Improving the Time Complexity of Matsui's Linear Cryptanalysis

B. Collard, F.-X. Standaert*, J.-J. Quisquater

UCL Crypto Group, Université Catholique de Louvain

Abstract. This paper reports on an improvement of Matsui's linear cryptanalysis that reduces the complexity of an attack with *algorithm 2*, by taking advantage of the Fast Fourier Transform. Using this improvement, the time complexity decreases from $O(2^k * 2^k)$ to $O(k * 2^k)$, where k is the number of bits in the keyguess. This improvement is very generic and can be applied against a broad variety of ciphers including SPN and Feistel schemes. In certain (practically meaningful) contexts, it also involves a reduction of the attacks data complexity (which is usually the limiting factor in the linear cryptanalysis of block ciphers). For illustration, the method is applied against the AES candidate Serpent and the speed-up is given for exemplary attacks.

Keywords: block ciphers, linear cryptanalysis, Fast Fourier Transform.

1 Introduction

The linear cryptanalysis [1] is one of the most powerful attacks against modern block ciphers in which an adversary exploits a linear approximation of the type:

$$P[\chi_P] \oplus C[\chi_C] = K[\chi_K] \tag{1}$$

In this expression, P, C and K respectively denote the plaintext, ciphertext and the secret key while $A[\chi]$ stands for $A_{a_1} \oplus A_{a_2} \oplus ... \oplus A_{a_n}$,with $A_{a_1}, ..., A_{a_n}$ representing particular bits of A in positions $a_1, ..., a_n$ (χ is usually denoted as a mask). In practice, linear approximations of block ciphers can be obtained by the concatenation of one-round approximations and such concatenations (also called characteristics) are mainly interesting if they maximize the deviation (or bias) $\epsilon = p - \frac{1}{2}$ (where p is the probability of a given linear approximation).

In its original paper, Matsui described two methods for exploiting the linear approximations of a block cipher, respectively denoted as *algorithm 1* and *algorithm 2*. In the first one, given an r-round linear approximation with sufficient bias, the algorithm simply counts the number of times the left side of Equation 1 is equal to zero for N pairs (plaintext, ciphertext). If $T > N/2$, then it assumes either $K[\chi_K] = 0$ if $\epsilon > 0$ or $K[\chi_K] = 1$ if $\epsilon < 0$ so that the experimental value $(T - N/2)/N$ matches the theoretical bias. If $T > N/2$, an opposite reasoning holds. For the attack to be successful, it is shown in [1] that the number of available (plaintext, ciphertext)-pairs must be proportional to $\frac{1}{\epsilon^2}$.

* Postdoctoral researcher of the Belgian Fund for Scientific Research (FNRS).

In the second method *algorithm 2*, an r-1-round characteristic is used and a partial decryption of the last round is performed by guessing the key bits involved in the approximation. As a consequence, all the guessed key bits can be recovered rather than the parity $K[\chi_K]$ which yields much more efficient attacks in practice. Moreover, as it uses a r-1-round characteristic instead of a r-round one for *algorithm 1*, it has a smaller data complexity. However, this improved efficiency has its counterpart in a higher computational complexity, due to the use of possibly large key guesses (*i.e.* involving a large number of key bits).

In this paper, we consequently introduce a general method for improving the time complexity of a linear cryptanalysis attack using *algorithm 2*. Although the limiting factor for linear cryptanalysis attacks is usually the data complexity, such an improvement is relevant and can be motivated both by practical and theoretical reasons, as the following scenarios underline.

- In the evaluation of linear cryptanalysis attacks against modern block ciphers, the attacker usually does a tradeoff between the bias of the approximation and the size of the keyguess. For example, when targeting 10 rounds of Serpent in [6], the authors found a 9-round approximation with bias 2^{-52} but with 92 bits of keyguess (23 active Sboxes). As this leads to an attack with time complexity $O(2^{184})$, they had to choose an approximation with a smaller bias (namely 2^{-58}) but only 44 bits of keyguess. Using our improvement, we can take advantage of the 2^{-52} bias with a time complexity of about $2^{98.5}$, thus reducing the data complexity from 2^{118} to 2^{106}.
- Since most recent ciphers (*e.g.* the AES candidates) have strong diffusion properties, the number of active S-boxes in linear cryptanalysis attacks against their reduced-round versions is usually too high for the time complexity of these attacks to be tractable. The improvement proposed in this paper can consequently be used to perform actual cryptanalytic experiments against these reduced-round ciphers. Therefore, we expect that it will lead to a better understanding of certain open issues, *e.g.* about the exploitation of multiple approximations in linear cryptanalysis.

