White Box Cryptography: a new attempt

Julien Bringer[1], Hervé Chabanne[1], and Emmanuelle Dottax*

[1] Sagem Défense Sécurité

Abstract. At CMS 2006 Bringer *et al.* show how to conceal the algebraic structure of a "traceable block cipher" by adding perturbations to its description. We here exploit and strengthen their ideas by further perturbing the representation of a cipher towards a white box implementation. Our technique is quite general, and we apply it – as a challenging example in the domain of white box cryptography – to the block cipher AES.

Key words: white box cryptography, obfuscation, perturbations, AES.

1 Introduction

In this work, we aim at protecting the design of a block cipher, as well as any secrets involved in the computations, while the whole implementation is actually available to attackers, i.e. in a *white box* environment. *White box cryptography* is indeed intended to provide a practical level of protection of software implementations on an untrusted host. The main constraint is that the result must be directly executable.

Here, the attacker has in his possession the full implementation of a keyed block cipher and he looks for the key. The general idea to secure an implementation is to add one or more input and output encodings resulting in a obfuscated function that is harder to analyse or tamper with. Chow *et al.* introduce this idea and propose a white box implementation of DES in [4] by interleaving affine transformations and using de-linearization techniques. An improvement is explained in [10]. An implementation of AES is also given in [5] by representing it with a set of key-dependent look-up tables. Some attacks that can exploit the non-linear properties of S-boxes have also been proposed regarding these attempts, such as the statistical bucketing attack and the injection of faults (e.g. see [9, 10]) for DES implementation or by the analysis of look-up tables composition for AES [2]. Our approach here is different since, in

* Work done while the third author was at Sagem Défense Sécurité.

addition to mixing transformations, we introduce some perturbations in the cipher.

In [1], Billet and Gilbert proposed a traceable block cipher. The idea was to represent and implement the same instance of a cipher in different ways. An authority, which knows secret components of the cipher, is able to distinguish the different instances, while attackers are unable to forge untraceable representations. The security of this scheme is based on the Isomorphism of Polynomials (IP) problem [12]. In [8], algorithms to solve this problem were proposed, and a challenging example of [1] has been broken.

In [7], the idea of introducing perturbations to reinforce an IP-based cryptosystem is introduced. In a similar approach, Bringer *et al.* show in [3] how to perturb the representation of the traceable block cipher by Billet and Gilbert to move it away from these attacks [8]. The principle of the protection is to add *perturbations* to the original equations in order to dissimulate the algebraic structure, which is actually needed to mount the attack. Based on this work, we here propose new ideas which should help protecting better. Particularly, we add specific terms to the first round which are cancelled only in the last round such that every round is tightly linked to the others. As a challenging example, we give an application of our techniques to the AES block cipher, and propose an implementation in a *white box cryptography* approach.

The paper is organized as follows. In Sect. 2, we give a brief recall of the principle of the protection and of the perturbations introduced by [3]. In Sect. 3, we present our modifications of these previous protections. In Sect. 4, we give practical implementations of our ideas to AES for a white box representation. In Sect. 5 we analyse the security of this new representation. Finally, we conclude in Sect. 6.

2 Bringer *et al.*'s perturbations [3]

The traceable block cipher introduced by Billet and Gilbert [1] consists of rounds, $G_{i,j}$ denoting the system of equations of user j's representation of the i-th round. The attack of [8] uses the algebraic structure of this system to recover the secret components, thus the aim of [3] is to hide this structure (or merely to modify it). To describe the protection proposed by [3], we first have to introduce some notations. Let $\tilde{0}$ be a polynomial which "often" vanishes – or, say, in a controlled way – and $\tilde{P} = P + \tilde{0}$, where P is a polynomial. By the way, $\tilde{S}$ stands for a system S of equations where such substitutions are made, replacing some equations P by $\tilde{P}$.

The idea of [3] is simply to replace the original systems of polynomials $G_{i,j}$ by $\widetilde{G_{i,j}}$ and to insert random linear transformations between two rounds to hide this modification. Then, the algebraic structure of each round is made less accessible to an attacker, and the attack [8] seems hard to apply anymore since the description of the original system is not available.

On the other hand, as the new system does not always give the right result anymore, the system $G_{i,j}$ has to be replaced by distinct, concurrent systems $\widetilde{G_{i,j}}$ with correlated polynomials $\tilde{0}$, such that a majority vote allows to decide which result has to be kept.

