Pioneer Journal of Algebra, Number Theory and its Applications
Volume 4, Number 1, 2012, Pages 5-34
This paper is available online at http://www.pspchv.com/content_PJANTA.html

PROPOSAL FOR A NEW EQUATION SYSTEM MODELLING OF BLOCK CIPHERS AND APPLICATION TO AES 128

MICHEL DUBOIS and ERIC FILIOL

Laboratory of Operational Virology and Cryptology
Laval, France
e-mail: michel.dubois@esiea-ouest.fr
eric.filiol@esiea.fr

Abstract

One of the major issues of cryptography is the cryptanalysis of cipher algorithms. Cryptanalysis is the study of methods for obtaining the meaning of encrypted information, without access to the secret information that is normally required. Some mechanisms for breaking codes include differential cryptanalysis, advanced statistics and brute-force.

Recent works also attempt to use algebraic tools to reduce the cryptanalysis of a block cipher algorithm to the resolution of a system of quadratic equations describing the ciphering structure. As an example, Nicolas Courtois and Josef Pieprzyk have described the AES-128 algorithm as a system of 8000 quadratic equations with 1600 variables. Unfortunately, these approaches are, currently, deadlocks because of the lack of efficient algorithms to solve large systems of equations.

In our study, we will also use algebraic tools but in a new way: by using Boolean functions and their properties. A Boolean function is a function from $F_2^n \to F_2$ with $n > 1$, characterized by its truth table. The arguments of Boolean functions are binary words of length n. Any Boolean function can be represented, uniquely, by its algebraic normal

Received September 6, 2012

2010 Mathematics Subject Classification: 11T71, 14G50, 12E12, 34A34, 94A60.

Keywords and phrases: block cipher, Boolean function, cryptanalysis, AES.


Pioneer Journal of Algebra, Number Theory and its Applications
Volume …, Number …, 2012, Pages …
This paper is available online at http://www.pspchv.com/content_PJANTA.html

PROPOSAL FOR A NEW EQUATION SYSTEM MODELLING OF BLOCK CIPHERS AND APPLICATION TO AES 128

MICHEL DUBOIS and ERIC FILIOL

Laboratory of Operational Virology and Cryptology
Laval, France
e-mail: michel.dubois@esiea-ouest.fr
eric.filiol@esiea.fr

Abstract

One of the major issues of cryptography is the cryptanalysis of cipher algorithms. Cryptanalysis is the study of methods for obtaining the meaning of encrypted information, without access to the secret information that is normally required. Some mechanisms for breaking codes include differential cryptanalysis, advanced statistics and brute-force.

Recent works also attempt to use algebraic tools to reduce the cryptanalysis of a block cipher algorithm to the resolution of a system of quadratic equations describing the ciphering structure. As an example, Nicolas Courtois and Josef Pieprzyk have described the AES-128 algorithm as a system of 8000 quadratic equations with 1600 variables. Unfortunately, these approaches are, currently, deadlocks because of the lack of efficient algorithms to solve large systems of equations.

In our study, we will also use algebraic tools but in a new way: by using Boolean functions and their properties. A Boolean function is a function from $F_2^n \to F_2$ with $n > 1$, characterized by its truth table. The arguments of Boolean functions are binary words of length n. Any

Received September 6, 2012

2010 Mathematics Subject Classification: **Kindly Provide.**

Keywords and phrases: block cipher, Boolean function, cryptanalysis, AES.

Boolean function can be represented, uniquely, by its algebraic normal form which is an equation which only contains additions modulo 2-the XOR function – and multiplications modulo 2 – the AND function.

Our aim is to describe a block cipher algorithm as a set of Boolean functions then calculate their algebraic normal forms by using the Möbius transforms. After, we use a specific representation for these equations to facilitate their analysis and particularly to try a combinatorial study. Through this approach we obtain a new kind of equations system. This equations system is more easily implementable and could open new ways to cryptanalysis.

To test our approach we first apply this principle to the mini-AES cipher and in a second time to AES-128 algorithm.

1. Introduction

The block cipher algorithms are a family of cipher algorithms which use symmetric key and work on fixed length blocks of data. As an example Rijndael[1] is a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits [6].

One of the major issues of cryptography is the cryptanalysis of cipher algorithms. Cryptanalysis is the study of methods for obtaining the meaning of encrypted information, without access to the secret information that is normally required [15]. Some mechanisms for breaking codes include differential cryptanalysis, advanced statistics and brute-force.

Recent works, like [9] or [4], attempt to use algebraic tools to reduce the cryptanalysis of a block cipher algorithm to the resolution of a system of quadratic equations describing the ciphering structure. These approaches are infeasible because of the difficulty of solving large systems of equations – for example, 8000 quadratic equations with 1600 variables for the AES-128 as described in [4].

In our study, we will also use algebraic tools but in a new way by using Boolean functions and their properties. Our aim is to describe a block cipher algorithm as a set

[1] The Rijndael cipher algorithm has become a standard cryptographic algorithm in 2000 under the name of Advanced Encryption Standard (AES).

of Boolean functions then calculate their algebraic normal forms by using the Möbius transforms. To test our approach we apply this principle to the mini-AES cipher.

Since November 26, 2001, the block cipher algorithm "Rijndael", in its 128 bits version, became the successor of DES under the name of Advanced Encryption Standard (AES). Its designers, Joan Daemen and Vincent Rijmen used algebraic tools to give to their algorithm an unequaled level of assurance against the standard statistical techniques of cryptanalysis.

Recent works suggest that what is supposed to be the AES strength could be is weakness. Indeed, according to these studies, to cryptanalyse the AES could be reduced to solving a system of quadratic equations describing the ciphering structure of the AES. These results are not implementable in real life and do not represent a true danger for the AES.

In our study, we will look at a reduced version of the AES: the Mini-AES. Our goal is to describe it under the form of systems of Boolean functions and to calculate their algebraic normal forms by using the Möbius transforms. The system of equations obtained is more easily implementable and once extended to the AES could open new ways to cryptanalysis of the AES.

2. Feasibility of the Resolution of Large Systems of Quadratic Equations Describing the AES Algorithm

2.1. AES as a system of equations

"Breaking a good cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type." [14].

This famous sentence of Claude Elwood Shannon is in the heart of what we describe now. Indeed, describe algebraically the AES allow us to define an environment in which it is possible to describe a quadratic equation system in multiple variables.