Independently of these theoretical expectations, we believe that the proposed improvement of the time complexity, from $O(2^k * 2^k)$ to $O(k * 2^k)$ (where k is the number of bits in the keyguess) is meaningful in itself. We note finally that the idea of taking advantage of the Fast Fourier Transform to speed-up the computations in cryptanalysis is not new. For example [3] and [4] describe FFT-based techniques to improve correlation attacks against stream ciphers. However, we are not aware of any publication mentioning explicitly the applicability of the FFT to the linear cryptanalysis of block ciphers.

The rest of the paper is structured as follows. Section 2 describes a generic framework for the analysis of Matsui's linear cryptanalysis using *algorithm 2*. Section 3 details our improved key guess strategy and Section 4 applies our technique to improve some previous cryptanalytic results against the AES candidate Serpent. Finally, our conclusions are in Section 5.

2 General Framework for *algorithm 2*

Suppose a linear approximation on r rounds with bias ϵ, requiring $N \approx O(1/\epsilon^2)$ known plaintext-ciphertext pairs for a successful attack. Moreover, this approximation has q active S-boxes in the last round and k bits to guess. In the original *algorithm 2* proposed by Matsui, a partial decryption of the last round is performed for every ciphertext by guessing the key bits involved in the approximation. The parity of the approximation for the plaintext and the partially decrypted ciphertext is then evaluated and a counter corresponding to the guess is incremented if the relation holds, decremented otherwise. The key candidate with the highest counter in absolute value is finally assumed to be the correct key. As a partial decryption is proceeded for every ciphertext and every keyguess, the time complexity of this algorithm is in $O(N \cdot 2^k)$ partial decryptions.

However, as we only consider a limited number of bits (those in the active S-boxes) during the partial decryption of the ciphertexts, the same work is done many times. Indeed, the number of texts required to mount an attack is typically largely superior to the size of the keyguess (*i.e.* $N \gg 2^k$). On the basis of this observation, Matsui proposed in [2] an improvement which considerably reduces the time complexity of an attack. Although it was first applied to the DES, this improvement is valid in the general case. The modified algorithm can be divided in 2 phases (according to the framework proposed in [7] sec. 2.1):

Distillation phase (for each generated ciphertext)**:**

- Initialize an array of 2^k counters.
- For each generated ciphertext, extract the k-bit value corresponding to the active S-boxes and evaluate the parity of the plaintext subset defined by the approximation. Increment or decrement the counter corresponding to the extracted k-bit value according to the parity.

Analysis phase (once all the ciphertext have been generated)**:**

- For each k-bit ciphertext and k-bit subkey, partially decrypt the k-bit ciphertext under the k-bit subkey and evaluate the parity of the output subset (as defined by the linear approximation). Keep this value in a table of size $2^k \cdot 2^k$.
- For each k-bit subkey, evaluate its experimental bias by checking, for each k-bit ciphertext, the parity of the approximation and the value of the corresponding counter. Then output the subkey which has maximal bias.

During the distillation phase, we construct a table that indexes for each k-bit ciphertext, the difference between the frequency of its apparition leading to a null input parity and the frequency leading to a non-null input parity. This information is sufficient to evaluate the bias of the approximation for each key during the analysis phase. This process can be done "on the fly", while the plaintexts are being encrypted. As only simple operations like bit extractions and incrementations are performed during this phase, its complexity is generally assumed to be negligible compared to the one of the encryption process.

During the analysis phase, the actual bias for each subkey candidate is evaluated. In order to avoid multiple evaluations of the same operation, a table is constructed which indexes, for each k-bit ciphertext and each subkey candidate, the parity of the output subset obtained after the partial decryption of the ciphertext XORed with the subkey. For a given subkey candidate, its bias can then be evaluated by summing, for each k-bit ciphertext, the corresponding counter, taking the parity of the approximation for the given ciphertext and subkey into account (this parity is given by the sign of the counter and the correct index in the precomputed table). In this way, the table is accessed 2^k times for each possible subkey, leading to a total time complexity of $O(2^k \cdot 2^k)$, compared to the $O(N \cdot 2^k)$ operations for a naive implementation of *algorithm 2*. Importantly, this complexity depends only on the number of subkey candidates and not on the number of texts used. Note finally that the table can be computed row by row in order to save memory space.