The way polynomials $\tilde{0}$ are constructed is recalled below as they are part of our new protection. They start with well-known linearized polynomials:

Definition 1. *Let q_0 a power of 2 and $q = q_0^m$,*

- a q_0-polynomial *over $GF(q)$ is a polynomial of the form $L(X) = \sum_{i=0}^{e} a_i X^{q_0{}^i}$, with $e \in \mathbb{N}$ and $(a_0, \ldots, a_e) \in GF(q)^{e+1}$,*
- an affine q_0-polynomial *over $GF(q)$ is a polynomial of the form $A(X) = L(X) - \alpha$ where $\alpha \in GF(q)$ and L is a q_0-polynomial.*

Proposition 1. *Let L be a q_0-polynomial, then the set of its roots is a linear subspace of its splitting field, i.e. $L(X) = \prod_{\alpha \in V} (X - \alpha)^{\kappa}$ for V a linear subspace and some $\kappa \geq 1$. In fact, for a q_0-polynomial with simple roots, $\kappa = 1$.*

The construction of a polynomial $\tilde{0}$ given in [3] is then the following:

1. let Q be an affine q_0-polynomial over $GF(q_0^m)$ which equals zero over a subspace U of dimension e,
2. a multivariate version of Q is made by choosing a random multivariate polynomial with a small number of terms $f \in GF(q_0^m)[X_1, \ldots, X_{n_f}]$, computing $Q_f = Q(f(X_1, \ldots, X_{n_f}))$ and checking if at least $1/2^{m-e}$ points of $GF(q_0^m)^{n_f}$ have an image following f in U,
3. then $\tilde{0} = L(f(X_1, \ldots, X_{n_f}))H(X_1, \ldots, X_n)$ where H is a random polynomial in n variables over $GF(q)$ chosen s.t. $\tilde{0}$ has at least some monomials of chosen shapes.

The resulting polynomial vanishes at least on $1/2^{m-e}$ points.

3 Perturbing a block cipher to hide its structure

3.1 Overview

We present a generalization of the ideas above by modifying rounds of a block cipher, when these rounds are expressed as a system of polynomial equations in a finite field. It leads to a polynomial white box implementation.

For this, we introduce additional terms that are mixed to the ones of the original system by linear transformations:

- we introduce random variables in every round (but the last one), they are here to dissimulate the information;
- we also add specific terms to the first round, that will often take some predetermined value. These terms generalize the $\tilde{0}$ described in the previous section. They are carried round after round, up to the last one;
- finally, corresponding polynomials – chosen for vanishing when the predetermined value is reached – are added to the last round.

Thus doing, intermediate results given by rounds are 'false' with a high probability as the specific terms added in the first round perturb all the intermediate values until they are cancelled in the last round.

3.2 Notations

Now, we work with a generic block cipher and in the remainder of the paper, we will adopt the following notations:

- $X = (x_1, \ldots, x_n)$ denotes the input variable of the cipher,
- $Y_i^1 = (y_{i,1}^1, \ldots, y_{i,n}^1)$ the output variable of the i-th round.

These variables lie in a finite field and each operation – including S-boxes – appearing in the round can be expressed as a polynomial, obtained by combination of its component functions. This way, we compute the system of equations representing the whole round and denote by S_i the system for the i-th round of the cipher.

We get $Y_1^1 = S_1(X)$ and, for the other rounds, $Y_i^1 = S_i(Y_{i-1}^1)$. We also introduce the following variables, associated to the i-th round:

$$\begin{aligned} Y_i^2 &= (y_{i,1}^2, \ldots, y_{i,s}^2), \\ Y_i^3 &= (y_{i,1}^3, \ldots, y_{i,t_i}^3), \\ Z_i &= (z_{i,1}, \ldots, z_{i,n+s+t_i}). \end{aligned}$$

In the new representation, the i-th round outputs $n + s + t_i$ variables, written Z_i. We now describe the operations implemented in the modified rounds for a cipher consisting of R rounds. We distinguish the first and the last ones.