In [2], Cid et al. define two reduced variants of the AES denoted $SR(n, r, c, e)$ and $SR^*(n, r, c, e)$. Those two variants have the same parameters that are: n the number of round for encryption, r and c define respectively the number of rows and columns of the entries array and e the size of a word in bits. SR and SR^* differ by the

form of the last round. Those two environments works on blocks of size $r.c.e$ and the state array is an array $r * c$ containing words of e bits.

Finally, the complete version of the AES 128 is described by

$$SR^{*}(10, 4, 4, 8).$$

2.1.1. The big encryption system

The existence of a quadratic equations system in multiple variables has been shown [9] by defining the big encryption system (BES). BES is a system of block encryption using blocks of 128 bytes and a key of 16 bytes. AES and BES both use a state array containing bytes, array that is transformed during the different rounds steps.

The spaces of states of the AES and the BES are respectively the vector spaces $A = GF(2^8)^{16}$ and $B = GF(2^8)^{128}$. All the clear texts, the cipher texts and the possible keys are elements of A for the AES and of B for the BES. Thus, AES is a subset of BES and the mapping function defining image of AES in BES is given by

$$B_A = \phi(A) \subset B, \tag{1}$$

where $\phi(\,) : GF(2^8) \mapsto GF(2^8)^8$ is the conjugate vector of the map between A and B defined by

$$\phi(a) = (a^{2^0}, a^{2^1}, a^{2^2}, a^{2^3}, a^{2^4}, a^{2^5}, a^{2^6}, a^{2^7}).$$

Thus, each element $a \in GF(2^8)$ can be included as an element

$$(a^{2^0}, a^{2^1}, a^{2^2}, a^{2^3}, a^{2^4}, a^{2^5}, a^{2^6}, a^{2^7}) \in GF(2^8)^8$$

with $\phi(\,)$.

2.1.2. Equation system for the encryption

The state space of the BES is $B = GF(2^8)^{128}$ and the equation of an encryption round in BES is given by

$$b \mapsto M_B(b^{(-1)}) + k_{B_i}, \tag{2}$$

where M_B is the linear diffusion matrix defined in [3] and k_{B_i} is the BES key for the round i.

The encryption of the clear text $p \in B$ to the cipher text $c \in B$ with the BES is described by

$$w_0 \in p + k_0, \tag{3a}$$

$$x_i = w_i^{(-1)}, \quad [i = 0, ..., 9], \tag{3b}$$

$$w_i = M_B x_{i-1} + k_i, \quad [i = 0, ..., 9], \tag{3c}$$

$$c = M_B^* x_9 + k_{10}, \tag{3d}$$

where $w \in B$ and $x \in B$ are the state vectors respectively before and after the inversion operation and M_B^* is the modified version of the matrix M_B for the final round.

2.2. Implementation of *SR*

2.2.1. The SageMath software

To study the equations systems previously described we will use the software Sage[2]. Sage is the acronym for *Software for Algebra and Geometry Experimentation*, it is an open source software based on the python language. It aims to provide a viable alternative to Magma, Maple, Mathematica and Matlab.

Sage comes by default with some modules including tools to work with algebraic constructs. Thus, we can create a univariate polynomial ring over the ring $\mathbb{Z}$:

```
sage: Z.<x> = ZZ[]
sage: Z
Univariate Polynomial Ring in x over Integer Ring
sage:
```

From this definition several operations are possible:

[2] http://www.sagemath.org/index.html

```
sage: Z.random_element()
-x^2 - x + 3
sage: Z.characteristic()
0
sage: Z.is_finite()
False
```

Well, it is possible to work with polynomial rings over $GF(2)$:

```
sage: A.<x> = GF(2)[]
sage: A
Univariate Polynomial Ring in x over Finite Field of size 2
sage: (x^2 + 1).is_irreducible()
False
sage: (x^3 + x + 1).is_irreducible()
True
sage: x.degree()
1
sage: x.list()
[0, 1]
sage:
```

2.2.2. Study of a lighter version

In 2007, Martin Albrecht[3] has begun to develop a module for the Sage software implementing the *SR* environment detailed above.

We start by working with $SR(1, 1, 1, 4)$ in other words with an AES encryption on 1 round with a state array of 1 column and 1 row and a word of 4 bits.

Construction of $SR(1, 1, 1, 4)$ and display the result:

```
sage: sr = mq.SR(1, 1, 1, 4)
sage: sr
SR(1,1,1,4)
sage:
```

Display of the variables:

[3]http://www.informatik.uni-bremen.de/~malb/ and

http://www.informatik.uni-bremen.de/cgi-bin/cgiwrap/malb/blosxom.pl/

```
sage: print sr.R.repr_long()
Polynomial Ring
  Base Ring : Finite Field in a of size 2^4
       Size : 20 Variables
   Block  0 : Ordering : degrevlex
              Names    : k100, k101, k102, k103, x100, ...
sage:
```

Calculate and display the irreducible polynomial:

```
sage: sr.base_ring().polynomial()
a^4 + a + 1
sage:
```

Displaying the S-BOX:

```
sage: sr.sbox()
(6, 11, 5, 4, 2, 14, 7, 10, 9, 13, 15, 12, 3, 1, 0, 8)
sage:
```

Calculate of the polynomial equation system:

```
sage: sr.polynomial_system()
(Polynomial System with 40 Polynomials in 20 Variables,
 {k003: a, k002: a^2 + 1, k001: a + 1, k000: a^2})
sage:
```

2.2.3. Some data for the AES

As we have seen previously, in the environment *SR*, the AES-128 is described by: $SR^*(10, 4, 4, 8)$. By using the *SR* module of Sage, we obtain the following data:

Construction of $SR^*(10, 4, 4, 8)$ and display of the result:

```
sage: sr = mq.SR(10, 4, 4, 8, star=True)
sage: sr
SR*(10,4,4,8)
sage: sr.base_ring()
Finite Field in a of size 2^8
sage:
```

Calculate and display of the corresponding polynomial system:

```
sage: sr.polynomial_system()

(Polynomial System with 8576 Polynomials in 4288 Variables,
 {k001507: a^7 + a^6 + a,
  k001506: a^6 + a^5 + a + 1,
  k001505: a^5 + a^4 + a^2 + a,
  k001504: a^5 + a^4 + a^3 + 1,
  k001503: a^5 + a^3 + a^2 + a + 1,
  k001502: a^7 + a^6 + a^4 + a + 1,
  k001501: a^6 + a^5 + a^2 + a,
  k001500: a^5 + a^4 + a^2 + 1,
...
sage:
```