3 Improving the framework

In this section, we present a simple but powerful modification of the above algorithm that allows us to significantly decrease the time complexity of an attack. As the modification concerns only the analysis phase, the distillation phase remains unchanged and so does the data complexity.

3.1 Rewriting the algorithm

The table defined during the analysis phase can be seen as a large matrix $\boldsymbol{C}$ of size $2^k \cdot 2^k$ defined by the following function:

$$\boldsymbol{C}(i,j) = parity(\mathrm{S}^{-1}(i \oplus j)) \qquad (0 \le i,j \le 2^k - 1) \tag{2}$$

where $\mathrm{S}^{-1}(l)$ represents the inverse of the last layer of S-boxes for the k-bit digit l and $parity()$ is a function mapping any k-bit subset to ± 1 according to its parity ($+1$ if the parity of the subset is zero, -1 otherwise). With such a definition, the bias ϵ_i corresponding to a particular keyguess i is given by the equation:

$$\epsilon_i = \sum_{j=0}^{2^k-1} parity(\mathrm{S}^{-1}(i \oplus j)) \cdot \boldsymbol{x}(j) = \sum_{j=0}^{2^k-1} \boldsymbol{C}(i,j) \cdot \boldsymbol{x}(j) = \boldsymbol{C}(i,:) \cdot \boldsymbol{x} \tag{3}$$

where $\boldsymbol{x}$ is the vector of counters such as defined in the distillation phase. This equation evaluates the bias of the linear approximation for a particular k-bit subkey candidate i. Consequently, the vector $\boldsymbol{\epsilon}$ of the experimental bias for every subkey candidates can be computed by the matrix-vector product:

$$\boldsymbol{\epsilon} = \boldsymbol{C} \cdot \boldsymbol{x} \tag{4}$$

At this point, the complexity for the evaluation of the experimental biases is still in $O(2^k \cdot 2^k)$ as it implies a matrix-vector product with size 2^k.

3.2 Analysis of the new algorithm

We underline the fact that the matrix $\boldsymbol{C}$ has a very particular structure. Taking this structure into account will allow us to significantly reduce the number of operations required to evaluate the vector $\boldsymbol{\epsilon}$ of the bias.

First, as $\boldsymbol{C} = f(i \oplus j)$ for a known function f, every rows or column of $\boldsymbol{C}$ defines the complete matrix (in particular, $\boldsymbol{C}$ is symmetric). For example,

$$\boldsymbol{C}(i,j) = f(i \oplus j) = f((i \oplus j) \oplus 0) = \boldsymbol{C}(i \oplus j, 0) \tag{5}$$

Let us introduce the following definitions (*cfr.* [10]):

Definition 1 (circulant). *A matrix is circulant iff each row (column) vector is rotated one element to the right relative to the preceding row (column) vector.*

Definition 2 (block circulant). *A matrix is m-block circulant iff it is circulant blockwise and the number of blocks in each row (or column) is m.*

Definition 3 (level circulant).

- *(1) A matrix is level-*1 *circulant with type(n) iff it is circulant of size n.*
- *(2) A matrix is level-*2 *circulant with type(m,n) iff it is m-block-circulant and each block is a circulant of size n itself.*
- *(3) A matrix is level-*3 *circulant with type(m,n,o) iff it is a m-block circulant whose blocks are level* 2 *circulant with type (n,o).*
- *(q) A matrix is level-q circulant with type(m,n,o,...) iff it is a m-block circulant whose blocks are level* $q-1$ *circulant with type (n,o,...).*

Proposition 1. *Let* $\boldsymbol{C}(i,j) = f(i \oplus j)$*, then* $\boldsymbol{C}$ *is level-k circulant with type* $\underbrace{(2,2,...,2)}_{k \quad times}$.