3.3 Modifying the representation of each round

First round. The first round consists in the execution of the following operations (see. Fig. 1):

1. $Y_1^1 = S_1(X), \quad Y_1^2 = \widetilde{\Phi}(X), \quad Y_1^3 = R_1(X),$
2. $Z_1 = M_1(Y_1^1, Y_1^2, Y_1^3),$

where:

- S_1 is the system corresponding to the first round of the original cipher;
- $\widetilde{\Phi}$ is a system of s polynomials taking "often" a value $(\varphi_1, \ldots, \varphi_s)$ and constructed as $\widetilde{\Phi}(X) = (\tilde{0}(X) + \varphi_1, \ldots, \tilde{0}(X) + \varphi_s)$ for a given polynomial $\tilde{0}$ (e.g. as in Sec. 2);
- R_1 is a random system of t_1 polynomial equations;
- M_1 is a linear bijection, represented by an $n + s + t_1$ square matrix.

The first round is implemented as the composition of these operations, i.e. by a system of $n + s + t_1$ equations: $Z_1 = S_1'(X)$.

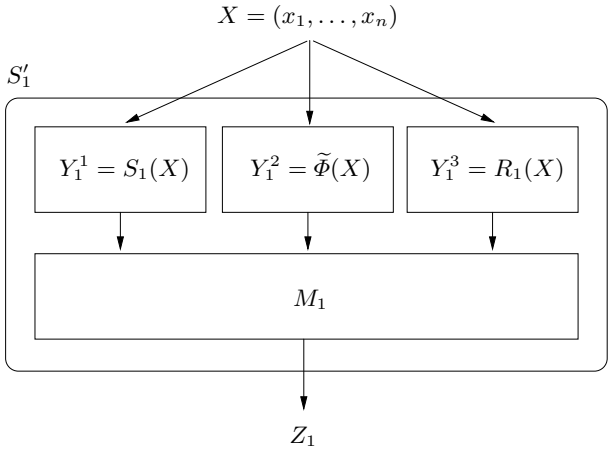

Fig. 1. Protected representation of first round.

Following rounds. The following rounds, except the last one, consist in the following operations (see. Fig. 2). For $i \in \{2, \ldots, R-1\}$,

1. $(Y_{i-1}^1, Y_{i-1}^2, Y_{i-1}^3) = M_{i-1}^{-1}(Z_{i-1})$,
2. $Y_i^1 = S_i(Y_{i-1}^1), \quad Y_i^2 = Y_{i-1}^2, \quad Y_i^3 = R_i(Z_{i-1})$,
3. $Z_i = M_i(Y_i^1, Y_i^2, Y_i^3)$,

where:

- S_i is the system corresponding to the i-th round of the original cipher;
- R_i is a new random system of t_i polynomial equations;
- M_i is a linear bijection, represented by an $n + s + t_i$ square matrix.

The i-th round is then implemented as the composition of these operations: $Z_i = S_i'(Z_{i-1})$.

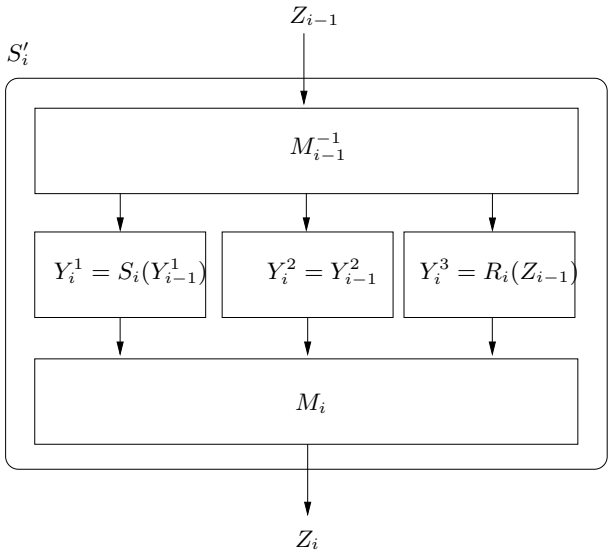

Fig. 2. Protected representation of inside rounds.

Last round. For the last round we execute (see Fig. 3):

1. $(Y_{R-1}^1, Y_{R-1}^2, Y_{R-1}^3) = M_{R-1}^{-1}(Z_{R-1})$,
2. $Y_R^1 = S_R(Y_{R-1}^1), \quad Y_R^2 = O_\Phi(Y_{R-1}^2)$,
3. $Z_R = Y_R^1 + Y_R^2$,

where:

- S_R is the system corresponding to the last round of the original cipher;
- O_Φ is a system of n polynomial equations that vanishes in $(\varphi_1, \ldots, \varphi_s)$.