It is possible to go further by developing its own python scripts using the Sage libraries. For this, the shebang of the script and the imported modules are the followings:

```
#!/usr/bin/env sage -python
# -*- coding: utf-8 -*-

from sage.all import *
from files_aes import *
```

Then, the following function realize some encryption tests from the vectors given in [6].

```
1  def aes_128(round):
2          """ AES: FIPS 197 implementation """
3          aes = mq.SR(round, 4, 4, 8, star=True, gf2=True,
4                  allow_zero_inversions=True)
5          print aes
6          print aes.base_ring()
7          print aes.base_ring().base()
8          print aes.base_ring().degree()
9          print aes.base_ring().cardinality()
10         print aes.base_ring().polynomial()
11         print aes.base_ring().ideal()
12         print aes.base_ring().zero_element()
13         print aes.sbox_constant()
14         print aes.block_order()
15         print
16         print '#### key schedule ####'
17         key = '2b7e151628aed2a6abf7158809cf4f3c'
18         print 'key:', key
19         temp = [aes.base_ring().fetch_int(ZZ(key[i:i+2], 16))\
20                 for i in range(0, len(key), 2)]
21         key = aes.state_array(temp)
22         print key
23         print aes.hex_str(key)
24         for r in range(aes.n):
25                 key = aes.key_schedule(key, r+1)
26                 print aes.hex_str(key)
27         print
28         print '#### first test of AES cipher ####'
29         plain = '3243f6a8885a308d313198a2e0370734'
30         key = '2b7e151628aed2a6abf7158809cf4f3c'
31         set_verbose(2)
32         cipher = aes(plain, key)
33         set_verbose(0)
34         print
35         print '#### second test of AES cipher ####'
36         plain = aes.vector([0, 0, 1, 1,...,1, 0, 0]) # fips 197 vector
37         key = aes.vector([0, 0, 1, 0,..., 0, 0]) # fips 197 vector
38         set_verbose(2)
39         cipher = aes(plain, key)
40         set_verbose(0)
```

The above function aes_128 takes as parameters the number of rounds to perform. To execute a full AES encryption, we have to realize 10 rounds.

The lines 4 to 13 print some informations on the field on which AES is defined. The lines 15 to 25 print the rounds key. The lines 26 to 32 realize the encryption of a vector of a clear text with a key, both given in [6]. Finally, the lines 33 to 38 realize the encryption of the same clear text vector with the same key but under the form of a 128 bits string.

2.3. Deciphering the AES with the help of SR module

As we have seen in our introduction, deciphering the AES by resolving a large system of quadratics equation is a deadlock. Unfortunately there is currently no efficient algorithm to resolve a such large system of equations.

Even using a Groebner basis as proposed by Cid, Murphy and Robshaw the computation time is too large.

As an example, we used the SR module, from sage, to decipher some simplified versions of the AES algorithm. To do that we used the following code:

```
1  #!/usr/bin/env sage -python
2  # -*- coding: utf-8 -*-
3
4  import time
5  from sage.all import *
6
7  def recoverKey():
8          t0 = time.time()
9          aes = mq.SR(1, 1, 1, 4, gf2=True, star=True,\
10                 allow_zero_inversions=True)
11         plain = aes.vector([0, 1, 1, 0])
12         print "plain:", plain._list()
13         key = aes.vector([1, 1, 0, 0])
14         print "key:", key._list()
15         set_verbose(2)
16         cipher = aes(plain, key)
17         set_verbose(0)
18         print "cipher:", cipher._list()
19         F, s = aes.polynomial_system(P=plain, C=cipher)
20         print "number of solutions:", len(F.ideal().variety())
21         print F.groebner_basis()
22         for V in F.ideal().variety():
23                 tmp = []
24                 for key, value in sorted(V.items()):
25                         if str(key)[0:2] == "k0":
26                                 tmp.append(int(value))
27                 result = []
28                 for i in range(len(tmp)):
29                         result.append(tmp[len(tmp)-1-i])
30                 print result
31         print time.time() - t0
32
33 if __name__ == "__main__":
34         recoverKey()
```

The line 8 defines the time $t0$ to calculate the duration of the computation. The line 9 defines a simplified AES algorithm using the SR module. In our case we define

AES in $GF(2^4)$ with one round and a state array of one column and one row. The lines 11 to 14 define a plain text and a key. The lines 15 to 17 realize the ciphering of the plain text. The lines 19 to 30 calculate the system of equation and compute a solution through a Groebner basis. Finally the lines 31 print the duration of the calculation.

By launching our script we obtain the following results:

```
$ ./sage_aes.py
plain: [0, 1, 1, 0]
key: [1, 1, 0, 0]
R[01].s_box   F
R[01].s_row   F
R[01].k_sch   2
R[01].output  D
cipher: [1, 1, 0, 1]
number of solutions: 2
Polynomial Sequence with 20 Polynomials in 20 Variables
[1, 1, 0, 0]
[1, 0, 1, 0]
1.0291390419
```

Thus, for an AES algorithm reduced to one round and a state array of one column and one row in $GF(2^4)$ we obtain a system with 20 Polynomials and 20 variables with two solutions. One of these solutions is the key used to cipher the plain text. The total time of computation is 1:0291390419 seconds.

In the following array we give some results obtained by changing some parameters[4]:

Round	Column	Row	$GF(2^n)$	Polynomials	Variables	Duration
1	1	1	$n = 4$	20	20	1.02s
2	1	1	$n = 4$	36	36	4.14s
3	1	1	$n = 4$	52	52	21.15s
4	1	1	$n = 4$	68	68	837.50s

[4]All calculations were performed on a macbook pro with a processor Intel Core 2 Duo cadenced at 2.53 GHZ and 4GB of RAM.

1	1	1	$n = 8$	40	40	7.13s
2	1	1	$n = 8$	72	72	200313.77s

We can be seen that the calculation time is correlated to the number of Polynomials and Variables. This can be highlighted by the graph of Figure 1 page 10.

This graph demonstrate the large amount of time required to compute a system of equations with a great number of Polynomials and Variables. With 72 Polynomials ans 72 Variables we need more than 200313 seconds – whether more than 55 hours – to solve the system of equations. We can easily imagine than, even on a more powerful computer, the calculus time needed to solve a system of equations describing the AES-128 algorithm is huge. So we need another approach that we propose now.

