 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 :

(3) $Y_4 \gg 9 \oplus Y_2 \oplus Y_1 \gg 9 \oplus X_2 \gg 18 \oplus T_3 \gg 18 = (T_1' \oplus T_2') \gg 18 \oplus (T_3' \oplus T_4') \gg 9 \oplus (S_4^i \gg 9 \oplus S_2^{i-1} \gg 18 \oplus S_2^i \oplus S_1^i \gg 9)$

where

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

and

$$T_1' = T[p_1], T_2' = T[q_2], T_3' = T[p_3] and T_4' = T[q_4].$$

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 find the relation between the inputs and outputs of table T up to a 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 found 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 the combination of the generator output words given by the left term of (4) in order to find out possible values 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 find 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 inputs and outputs of table T from equation (5):

(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 input 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 a 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 occurences 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 $(\frac{1}{r}, \frac{1}{r}, \ldots, \frac{1}{r})$, the expectation and the standard deviation of the S_{χ^2} estimator would be given by :

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

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

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

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 :

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

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 40 for the 4 least significant bits and greater than 320 for the 8 least significant bits. In other words, the test proves that the distribution is a biased one.

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).

The former test test can be slightly improved as follows : let us denote the four least significant bits of S_4^i by s_4^i. For each of the 16 possible values of s_4^i, apply the S_{χ^2} test to the 4 or the 8 least significant bits of $(Y_4 - s_4^i) \oplus Y_3 \gg 9 \oplus X_4 \gg 18$. The test with the correct s_4^i value detects a bias with 2^{30} (n, l) values only (see Figure 5). Note that a significant bias is also detected whenever the two least significant bits of s_4^i are correct.

S_{χ^2}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}
4 bits	9.45	13.56	20.62	25.59	32.77	71.90	83.63	130	250	438	928
8 bits	253	271	292	321	276	357	379	438	569	838	1520

Fig. 5. Results of the test with the correct value of the four s_4^i bits.

4.5 Deriving first information on SEAL

The interpretation of the above improved test is straightforward : when the two least significant bits of s_4^i are correct, r_4 is partly known 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.

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

5 Deriving information on the T table

In this Section we give some evidence that the initial step of the attack on the simplified version of SEAL introduced in Section 3 can be adapted to provide large parts of the T table for the real algorithm.

Let us consider relation (6).

As seen in Section 3.1, the distribution of the $T_3 \gg 9 \oplus (T_1 \oplus T_2) \gg 18 \oplus \Delta^i \gg 9$ value at the right of (6) is unbalanced. The most frequent values are provided by the 512 $T \oplus \Delta^i$ words.

On the other hand the distribution of the carry words $r_1 \oplus r_2 \oplus r_3$ and r_4 is also unbalanced. More precisely, due to the fact that in any carry word r each bit $r[j]$ has a $\frac{3}{4}$ probability of being equal to the next bit $r[j+1]$, the number of 'inversions', i.e. j values s.t. $r[j] \neq r[j+1]$ is likely to be small when r is a carry word or an exclusive or of carry words.

Thus we can expect the 512 $T \oplus \Delta^i$ values to give rise to 'spread' probability maxima in the distribution probability of the left term of (6).

Based on 2^{32} (n,l) values, we did the following experiment :

We analysed the probability distribution of the 23 lowest weight bits of the left combination of (6) in order to reduce the memory requirements. So in the rest of this Section, though we do not introduce any new notation, we implicitly refer to 23-bit words instead of 32-bit words.

For several $T \oplus \Delta^i$ values (about 25), we computed the sum of the probabilities of the neighbours of $T \oplus \Delta^i$, i.e. the values of the form $T \oplus \Delta^i + r$, where r is one of the approximately 2^{22} 23-bit words with at most 11 inversions.

We computed the same sum of approximately 2^{22} probabilities around arbitrarily chosen values other than the 512 $T \oplus \Delta^i$ values.

For more than half of the $T \oplus \Delta^i$ values, the obtained sum was larger than all the sums associated to the arbitrarily chosen values.

The complexity of the search of the $T \oplus \Delta^i$ values is quite high if an independent computation of a sum of 2^{22} probabilities is made for each of the 2^{23} candidate values. This approach leads to a 2^{45} complexity, far over the computing capabilities of the computer we used for the experiments ; however, substantial gains might be achieved by reusing appropriately selected partial sums of probabilities.

Thus in summary, we believe that with slightly more than 2^{32} (n,l) values, it should be possible to recover a substantial part of the information on the T table, up to an unknown constant.

6 Conclusion

We have shown in some detail in Section 3 that the simpler scheme with xors instead of sums can be attacked with the generator output corresponding to about 2^{24} samples of (n,l), e.g. 2^{18} n values and 2^6 l values for each n value.

The test of Section 4 can be applied with about 2^{30} samples of (n,l), e.g. 2^{24} n values and 2^6 l values for each n value. Moreover, information about table S can be derived from this test with 2^{30} samples of (n,l) as well, and large amounts of information contained in table T can be derived from approximately 2^{32} samples of (n,l) or slightly more.

Despite of their relatively low time and space complexity, which enabled us to perform the computer simulations mentioned in Sections 4 and 5, the attacks reported in this paper do not seriously endanger the practical security of SEAL, because a too large amount of keystream samples (corresponding to more than 2^{30} (n,l) initialisation vectors) is required. These attacks suggest however that simple modifications of some design features of SEAL, e.g. the detail of the involvement of the IV-dependent values n_1 to n_4 in the second generator, would probably strengthen the algorithm without significant impact upon its performance.

7 Acknowledgements

We thank Thierry Baritaud and Pascal Chauvaud for the elaboration of the LaTeX 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.