Last round is implemented as the system representing the composition of these operations: $Z_R = S_R'(Z_{R-1})$.

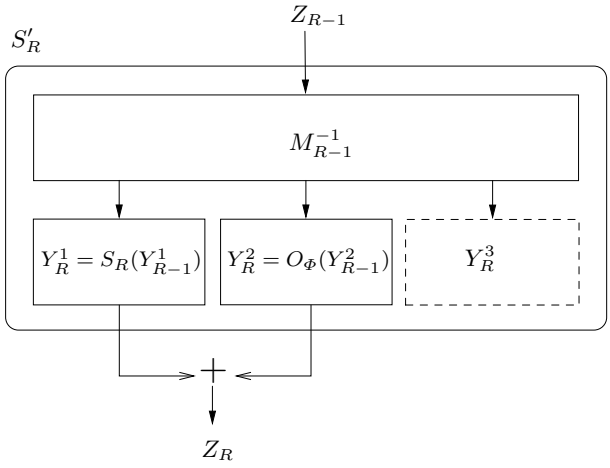

Fig. 3. Protected representation of last round.

3.4 Getting the right result

As we see, the resulting algorithm gives the right result (i.e. the result that would be given by the original algorithm) only when the system $\widetilde{\Phi}$ actually takes the value $(\varphi_1, \ldots, \varphi_s)$. So several instances should be implemented, with well chosen $\widetilde{\Phi}$'s, so that the right result can be deduced from a majority vote (cf. Sec. 4.4).

Remark 1. Note that, when the finite field is $GF(2)$ (the cipher can always be expressed by boolean equations), one instance of the protected cipher is enough. Indeed, the system $\widetilde{\Phi}(X) = (\tilde{0}(X) + \varphi_1, \ldots, \tilde{0}(X) + \varphi_s)$ will take only two different values, $(\varphi_1, \ldots, \varphi_s)$ and $(\varphi_1 + 1, \ldots, \varphi_s + 1)$. By taking O_Φ such that $O_\Phi(\varphi_1, \ldots, \varphi_s) = O_\Phi((\varphi_1 + 1, \ldots, \varphi_s + 1)) = 0$, we can get the right result with only one instance of the protected cipher.

It is not always practical to represent a block cipher in $GF(2)$, as for AES which representation would be too large, but some ciphers are naturally fitted to. For instance, we can compute a practical implementation of the block cipher 3-Way [6] in $GF(2)$ following our ideas. We can also remark the simplicity of its polynomial representation and it seems well-suited for our work, but its white box implementation does not resist to interpolation (see Sec. 5.2).

4 An example: white box represention of AES

In the following, we present a practical set-up of our method on the AES block cipher [11]. Its main advantage, for this work, is that it can be efficiently written as a transformation on elements in $GF(2^8)$ (cf. [13]).

4.1 Brief description of AES

AES takes as input a 16-byte block and consists mainly in the iteration of a round. The 16-byte block is represented as a 4×4 square called a *state* and subject to the following operations:

- `AddRoundKey`: this operation adds a round key to the state by a bitwise XOR operation;
- `SubBytes`: the non-linear byte substitution operates independently on each byte;
- `ShiftRows`: it is a permutation on the 16 bytes;
- `MixColumns`: this operation treats each column as a 4-byte vector and multiplies it by a matrix.

The complete algorithm consists in the sequence of operations on the left part below but for our application, we will write the algorithm, as on the right part, i.e. in a slightly different, though equivalent, way.

1. `AddRoundKey`(K_0)
2. for i from 1 to 9:
   (a) `SubBytes`;
   (b) `ShiftRows`;
   (c) `MixColumns`;
   (d) `AddRoundKey`(K_i).
3. `SubBytes`;
4. `ShiftRows`;
5. `AddRoundKey`(K_{10}).

1. for i from 1 to 9:
   (a) `AddRoundKey`(K_{i-1});
   (b) `SubBytes`;
   (c) `ShiftRows`;
   (d) `MixColumns`;
2. `AddRoundKey`(K_9).
3. `SubBytes`;
4. `ShiftRows`;
5. `AddRoundKey`(K_{10}).