3. Boolean Function

3.1. Definition. A Boolean function is a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, with $n > 1$, characterized by its truth table. The arguments of Boolean functions are binary words of length n. Thus, if we take $n = 2$ we can define the Boolean function OR – the logical OR – characterized by its truth table. (cf. Figure 2 page 10).

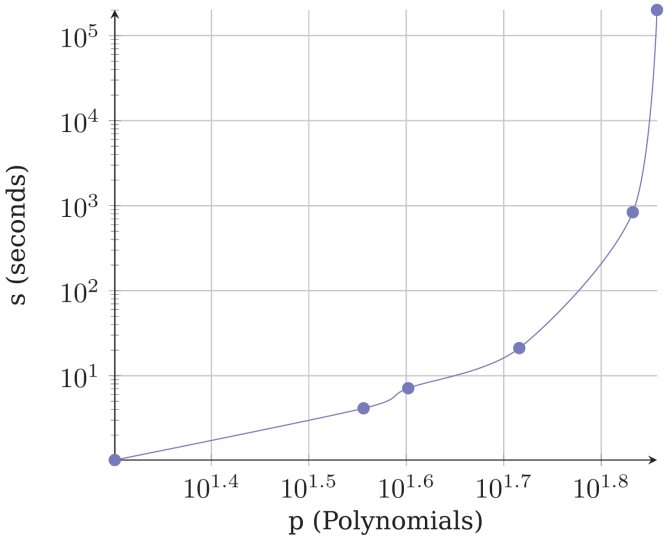


Figure 1. Evolution of computation time based on the number of polynomials.

x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1

Figure 2. The truth table of the function OR.

The support of a Boolean function $supp(f)$ is the set of elements of x such that $f(x) \neq 0$, the weight of a Boolean function $wt(f)$ is the cardinal of its support. Thus, the support of the OR function is $supp(\text{OR}) = \{(0, 1), (1, 0), (1, 1)\}$ and its weight is $wt(\text{OR}) = 3$.

3.2. Algebraic normal form

The algebraic normal form (ANF) of a Boolean function f in n variables is the only polynomial

$$Q_f : \mathbb{F}_2[x_1, ..., x_n]/(x_1^2 - x_1, ..., x_n^2 - x_n)$$

such as $\forall (x_1, ..., x_n) \in \mathbb{F}_2^n$, we have

$$f(x_1, ..., x_n) = Q_f(x_1, ..., x_n)$$

$$= \sum_{(u_1, ..., u_n) \in \mathbb{F}^n} a_u \prod_{i=1}^{n} x_i^{u_i}.$$

We call degree of f, denoted $\deg(f)$, the degree of the polynomial Q_f which corresponds to the highest degree of monomials with nonzero coefficients from the ANF of f. Furthermore, the ANF of a Boolean function exists and is single.

In short, any Boolean function can be represented, uniquely, by its algebraic normal form under the form

$$f(x_1, ..., x_n) = a_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n$$

$$+ a_{1,2}x_1x_2 + \cdots + a_{n-1,n}x_{n-1}x_n + \cdots + a_{1,2,\ldots,n}x_1x_2 \cdots x_n$$

Thus, the algebraic normal form of a Boolean function only contains additions modulo 2 – the XOR function – and multiplications modulo 2 – the AND function.

By using the example of the function OR, we have

$$\begin{aligned} f(x_1, x_2) &= x_1 \text{ OR } x_2 \\ &= f(0, x_2) + f(x_1, 0) + f(x_1, x_2) \\ &= x_1 + x_2 + x_1x_2. \end{aligned}$$

3.3. Möbius transform

It is with the help of the Möbius transform that we will calculate the ANF of a Boolean function.

The Möbius transform of the Boolean function f is defined by [8]:

$$\begin{aligned} MT(f) : \mathbb{F}_2^n &\rightarrow \mathbb{F}_2 \\ u &= \sum_{v \leq u} f(v) \bmod 2 \end{aligned}$$

with $v \leq u$ if and only if $\forall i,\ v_i = 1 \Rightarrow u_i = 1$.

From there, we can define the algebraic normal form of a Boolean function f in n variables by

$$\sum_{u=(u_1, \ldots, u_n) \in \mathbb{F}_2^n} MT(u)\, x_1^{u_1} \cdots x_n^{u_n}.$$

Table 1. The truth table of the function MajAmong3

x_1	x_2	x_3	MajAmong3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1

1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 2. Calculation of the Möbius transform for MajAmong3

x_1	x_2	x_3	MajAmong3	→	calculation of MT(f)				MT(f)
0	0	0	0	→	0	0	0	0	0
0	0	1	0	→	0	0	0	0	0
0	1	0	0	→	0	0	0	0	0
0	1	1	1	→	1	1	1	1	1
1	0	0	0	→	0	0	0	0	0
1	0	1	1	→	1	1	1	1	1
1	1	0	1	→	1	1	1	1	1
1	1	1	1	→	1	0	1	0	0

4. Workings of the Equations

The goal of our study is to propose a model of the mini-AES as a set of equations obtained by using the Boolean functions. This modelling is a first step, the final goal is to use this method on the AES.

4.1. Principle of elaboration

The principle adopted for the system of equations consists, from the truth table of a Boolean function, to calculate its ANF by using its Möbius transform.

4.2. Example with the MajAmong3 function

To understand more easily the mechanisms implemented, consider an example with the function MajAmong3. This function: $\mathbb{F}_2^3 \to \mathbb{F}_2$, is characterized by the truth table presented in Table 1 pages 14-15.

By calculating the Möbius transform of the function we obtain the result of Table 2 page 15.