Demonstration 1. *Let us define the matrix* $\boldsymbol{M}$ *of size* $(2^k * 2^k)$ *as :* $\boldsymbol{M}(i,j) = i \oplus j (0 \le i,j \le 2^k - 1)$*. Let us divide* $\boldsymbol{M}$ *in 4 blocks with size* $(2^{k-1} * 2^{k-1})$ *each:*

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M_{11}} & \boldsymbol{M_{12}} \\ \boldsymbol{M_{21}} & \boldsymbol{M_{22}} \end{pmatrix}$$

Then for $(0 \le a,b \le 2^{k-1} - 1)$*:*

- $\boldsymbol{M_{11}}(a,b) = \boldsymbol{M}(a,b) = a \oplus b,$
- $\boldsymbol{M_{12}}(a,b) = \boldsymbol{M}(a, b + 2^{k-1}) = a \oplus b \oplus 2^{k-1},$
- $\boldsymbol{M_{21}}(a,b) = \boldsymbol{M}(a + 2^{k-1}, b) = a \oplus 2^{k-1} \oplus b,$
- $\boldsymbol{M_{22}}(a,b) = \boldsymbol{M}(a + 2^{k-1}, b + 2^{k-1}) = a \oplus 2^{k-1} \oplus b \oplus 2^{k-1},$

This is true because $a + 2^{k-1}$ *is equivalent to* $a \oplus 2^{k-1}$ *since* $0 \le a \le 2^{k-1} - 1$*. Consequently,* $\boldsymbol{M_{11}} = \boldsymbol{M_{22}}$ *and* $\boldsymbol{M_{12}} = \boldsymbol{M_{21}}$*, thus* $\boldsymbol{M}$ *is 2-block circulant. Moreover,* $\boldsymbol{M_{12}} = \boldsymbol{M_{11}} \oplus 2^{k-1}$*, so it has the same circulant structure as* $\boldsymbol{M_{11}}$*. We can repeat the same reasoning for* $\boldsymbol{M} = \boldsymbol{M_{11}}$ *with* $k = k-1$*, and so the proposition is proved by induction. Finally, it is obvious that* $f(\boldsymbol{M})$ *keeps the circulant properties if* $f()$ *is applied elementwise.*

3.3 Fast algorithm

We now describe how the properties given above can be used to speed up the linear cryptanalysis. We exploit the following result (*cfr.* [10] for a proof):

Theorem 1. *A circulant $\boldsymbol{C}$ of level k and type (m,n,o,...,r) is diagonalizable by the unitary matrix $\boldsymbol{F} = \boldsymbol{F_m} \otimes \boldsymbol{F_n} \otimes \boldsymbol{F_o} \otimes ... \otimes \boldsymbol{F_r}$:*

$$\boldsymbol{C} = \boldsymbol{F}^* diag(\boldsymbol{\lambda})\boldsymbol{F}, \tag{6}$$

where $\boldsymbol{\lambda}$ is the vector of eigenvalues of $\boldsymbol{C}$, The symbol $\otimes$ is the Kronecker product and F_n is the Fourier matrix of size $n * n$ defined by:

$$\boldsymbol{F_n}(i,j) = \frac{1}{\sqrt{n}} w^{i \cdot j} \quad (0 \leq i, j \leq n-1), \tag{7}$$

with:

$$w = e^{\frac{2\pi\sqrt{-1}}{n}} \tag{8}$$

The matrix $\boldsymbol{F}$ is the k-dimensional Discrete Fourier Transform matrix. Therefore, the multidimensional Fast Fourier Transform allows us to quickly compute the matrix-vector product with $\boldsymbol{F}$ or $\boldsymbol{F}^*$. Using the FFT, the complexity of this product decrease from $O(n^2)$ to $O(n\ log_2(n))$ [9].

Proposition 2. *The eigenvalues vector $\boldsymbol{\lambda}$ of a circulant matrix $\boldsymbol{C}$ of level k and type (m,n,o,...,r) can be computed with the following matrix-vector product:*

$$\boldsymbol{\lambda} = \boldsymbol{F}\boldsymbol{C}(:,1)\sqrt{mno...r}, \tag{9}$$

where $\boldsymbol{C}(:,1)$ means (using Matlab notation) we take the first column of $\boldsymbol{C}$.