Thus, the cipher consists now in the repetition of a sequence of operations named rounds 1 to 9, the four remaining operations compose the 10-th and last round.

4.2 Representation of the original rounds

If we compose the operations of one round we obtain the previously defined systems $(S_i)_{1 \leq i \leq 9}$; they are systems of $n = 16$ equations in $GF(2^8)$,

each one depending on 4 variables and having around 1020 monomials of maximal degree 254. The last round gives a system S_{10} of 16 equations, each having around 255 monomials of maximal degree 254.

4.3 Choice of parameters

Firstly, we choose a polynomial $\tilde{0}$ following Sec. 2, which equals zero at least half of the time, and we will construct $\widetilde{\Phi}$ accordingly.

Now, there are two important points to deal with: the efficiency of the representation, and the protection of the original rounds. Let $i \in \{2, \ldots, 10\}$ be a round index, let $N_{i-1} = n + s + t_{i-1}$ be the number of inputs involved in this round and let $(L_{i-1,1}, \ldots, L_{i-1,N_{i-1}})$ be the lines of M_{i-1}^{-1}. Then, we obtain

$$Y_i^1 = S_i(\begin{pmatrix} L_{i-1,1} \\ \vdots \\ L_{i-1,n} \end{pmatrix} Z_{i-1}) \text{ and } Y_i^2 = \begin{pmatrix} L_{i-1,n+1} \\ \vdots \\ L_{i-1,n+s} \end{pmatrix} Z_{i-1}.$$

The parameters should be chosen according to the following conditions:

Cond. 1 The size s of $\widetilde{\Phi}$ is chosen in order to give a sufficient number of possible values for $(\varphi_1, \ldots, \varphi_s)$, and such that it consists in enough equations to ensure that the intermediate rounds are not valid, with a high probability when evaluated individually with random inputs (i.e. the system Y_R^2 will not be cancelled in the last round). We choose here a system of $s = 4$ equations.

Cond. 2 For the n first lines of M_{i-1}^{-1}, the number of non-zero coefficients in a line should be small, otherwise the number of terms in the resulting image Y_i^1 of the block S_i would have an exponential number of monomials in variables $z_{i-1,1}, \ldots, z_{i-1,N_{i-1}}$. For instance with our example of AES white box representation, we set this number to exactly 2 variables by lines.

Cond. 3 As the value of Y_i^2 will be a constant with a probability greater than one half when the inputs come from the previous rounds, the number of non-zero inputs of lines $L_{i-1,j}$ (for $j \in \{n+1 \ldots n+s\}$), which determines the number of variables $z_{i-1,1}, \ldots, z_{i-1,N_{i-1}}$ involved in the computation of Y_i^2, should be sufficient to avoid an attacker to guess the value of the line; note that for this he could validate his guess by looking for an almost constant result (with probability $> 1/2$).

Cond. 4 We should also control the number of non-zero coefficients in M_{i-1} to bound the number of monomials in each equations at the output of the $(i-1)$-th round.

Cond. 5 At last, random systems R_{i-1} should contain a sufficient number of equations (say at least as many as the total number of equations of Y_{i-1}^1 and Y_{i-1}^2) and should look like the other systems in order to mask all the operations.

To manage all these points, we chose random systems R_i of $t_i = 23$ equations for the i-th round ($1 \leq i \leq 9$) and matrix M_i are constructed as block matrix having the following form:

$$M_i = \pi_i \circ \begin{pmatrix} A_i^{(1)} & & & \\ & \ddots & & \\ & & A_i^{(7)} & \\ & & & B_i \end{pmatrix} \circ \sigma_i,$$

where

- the $A_i^{(j)}$ are random invertible blocks of size 5×5 such that their inverses have exactly 2 non-zero coefficients on each line;
- B_i is a random invertible submatrix of size 8×8 with its inverse having at least 7 non-zero coefficients by line;
- π_i is a random permutation of $\{1, \ldots, N_i\}$;
- σ_i a random permutation of $\{1, \ldots, N_i\}$ such that

$$\sigma_i(\{1, \ldots, n\} \cup \{n + s + 1, \ldots, n + s + 19\}) = \{1, \ldots, n + 19\},$$
$$\sigma_i(\{n + 1, \ldots, n + s\} \cup \{n + s + 20, \ldots, N_i\}) = \{n + 20, \ldots, N_i\}$$

In other words, the $A_i^{(j)}$ are used to mix the 16 original equations with 19 random equations (19 is in fact the first number k greater than $n = 16$ such that $5 \mid k + n$) and B_i is used to mixed the second system Y_i^2 (of s equations) with exactly $s = 4$ other random equations.