Once the Möbius transform obtained we take the elements of $\mathbb{F}_2^3$, where $MT(\text{MajAmong3}) \neq 0$. In our case we have the triplets $(0, 1, 1)$, $(1, 0, 1)$, $(1, 1, 0)$ from which we can deduce the equation:

$$\text{MajAmong3}(x_1, x_2, x_3) = x_2x_3 \oplus x_1x_2.$$

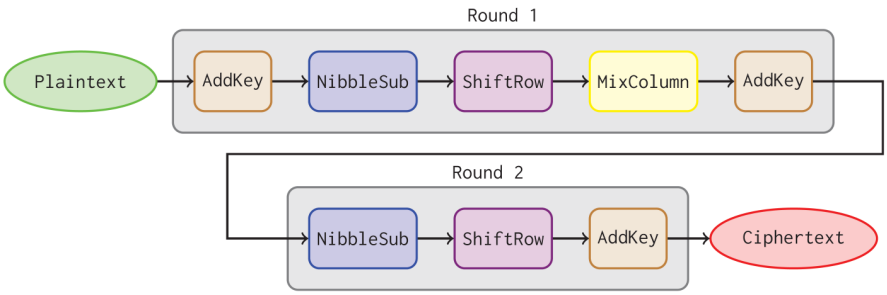


Figure 3. Architecture of the mini-AES rounds.

With the addition matching to a XOR and the multiplication to a AND.

5. Application to the Mini-AES

5.1. The mini-AES

With the goal of helping students in cryptography and the cryptanalysts better to understand the internal mechanisms of the AES that Raphael Chung-Wei Phan presented, in 2002, his mini version of the AES [10]. This version uses restrained parameters compared to the AES while preserving its internal structure.

The mini-AES is a block ciphering algorithm based on the same mathematical primitives than its big brother the AES. The atomic elements with which the mini-AES works are elements of the finite field $GF(2^4)$ called nibbles. As with AES, mini-AES uses a state array containing four nibbles, making it a 16 bits block ciphering algorithm.

The ciphering process of the mini-AES consists on two rounds involving the

functions NibbleSub applying the S-BOX[5] to the state array, ShiftRow performing a rotation of the cells of the state array and MixColumn multiplying each column of the state array by a constant matrix.

The rounds architecture is presented on Figure 3.

5.2. Equations for the mini-AES

By taking the principle of generating equations showed in the paragraph 4, we will define the equations for the mini-AES.

To simplify the process, we reduce the mini-AES to five Boolean functions: $\mathbb{F}_2^{16} \rightarrow \mathbb{F}_2^{16}$, one function for each round and three for the derivation key process.

The functions for the rounds X_1 and X_2 result from the conjunction of functions NibbleSub $NS(\,)$, ShiftRow $SR(\,)$ and MixColumn $MC(\,)$ such as, by taking a 16 bits plain text block $b = (b_1, ..., b_{16})$, we have $X_1(b) = MC \circ SR \circ NS(b)$ and $X_2(b) = SR \circ NS(b)$.

The three functions K_0, K_1 and K_2 describe the derivation key process such that, from a 16 bits key block $k = (k_1, ..., k_{16})$, we have the keys $K_{i_{i\in(0,1,2)}}(k) = (k_{i,1}, ..., k_{i,16})$ used in the rounds.

Ultimately, the mini-AES can be written as two equations R_1 and R_2 each one describing a round such that:

$$\begin{aligned} b' = R_1(b) &= K_0(k) \oplus X_1(b) \oplus K_1(k) \\ &= (k_{1,1}, ..., k_{1,16}) \oplus (x_{1,1}, ..., x_{1,16}) \oplus (k_{2,1}, ..., k_{2,16}) \\ &= (b'_1, ..., b'_{16}) \\ b'' = R_2(b') &= X_2(b') \oplus K_2(k) \end{aligned}$$

[5]A Substitution-box (S-BOX) is a basic component of symmetric key algorithms which performs substitution. In cryptography, substitution is used to obscure the relationship between the key and the cipher text.

$$= (x_{2,1}, ..., x_{2,16}) \oplus (k_{3,1}, ..., k_{3,16})$$

$$= (b''_1, ..., b''_{16}).$$

With b, b' and b'' respectively the block of 16 bits in input, at the end of the first round and at the end of the second round and k the block of 16 bits of the key.

We can then calculate the truth table of the Boolean functions K_0, K_1, K_2, X_1 and X_2 (see Figure 4). Then by using the methodology chosen for the function MajAmong3, we obtain a set of 16 equations for each Boolean functions. One equation for each bit of the block.

The resulting equations for the X_1 function are presented in Appendix A.

5.3. Formatting the equations

To facilitate the analysis of this set of equations and particularly to try later a combinatorial study we will use a specific representation for these equations.

The adopted principle consists of generating a file for each of the 16 equations of the R_1 and R_2 functions. Ultimately we will obtain 32 files. The contents of each of these files consists of lines containing sequences of 0 and 1. Each line describes a monomial of the equation and the transition from one line to another means the application of the XOR operation (see Figure 5).

In order to facilitate understanding of the mechanism chosen we detail the implementation of the file corresponding to the bit b'_1 of the function R_1 in Figure 6.

$$
\begin{aligned}
X_1(1026) &= 1000101101011001 \\
X_1(1027) &= 0011110001011001 \\
X_1(1028) &= 0101100101011001 \\
X_1(1029) &= 1100110101011001 \\
&\dots \\
X_1(32000) &= 0100001000000111 \\
X_1(32001) &= 0011111100000111 \\
X_1(32002) &= 0010011100000111 \\
X_1(32003) &= 1001000000000111 \\
&\dots \\
X_1(65000) &= 1111101100011000 \\
X_1(65001) &= 1110001100011000 \\
X_1(65002) &= 0101010000011000 \\
X_1(65003) &= 0010100100011000
\end{aligned}
$$

Figure 4. Some extracts of the truth table of the $X_1()$ function.

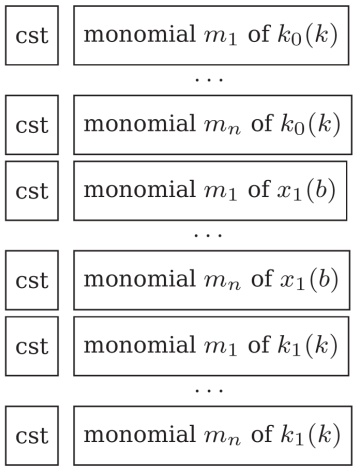


Figure 5. File structure for the function $R_1(b)$.