Demonstration 2. *From theorem 1, it follows that:*

$$\boldsymbol{C} = \boldsymbol{F}^* diag(\lambda)\boldsymbol{F} \tag{10}$$

Multiplying both sides by $\boldsymbol{F}$, this gives (as $\boldsymbol{F}\boldsymbol{F}^* = \boldsymbol{I}$):

$$\boldsymbol{F}\boldsymbol{C} = diag(\lambda)\boldsymbol{F} \tag{11}$$

If we consider the first column only, this reduces to:

$$(\boldsymbol{F}\boldsymbol{C})(:,1) = (diag(\lambda)\boldsymbol{F})(:,1) = \boldsymbol{\lambda} \circ \boldsymbol{F}(:,1) \tag{12}$$

From equation 7 and the definition of $\boldsymbol{F}$, it follows that:

$$\boldsymbol{F}(:,1) = \frac{1}{\sqrt{mno...r}}(1,1,1...1)^T \tag{13}$$

Consequently,

$$\boldsymbol{\lambda} = (\boldsymbol{F}\boldsymbol{C})(:,1)\sqrt{mno...r} = \boldsymbol{F}\boldsymbol{C}(:,1)\sqrt{mno...r} \tag{14}$$

Hence, the matrix-vector product $\boldsymbol{\epsilon} = \boldsymbol{Cx}$ is equivalent to $\boldsymbol{\epsilon} = \boldsymbol{F}^* diag(\boldsymbol{\lambda})\boldsymbol{Fx}$. The eigenvalues of $\boldsymbol{C}$ can be computed using the formula: $\boldsymbol{\lambda} = \boldsymbol{FC}(:,1)\sqrt{2^k}$. Therefore, the matrix-vector product can be computed using the three following matrix-vector products: $\boldsymbol{y} = \boldsymbol{Fx}$, $\boldsymbol{z} = \boldsymbol{FC}(:,1)\sqrt{2^k}$ and $\boldsymbol{\epsilon} = \boldsymbol{F}^*(\boldsymbol{z} \circ \boldsymbol{y})$ (where $\circ$ is the Hadamard product). As each of these three products involves the matrix $\boldsymbol{F}$, the complete computation can be made by applying only three k-dimension FFTs of size 2^k, leading to a complexity of $3 \cdot 2^k \cdot log_2(2^k) = 3 \cdot k \cdot 2^k$. The Matlab implementation code for this improved strategy is given below (see Algorithm 1). As a typical numerical example, for a $2^{20} * 2^{20}$ matrix $\boldsymbol{C}$, the matrix-vector product is computed in less than 5 seconds on a Pentium D 3.20GHz.

Algorithme 1 Matlab code

```
function b=product(C,x)

% compute the product Cx by the mean of the fft
% C is a level k circulant matrix of type (2,2,2,...,2).
% C is completely specified by its first column
% x is an unspecified vector

k= log2(size(C,1));

% compute F*x:
x= reshape(x,2*ones(k));
prod1= fftn(x);
prod1= reshape(prod1,2^k,1);

% compute the eigenvalues of C
c= reshape(C(:,1), 2*ones(k));
eig= fftn(c);
eig= reshape(eig,2^k,1);

% compute eig*F*x
prod2= eig.*prod1;

% compute b=F'*eig*F*x
b= reshape(prod2,2*ones(k));
b= ifftn(b);
b= reshape(b,2^k,1);

return b;
```

3.4 Implication for multiple linear approximations

In context of multiple linear approximations (*cfr.* [11], [7]), more speed-up may be achievable. Every input mask defines its own vector of counter $\boldsymbol{x}$, while every output mask defines a different matrix $\boldsymbol{C}$. Consequently, the use of multiple approximations with the same active S-boxes but different input masks (as it is usually the case) requires to compute the eigenvalues only once, as the matrix $\boldsymbol{C}$ remains the same. Thus, the time complexity of linear cryptanalysis with n approximations is reduced to the computation of $2n+1$ FFTs instead of $3n$ FFTs. This involves an additional reduction of up to 33% for the time complexity.

3.5 Extension to key additions *modulo* 2^k

In certain ciphers, the mixing with the key material is done using a modular addition instead of a XOR with the subkey (practical examples include [12], [13]). A similar approach as in the previous sections can be applied to reduce the time complexity of such systems.