Thus, the choice of a diagonal block matrix helps for Cond. 4. The construction of $A_i^{(j)}$ is implied by Cond. 2, the one of B_i comes from Cond. 3. For Cond. 5, the R_i will be taken with about the same number of terms as Y_i^1 and Y_i^2 gathered with the aim to look like both these systems. The parameter t_i is chosen to have enough equations to mask all other equations and with respect to the construction of M_i, i.e. to make possible the cutting in seven blocks of size 5×5 and one block of size 8×8. The size of each block has been set according to some security concerns as it will be explained in Sec. 5. As this construction is regular and limits the transformations, the matrix are preceded and followed by the two permutations π_i and σ_i in order to enlarge the space of possible transformations.

4.4 Resulting cipher

We have actually computed one instance of the resulting cipher according to the previous parameters. Table 1 gives the results of our implementation. Considering that each monomial can be coded on 17 bytes for the

Table 1. Size of the new representation of the cipher

round	s	t_i	# monomials
1	4	24	161.353
2 to 9	4	24	3.445.848
10	4	-	201.600

first round (one for the coefficient and the remainder for the exponents of the $x_1, \ldots, x_n$), on 5 bytes (one for the coefficient, two for the variables and two for the exponents, as there are in fact only binomials) in the rounds 2 to 9, and 44 bytes (one for the coefficient and the remainder for the exponents) for the last round, then 142MB are necessary to code one instance of the whole cipher.

To get the right result by a majority vote – as explained in Sect. 3.4 – four concurrent implementations are necessary. To this end, we choose four correlated polynomials $\tilde{0}$ such that, for any input, two of them equal zero while the two others will give two different values (i.e. the good result will be distinguishable).

5 Security analysis

5.1 General observations

Each round, taken individually, always returns 'false' results due to the random equations and the block Y_i^2. But, the whole algorithm often gives the right result. The rounds are in fact linked all together as the specific terms, $\widetilde{\Phi}$, are cancelled only at the end. The crucial point here is that an attack against all the rounds at the same time seems hard to conceive.

The random systems R_i are chosen such that they hide the equations of original systems S_i and additional systems, namely $\widetilde{\Phi}$ in the first round and identity in the following ones.

In M_i, the block's size are chosen such that each submatrix lie in a subspace large enough to prevent an attacker to find them, one after the other by exhaustive search. Indeed, a block of size 5×5 was the

minimum to have more than 2^{80} invertible matrices in $GF(2^8)$ with 2 non-zero coefficients on each line.

For the choice of the blocks B_i (with regard to the fourth condition page 9), it thwarts an attacker to recover $(\varphi_1, \ldots, \varphi_s)$ and the corresponding lines of M_i^{-1} as there are too many guesses to make. To recover one line for the i-th round, all the linear combination involving seven inputs of Z_i have to be checked (by looking for the probability of constancy), which means more than $\binom{43}{7}(2^8-1)^7 > 2^{80}$ possibilities.

5.2 Resistance to a specific attack: interpolation

An attacker could try to recover the original system by using interpolation and solve the system to recover the value of the key. In this configuration, where the attacker is restricted to perform the interpolation in the base field, the best algorithm [14] has the following characteristics: for an equation of k terms and n variables in $GF(q)$, the complexity is given by $q^4 n \log(k)\, k^{\log(q)}$. Note also that this algorithm assumes the attacker can choose the points, which is not the case in our problem due to the perturbations.

If we consider our example with AES, the original systems lie in $GF(2^8)$ and have the following characteristics:

- for all rounds except the last one: $k = 1020, n = 4$;
- for the last round: $k = 255, n = 4$.

The complexity is thus around $2^{32} \cdot 4 \cdot 10 \cdot 2^{10 \cdot 8}$ in the first case and $2^{32} \cdot 4 \cdot 8 \cdot 2^{8 \cdot 8}$ in the other, in each case it is more than 2^{100}.