$$b_1' =$$
$$k_1 \oplus 1 \oplus x_{15}x_{16} \oplus x_{14} \oplus x_{14}x_{16} \oplus x_{13} \oplus x_{13}x_{15} \oplus x_{13}x_{15}x_{16} \oplus x_4 \oplus x_3x_4 \oplus x_2x_4 \oplus x_2x_3 \oplus x_2x_3x_4 \oplus$$
$$x_1x_3 \oplus x_1x_3x_4 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus 1 \oplus k_{16} \oplus k_{14} \oplus k_{14}k_{15} \oplus k_{14}k_{15}k_{16} \oplus k_{13} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{15} \oplus k_1$$

k_1	0	1000000000000000
1	1	0000000000000000
$x_{15}x_{16}$	0	0000000000000011
x_{14}	0	0000000000000100
$x_{14}x_{16}$	0	0000000000000101
x_{13}	0	0000000000001000
$x_{13}x_{15}$	0	0000000000001010
$x_{13}x_{15}x_{16}$	0	0000000000001011
x_4	0	0001000000000000
x_3x_4	0	0011000000000000
x_2x_4	0	0101000000000000
x_2x_3	0	0110000000000000
$x_2x_3x_4$	0	0111000000000000
x_1x_3	0	1010000000000000
$x_1x_3x_4$	0	1011000000000000
x_1x_2	0	1100000000000000
$x_1x_2x_3$	0	1110000000000000
1	1	0000000000000000
k_{16}	0	0000000000000001
k_{14}	0	0000000000000100
$k_{14}k_{15}$	0	0000000000000110
$k_{14}k_{15}k_{16}$	0	0000000000000111
k_{13}	0	0000000000001000
$k_{13}k_{14}$	0	0000000000001100
$k_{13}k_{14}k_{15}$	0	0000000000001110
k_1	0	1000000000000000

Figure 6. File corresponding to the bit b_1'.

6. Application to the AES

The AES is an algorithm of symmetric block cipher. It encrypts and decrypts data blocks from a single key. Contrary to the DES, which is based on a Feistel network, the AES uses a network of substitution and permutation (SP-network). This includes substitution boxes, the S-Boxes, and permutation boxes, the P-Boxes. Each box takes a block of text and the key as input and provides a block of ciphered text as output. The information flow in a defined sequence of several P-Box and S-Box forms a round.

This mechanism implements the principles of diffusion and confusion developed by Shannon [14]. The objective of diffusion is to dissipate the statistical redundancy

of the plain text in the ciphered text. Permutation operations ensure the diffusion. The objective of the confusion is to make difficult the relationship between the plain text, the key and the ciphered text. The confusion is obtained by substitutions, chosen with care.

Historically, the AES has two predecessors. The first is the cipher block algorithm Shark [11] published in 1996 by Vincent Rijmen, Joan Daemen, Bart Preneel, Anton Bosselaers and Erik de Win. Shark uses blocks of 64 bits and a key of 128 bits. Like AES, it uses a SP-network with six rounds. The second algorithm called Square was published in 1997 by Joan Daemen and Vincent Rijmen. It uses an SP-network with eight rounds and works on blocks of 128 bits and also key of 128 bits.

6.1. The AES algorithm

The inputs and outputs of AES are 128 bits blocks and the key length can be 128, 192 or 256 bits. The basic unit of the algorithm is the byte. Blocks of data provided as input are transformed into arrays of four columns and four rows, each box containing a byte, whether $4 * 4 * 8 = 128$ bits per arrays.

At the beginning of operations of ciphering and deciphering of a block, the corresponding array of bytes is copied into the state array. The state array is a two-dimensional array of bytes containing n rows and m columns. For AES, $n = m = 4$. The operations of encryption and decryption are performed on this array and then the result is copied into an output array.

The ciphering operations (see fig. 7) are based on four predefined functions: AddRoundKey, SubBytes, ShiftRows and MixColumns. Each of these functions is executed on the state array. The ciphering cycle includes an initial transformation, some intermediate rounds and a final round.

The initial transformation consists of applying the function AddRoundKey to the state array. The intermediate rounds execute, in the order, the functions SubBytes, ShiftRows, MixColumns and AddRoundKey on the state array. The final round differs from the intermediate rounds, by the removal of the function MixColumns in the transformations cycle.

The number of rounds of AES depends of the key size. Thus, for a key of 128 bits, the number of rounds is 10, likewise, we have 12 rounds for a key of 192 bits

and 14 rounds for a 256 bits key.

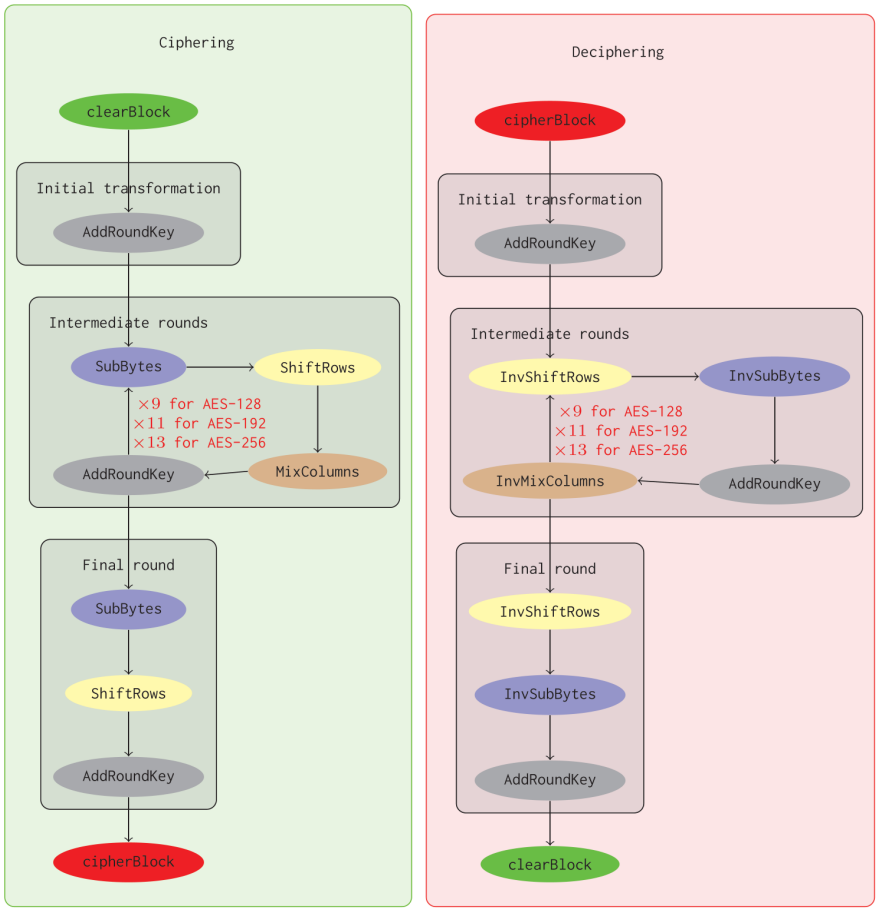


Figure 7. Ciphering and deciphering processes of AES

The deciphering (see fig. 7) is realized by performing the inverse operations of the four encryption functions, in the inverse order.