Definition 4 (left-circulant). *A matrix is left-circulant iff each row (column) vector is left-rotated by one element relative to the preceding row (column) vector.*

Proposition 3. *Let* $\boldsymbol{C_{left}}(i,j) = f(i+j \ mod \ 2^k)$ $(0 \leq i, j \leq 2^k-1)$, *then* $\boldsymbol{C_{left}}$ *is left-circulant.*

Demonstration 3. *For every* λ, $\boldsymbol{C_{left}}(a+\lambda, b-\lambda) = f(a+\lambda+b-\lambda \ mod \ 2^k) = f(a+b \ mod \ 2^k)$*. Thus, all the element in the same increasing diagonal of the matrix are equal. Moreover, the value in a diagonal is repeated every* 2^k *diagonal due to the mod* 2^k*, and so the matrix is left-circulant.*

We can easily convert a left-circulant matrix $\boldsymbol{C_{left}}$ to a circulant matrix $\boldsymbol{C}$ with the same first row thanks to a particular matrix of permutation $\boldsymbol{\Gamma}$:

$$\boldsymbol{C_{left}} = \boldsymbol{\Gamma}\boldsymbol{C}, \tag{15}$$

where:

$$\boldsymbol{\Gamma} = \begin{pmatrix} 1 \, 0 \, 0 \ldots 0 \, 0 \\ 0 \, 0 \, 0 \ldots 0 \, 1 \\ 0 \, 0 \, 0 \ldots 1 \, 0 \\ \vdots \, \vdots \, \vdots \quad\ \ \vdots \, \vdots \\ 0 \, 1 \, 0 \ldots 0 \, 0 \end{pmatrix}$$

This requires the application of 2^k row permutations and it is useful for the following result that we require in our investigations (see [10] pp.72-73):

Theorem 2. *A circulant* $\boldsymbol{C}$ *is diagonalizable by the Fourier matrix* $\boldsymbol{F}$ *of size* 2^k*:*

$$\boldsymbol{C} = \boldsymbol{F}^* diag(\boldsymbol{\lambda})\boldsymbol{F}, \tag{16}$$

where $\boldsymbol{\lambda}$ *is the vector of eigenvalues of* $\boldsymbol{C}$*.*

Again, we can easily compute the matrix-vector product between $\boldsymbol{C}$ and $\boldsymbol{x}$ using 3 (one-dimensional) FFTs. The algorithm is roughly the same as described above, except that the multi-dimensional FFTs are replaced by one-dimensional FFTs and that we must perform a last permutation $\boldsymbol{\Gamma}$ in the end (in order to switch from a right- to a left-circulant matrix). Finally, for a matrix $\boldsymbol{C_{left-toeplitz}}$:

$$\boldsymbol{C_{left-toeplitz}}(i,j) = f(i + j \ mod \ 2^m); \ (0 \leq i,j \leq 2^k - 1), \ m \neq k, \tag{17}$$

with the left-Toeplitz structure, it can be shown that the matrix-vector product can be computed by embedding the $2^k * 2^k$ matrix in a $2^{k+1} * 2^{k+1}$ matrix with left-circulant structure, leading to a complexity of $O(2 * (k+1) * 2^k)$.

4 Practical improvements

The improved algorithms described above can be straightforwardly applied to improve previous cryptanalytic results. For illustration purposes, we applied them to the AES candidate Serpent [5]. Using the FFT to compute the linear approximation biases for the subkey candidates allows speeding-up the attack of Biham *et al.* as summarized in Table 1. In Table 2, we additionally report on the improved linear and multiple linear cryptanalysis of Serpent, described in [8].

Rounds	Type of attack	complexity		
		data	time	memory
10	Lin.Cryptanalysis[6]	2^{118}KP	$2^{88} \rightarrow 2^{51}$	2^{44}
	Lin.Cryptanalysis[6]	2^{116}KP	$2^{96} \rightarrow 2^{55.2}$	2^{48}
	Lin.Cryptanalysis[6]	2^{106}KP	$2^{184} \rightarrow 2^{100.1}$	2^{92}
11	Lin.Cryptanalysis[6]	2^{118}KP	$2^{214} \rightarrow 2^{148.7}$	$2^{88} \rightarrow 2^{140}$
KP - Known Plaintexts Complexity is measured in number of arithmetic operation. Memory is mesured in Bytes.				