5.3 Additional protective measures

It seems that the beginning and the end of the algorithm, where no linear mixing is introduced, are where an attacker would concentrate. Following [5], additional operations could be added at this locations. Of course, such an implementation is not anymore compliant with the specification of the original cipher.

6 Conclusion

We give a new solution to the problem of white box representation of block ciphers. Following Bringer *et al.* [3], we improve their work on the perturbation of the algebraic structure of a cipher, implemented as systems

of multivariate polynomials in a finite field. As a challenge, we describe how our techniques can be applied to the AES block cipher and give some elements to evaluate the security of our proposition against some specific attacks. Intended to run in software on a general purpose processor, our solution has still to be improved to achieve better performances. This problem – or designing a new cipher dedicated to our white box solution while staying efficient – and the scrutiny of the security are open avenues for future works.

References

1. Olivier Billet and Henri Gilbert, *A Traceable Block Cipher*, Advances in Cryptology – ASIACRYPT 2003 (C.S. Laih, Ed.), LNCS vol. 2894, Springer, 2003, pp. 331–346.
2. Olivier Billet, Henri Gilbert and Charaf Ech-Chatbi, *Cryptanalysis of a White Box AES Implementation*, Selected Areas in Cryptography (H. Handschuh and M. A. Hasan, Eds.), LNCS vol. 3357, Springer, 2004, pp. 227–240.
3. Julien Bringer, Hervé Chabanne and Emmanuelle Dottax, *Perturbing and Protecting a Traceable Block Cipher*, CMS 2006, LNCS vol. 4237, Springer, 2006, pp. 109–119.
4. S. Chow, P. Eisen, H. Johnson and P.C. van Oorschot, *A White-Box DES Implementation for DRM Applications*, Proceedings of ACM CCS-9 Workshop DRM 2002 (J. Feigenbaum Ed.), LNCS vol. 2696, Springer, 2003, pp. 1–15.
5. S. Chow, P. Eisen, H. Johnson and P.C. van Oorschot, *White-Box Cryptography and an AES Implementation*, Proccedings of SAC'02 (K. Nyberg and H. M. Heys, Eds.), LNCS vol. 2595, Springer, 2003, pp 250–270.
6. Joan Daemen, René Govaerts, and Joos Vandewalle, *A New Approach Towards Block Cipher Design*, Proceedings of FSE'93 (R. Anderson Ed.), LNCS vol. 809, Springer, 1994, pp 18–32.
7. Jintai Ding, *A New Variant of the Matsumoto-Imai Cryptosystem through Perturbation*, Public Key Cryptography (F. Bao, R. H. Deng and J. Zhou, Eds.), LNCS vol. 2947, Springer, 2004, pp. 305–318.
8. Jean-Charles Faugère and Ludovic Perret, *Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects*, Advances in Cryptology – EUROCRYPT 2006 (Serge Vaudenay, Ed.), LNCS vol. 4004, Springer, 2006, pp 30–47.
9. Matthias Jacob, Dan Boneh and Edward W. Felten, *Attacking an Obfuscated Cipher by Injecting Faults*, Proceedings of ACM CCS-9 Workshop DRM 2002 (J. Feigenbaum Ed.), LNCS vol. 2696, Springer, 2003, pp 16–31.
10. Hamilton E. Link and William D. Neumann, *Clarifying Obfuscation: Improving the Security of White-Box Encoding*, Cryptology ePrint Archive, Report 2004/025, http://eprint.iacr.org/2004/025, 2004.
11. National Institute of Standards and Technology, *FIPS-197: Advanced Encryption Standard*, 2001, available at http://csrc.nist.gov/publications/fips/.
12. Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new Families of asymmetric Algorithms*, EUROCRYPT'96, LNCS vol. 1070, Springer, 1996, pp. 33-48.
13. Joan Daemen and Vincent Rijmen, *AES Proposal: Rijndael*, Selected as the Advanced Encryption Standard, available from http://csrc.nist.gov/encryption/aes/.
14. Kai Werther, *The Complexity of Sparse Polynomial Interpolation over Finite Fields*, Appl. Algebra Eng. Commun. Comput., vol. 5, 1994, pp. 91–103.