Thus, each function used in the ciphering operations disposes of its inverse function used for the deciphering: InvShiftRows, InvSubBytes and Inv-MixColumns. The AddRoundKey function stays unchanged. Like for ciphering, the deciphering process includes an initial transformation, some intermediate rounds and a final round.

The initial transformation consists of applying the AddRoundKey function at the

state array. The intermediate rounds execute, in order, the InvShift-Rows, InvSubBytes, AddRoundKey and InvMixColumns functions on the state array. The final round differs of the intermediate rounds by the suppression of the InvMixColumns function in the transformations sequence.

6.2. Equations for the AES

Our aim is to apply to the AES the mechanisms described above for the miniaes. However, with the mini-aes, we have Boolean functions from $\mathbb{F}_2^{16} \rightarrow \mathbb{F}_2^{16}$ and it is relatively easy to compute their truth tables. In the AES algorithm, the ciphering functions take 128 bits as input and 128 bits as output. So we should have Boolean functions from $\mathbb{F}_2^{128} \rightarrow \mathbb{F}_2^{128}$ and it is impossible to compute their truth table. Indeed, such truth table contains 2^{128} inputs.

We have to find solutions to describe each functions of the AES algorithm to obtain the same result as for the mini-aes algorithm.

At the end, like for mini-aes, we obtain 128 files, each one describing the transformations of a bit block. The content of each of these files consists of lines containing sequences of 0 and 1. Each line describes a monomial of the equations and the transition from one line to another means the application of the XOR operation.

6.2.1. Solution for the SubBytes function

The SubBytes transformation is a non-linear byte substitution that operates independently on each byte of the State array using a substitution table (S-Box).

This function is applied independently on each byte of the input block. So, we can reduce it as a Boolean function $\mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8$ describing the S-Box of the AES. Thus we compute the truth table of the S-Box and then apply the Möbius transform on the result. By switching these results on the 16 bytes of the input block, we obtain 128 equations each one describing one bit of the input block.

As an example the equation for low level bit is given on the figure 8.

6.2.2. Solution for the ShiftRows function

In the ShiftRows transformation, the bytes in the last three rows of the State

array are cyclically shifted over different numbers of bytes (offsets). The first row, $r = 0$, is not shifted.