Table 1. Previous and improved attacks on Reduced-rounds Serpent.

We note that, as previously mentioned, certain improvements are particularly relevant with respect to the experimental testing of the attacks. For example, targeting 7 rounds of Serpent with a multiple linear cryptanalysis attack appears as a reasonable target thanks to time complexity reduction. Using dedicated hardware like *Copacobana* [14], it could even be possible to attack up to 8-round Serpent. The reduced time complexity also allows considering the exploitation of better biased linear approximations with larger key guesses and therefore to reduce the data complexity of the best reported attacks. For example, Table 1 includes the scenario described in the introduction of this paper: moving from a time complexity of 2^{184} to a time complexity of 2^{100} allows to reduce the attack data complexity from 2^{118} to 2^{106}. This example clearly emphasizes the practical impact of our result on the overall complexity of linear cryptanalysis attacks.

Rounds	Type of attack	complexity		
		data	time	memory
7	Lin.cryptanalysis	2^{52}KP	$2^{40} \to 2^{25.9}$	2^{20}
	Mult.Lin.Cryptanalysis(8 appr.)	2^{47}KP	$2^{43} \to 2^{28.4}$	2^{23}
8	Lin.cryptanalysis	2^{62}KP	$2^{56} \to 2^{34.4}$	2^{28}
	Mult.Lin.Cryptanalysis(8 appr.)	2^{57}KP	$2^{59} \to 2^{36.9}$	2^{31}
	Mult.Lin.Cryptanalysis(104 appr.)	2^{55}KP	$2^{62.7} \to 2^{40.5}$	$2^{34.7}$
9	Lin.cryptanalysis	2^{80}KP	$2^{88} \to 2^{51}$	2^{44}
	Mult.Lin.Cryptanalysis(128 appr.)	2^{71}KP	$2^{95} \to 2^{57.5}$	2^{51}
	Mult.Lin.Cryptanalysis(3712 appr.)	2^{68}KP	$2^{99.9} \to 2^{62.3}$	$2^{55.9}$
10	Lin.cryptanalysis ($\epsilon = 2^{-55}$)	2^{112}KP	$2^{88} \to 2^{51}$	2^{44}
	Mult.Lin.Cryptanalysis(2048 appr.)	2^{99}KP	$2^{99} \to 2^{61.5}$	2^{55}
	Lin.cryptanalysis ($\epsilon = 2^{-59}$)	2^{120}KP	$2^{64} \to 2^{38.6}$	2^{32}
	Mult.Lin.Cryptanalysis(2048 appr.)	2^{107}KP	$2^{75} \to 2^{49}$	2^{43}
11	Lin.cryptanalysis ($\epsilon = 2^{-58}$)	2^{118}KP	$2^{178} \to 2^{116.3}$	2^{108}

Table 2. Additional improved attacks on Reduced-rounds Serpent (see [8]).

Additionally to the results presented in [8], Table 2 includes an attack against 11-round Serpent. We use a 9-round linear approximation starting and ending with S-box 9. This approximation was generated similarly to the ones presented in [8]. It has a bias of 2^{-58} and a total of 27 active S-boxes (15 in the first round and 12 in the last round). The attack follows the same principle as the ones presented so far, except that we must also perform a partial encryption in the beginning of the cipher. We first define an array $\boldsymbol{x}$ of 2^{108} counters in the following way: for each plaintext-ciphertext pair, we extract the 108-bit value corresponding to the active S-boxes and we increment the corresponding counter. Then we define a matrix $\boldsymbol{C}$ of size $2^{108} * 2^{108}$ as:

$$\boldsymbol{C(i,j)} = parity(S(i_{1:60} \oplus j_{1:60})||S^{-1}(i_{61:108} \oplus j_{61:108})) \tag{18}$$

That is to say, $\boldsymbol{C(i,j)}$ is the parity of the linear approximation after partial en/decryption of the 108-bit text i with 108-bit subkey j. As seen previously, the experimental bias for any keyguess is given by the matrix-vector product $\boldsymbol{C} \cdot \boldsymbol{x}$. Thanks to the level circulant structure of $\boldsymbol{C}$, the time complexity is equal to $3 \cdot 108 \cdot 2^{108} = 2^{116}$. Without this trick, the time complexity would have been $2^{88} + 2^{60} \cdot (2^{118} + 2^{88}) = 2^{178}$ (see [6] for the details). As a comparison, the best-reported attack on 11-round Serpent uses a combination of linear and differential cryptanalysis techniques [15]. It has a data complexity of $2^{125.3}$ choosen plaintexts, $2^{139.2}$ encryptions and 2^{60} bytes of memory.

5 Conclusion and further works

In this paper, we presented an improvement of Matsui's linear cryptanalysis that reduces the time complexity of an attack using *algorithm 2* from $O(2^k * 2^k)$ to $O(k * 2^k)$ partial decryptions, where k is the number of bits in the keyguess. Moreover, in the case of multiple linear cryptanalysis, additional speed-ups can be reached. This improvement is very generic and can be applied against a broad variety of ciphers including SPN and Feistel schemes. For illustration purposes, we applied the method to the block cipher Serpent and exhibited the reduced complexities of some (state-of-the-art) exemplary attacks. As a scope for further research, the exploitation of the improved time complexity of attacks using multiple linear approximations should allow performing actual experiments and therefore evaluate the validity of certain assumptions detailed in [7].

Acknowledgements

The authors thank Pr. Paul Van Dooren from UCL\INMA for his helpful advices. We also thank anonymous reviewers from SAC and ICISC for their comments.

References

1. M. Matsui, *Linear cryptanalysis method for DES cipher*, in the proceedings of Eurocrypt 1993, LNCS, vol. 765, pp. 386-397, Lofthus, Norway, May 1993.
2. M. Matsui, *The First Experimental Cryptanalysis of the Data Encryption Standard*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Science, vol. 839, pp. 1-11, Santa Barbara, California, USA, August 1994.
3. P. Chose, A. Joux, M. Mitton, *Fast Correlation Attacks: An Algorithmic Point of View*, in the proceedings of Eurocrypt 2002, Lecture Notes in Computer Science, vol. 2332, pp.209-221, Amsterdam, The Netherlands, May 2002.
4. Y. Lu, W. Meier, S. Vaudenay, *The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption*, in the proceedings of CRYPTO 2005, LNCS, vol. 3621, pp. 97-117, Santa Barbara, California, USA, August 2005.
5. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, in the proceedings of the First Advanced Encryption Standard (AES) Conference, Ventura, CA, 1998.
6. E. Biham, O. Dunkelman, N. Keller, *Linear Cryptanalysis of Reduced Round Serpent*, in the Proceedings of FSE 2001, Lecture Notes in Computer Science, vol. 2355, pp. 16-27, Yokohama, Japan, April 2001.
7. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of CRYPTO 2004, Lecture Notes in Computer Science, vol. 3152, pp.1-22, Santa Barbara, California, USA, August 2004.
8. B. Collard, F.-X. Standaert, J.-J. Quisquater, *Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent*, to appear in the proceedings of InsCrypt 2007.
9. J.W. Cooley and J.W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comput. 19, pp. 297301, 1965.
10. P.J. Davis, *Circulant Matrices*, Wiley-Interscience, pp. 176-191, NY, 1979.
11. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of CRYPTO 1994, Lecture Notes in Computer Sciences, vol. 839, pp. 26-39, Santa Barbara, California, USA, August 1994.
12. X. Lai, J.L. Massey, *A Proposal for a New Block Encryption Standard*, in the proceedings of Eurocrypt 1990, Lecture Notes in Computer Science, vol. 473, pp. 389-404, Aarhus, Denmak, May 1990.
13. D.J. Wheeler, R.M. Needham, *TEA, a Tiny Encryption Algorithm*, in the proceedings of Fast Software Encryption 1994, Lecture Notes in Computer Science, vol. 1008, pp. 363-366, Leuven, Belgium, December 1994.
14. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, M. Schimmler, *Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker*, in the proceedings of Cryptographic Hardware and Embedded Systems - Ches 2006, Lectrure Notes in Computer Science, vol.4249, Springer, 2006.
15. E. Biham, O. Dunkelman, N. Keller, *Differential-linear Cryptanalysis of Serpent*, in the Proceedings of Fast Software Encryption 2003, Lecture Notes in Computer Science, vol. 2887, pp. 9-21, Springer, 2004.