$$1 \oplus x_{127} \ominus x_{126}x_{127} \oplus x_{125} \oplus x_{125}x_{126} \oplus x_{124} \ominus x_{124}x_{126} \oplus x_{124}x_{125} \oplus x_{124}x_{125}x_{126} \oplus$$
$$x_{124}x_{125}x_{126}x_{127} \oplus x_{123} \oplus x_{123}x_{127} \oplus x_{123}x_{126} \oplus x_{123}x_{126}x_{127} \oplus x_{123}x_{125} \oplus x_{123}x_{125}x_{127} \oplus$$
$$x_{123}x_{125}x_{126} \oplus x_{123}x_{124}x_{127} \oplus x_{123}x_{124}x_{126} \oplus x_{123}x_{124}x_{125}x_{126}x_{127} \oplus x_{122}x_{127} \oplus$$
$$x_{122}x_{125}x_{127} \oplus x_{122}x_{125}x_{126}x_{127} \oplus x_{122}x_{124}x_{127} \oplus x_{122}x_{124}x_{125} \oplus x_{122}x_{124}x_{125}x_{127} \oplus$$
$$x_{122}x_{124}x_{125}x_{126} \oplus x_{122}x_{123}x_{126}x_{127} \oplus x_{122}x_{123}x_{125}x_{127} \oplus x_{122}x_{123}x_{125}x_{126}x_{127} \oplus$$
$$x_{122}x_{123}x_{124}x_{125}x_{127} \bigcirc x_{121}x_{127} \bigcirc x_{121}x_{126} \bigcirc x_{121}x_{126}x_{127} \bigcirc x_{121}x_{125} \bigcirc x_{121}x_{125}x_{127} \bigcirc$$
$$x_{121}x_{125}x_{126} \oplus x_{121}x_{125}x_{126}x_{127} \oplus x_{121}x_{124}x_{127} \oplus x_{121}x_{124}x_{125}x_{126} \oplus$$
$$x_{121}x_{124}x_{125}x_{126}x_{127} \oplus x_{121}x_{123} \oplus x_{121}x_{123}x_{127} \oplus x_{121}x_{123}x_{126} \oplus x_{121}x_{123}x_{125}x_{126} \oplus$$
$$x_{121}x_{123}x_{124}x_{127} \oplus x_{121}x_{123}x_{124}x_{126} \oplus x_{121}x_{123}x_{124}x_{126}x_{127} \oplus x_{121}x_{123}x_{124}x_{125}x_{127} \oplus$$
$$x_{121}x_{123}x_{124}x_{125}x_{126}x_{127} \oplus x_{121}x_{122} \oplus x_{121}x_{122}x_{126} \oplus x_{121}x_{122}x_{125} \oplus x_{121}x_{122}x_{125}x_{127} \oplus$$
$$x_{121}x_{122}x_{124}x_{127} \oplus x_{121}x_{122}x_{124}x_{126}x_{127} \oplus x_{121}x_{122}x_{124}x_{125}x_{126} \oplus x_{121}x_{122}x_{123} \oplus$$
$$x_{121}x_{122}x_{123}x_{127} \oplus x_{121}x_{122}x_{123}x_{126} \oplus x_{121}x_{122}x_{123}x_{126}x_{127} \oplus x_{121}x_{122}x_{123}x_{125} \oplus$$
$$x_{121}x_{122}x_{123}x_{125}x_{126} \ominus x_{121}x_{122}x_{123}x_{124}x_{127} \oplus x_{121}x_{122}x_{123}x_{124}x_{125} \ominus$$
$$x_{121}x_{122}x_{123}x_{124}x_{125}x_{127} \ominus x_{120}x_{126}x_{127} \ominus x_{120}x_{125} \ominus x_{120}x_{125}x_{127} \ominus x_{120}x_{125}x_{126}x_{127} \ominus$$
$$x_{120}x_{124}x_{126} \oplus x_{120}x_{124}x_{125} \oplus x_{120}x_{124}x_{125}x_{127} \oplus x_{120}x_{124}x_{125}x_{126} \oplus x_{120}x_{124}x_{125}x_{126}x_{127} \oplus$$
$$x_{120}x_{123}x_{127} \oplus x_{120}x_{123}x_{126}x_{127} \ominus x_{120}x_{123}x_{125} \oplus x_{120}x_{123}x_{125}x_{127} \ominus x_{120}x_{123}x_{125}x_{126} \ominus$$
$$x_{120}x_{123}x_{125}x_{126}x_{127} \ominus x_{120}x_{123}x_{124} \ominus x_{120}x_{123}x_{124}x_{125}x_{126}x_{127} \ominus x_{120}x_{122} \ominus$$
$$x_{120}x_{122}x_{125} \oplus x_{120}x_{122}x_{125}x_{127} \oplus x_{120}x_{122}x_{125}x_{126}x_{127} \oplus x_{120}x_{122}x_{124} \oplus$$
$$x_{120}x_{122}x_{124}x_{126}x_{127} \oplus x_{120}x_{122}x_{124}x_{125} \oplus x_{120}x_{122}x_{124}x_{125}x_{126} \oplus$$
$$x_{120}x_{122}x_{124}x_{125}x_{126}x_{127} \oplus x_{120}x_{122}x_{123}x_{127} \oplus x_{120}x_{122}x_{123}x_{126} \oplus x_{120}x_{122}x_{123}x_{125} \oplus$$
$$x_{120}x_{122}x_{123}x_{125}x_{127} \oplus x_{120}x_{122}x_{123}x_{125}x_{126}x_{127} \oplus x_{120}x_{122}x_{123}x_{124}x_{127} \oplus$$
$$x_{120}x_{122}x_{123}x_{124}x_{126} \oplus x_{120}x_{122}x_{123}x_{124}x_{125} \oplus x_{120}x_{122}x_{123}x_{124}x_{125}x_{127} \oplus x_{120}x_{121} \oplus$$
$$x_{120}x_{121}x_{125} \oplus x_{120}x_{121}x_{125}x_{126} \ominus x_{120}x_{121}x_{125}x_{126}x_{127} \ominus x_{120}x_{121}x_{124} \ominus$$
$$x_{120}x_{121}x_{124}x_{126} \oplus x_{120}x_{121}x_{124}x_{126}x_{127} \oplus x_{120}x_{121}x_{124}x_{125} \oplus x_{120}x_{121}x_{124}x_{125}x_{126}x_{127} \oplus$$
$$x_{120}x_{121}x_{123}x_{127} \oplus x_{120}x_{121}x_{123}x_{125}x_{127} \oplus x_{120}x_{121}x_{123}x_{125}x_{126} \oplus x_{120}x_{121}x_{123}x_{124}x_{127} \oplus$$
$$x_{120}x_{121}x_{123}x_{124}x_{126}x_{127} \oplus x_{120}x_{121}x_{123}x_{124}x_{125}x_{126} \oplus x_{120}x_{121}x_{123}x_{124}x_{125}x_{126}x_{127} \oplus$$
$$x_{120}x_{121}x_{122} \oplus x_{120}x_{121}x_{122}x_{126} \oplus x_{120}x_{121}x_{122}x_{126}x_{127} \oplus x_{120}x_{121}x_{122}x_{125}x_{126} \oplus$$
$$x_{120}x_{121}x_{122}x_{125}x_{126}x_{127} \oplus x_{120}x_{121}x_{122}x_{124} \oplus x_{120}x_{121}x_{122}x_{124}x_{127} \oplus$$
$$x_{120}x_{121}x_{122}x_{124}x_{125}x_{127} \oplus x_{120}x_{121}x_{122}x_{123}x_{126}x_{127} \oplus x_{120}x_{121}x_{122}x_{123}x_{125} \oplus$$
$$x_{120}x_{121}x_{122}x_{123}x_{125}x_{126} \oplus x_{120}x_{121}x_{122}x_{123}x_{125}x_{126}x_{127} \oplus x_{120}x_{121}x_{122}x_{123}x_{124}x_{126} \oplus$$
$$x_{120}x_{121}x_{122}x_{123}x_{124}x_{125} \oplus x_{120}x_{121}x_{122}x_{123}x_{124}x_{125}x_{127}$$

Figure 8. Equation for the low level bit of the SubByte function.

For this function, we do not need to compute Boolean function. Indeed, the only operation of the ShiftRows transformation consists of switching byte through the State array. In our file this operation can be easily solved by a XOR operation. As an example the second byte of the State array becomes the 6th after ShiftRows function. Therefore to transform this byte we apply the following XOR to the second byte:

6.2.3. Solution for the MixColumns function

The MixColumns transformation operates on the State array column-by-column, treating each column as a four-term polynomial. Each of these columns is multiplied by a square matrix. Thus we have for each column:

$$\begin{pmatrix} b'_i \\ b'_{i+1} \\ b'_{i+2} \\ b'_{i+3} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \bullet \begin{pmatrix} b'_i \\ b'_{i+1} \\ b'_{i+2} \\ b'_{i+3} \end{pmatrix}.$$

So, for the first byte of the column we have this equation:

$$b'_i = 02 \bullet b_i \oplus 03 \bullet b_{i+1} \oplus 01 \bullet b_{i+2} \oplus 01 \bullet b_{i+3}.$$

Since on GF_2^8, 01 is the identity for multiplication, This equation becomes:

$$b'_i = 02 \bullet b_i \oplus 03 \bullet b_{i+1} \oplus b_{i+2} \oplus b_{i+3}.$$

We have the same simplification for all equations describing the column multiplication by this square matrix. At the end we only need to compute the truth table for multiplication by 02 and by 03 over GF_2^8.

6.3. Formatting the equations

To format the equations we use the same representation as for the mini-aes. Thus we have 128 files, one by bit of block. Each line describes a monomial and the shift from one line to another means a XOR operation.

A file sample thus obtained is given in Annexe B.

7. Conclusion

We described the algorithm of the mini-AES and of the AES as Boolean

functions and then we translated it into systems of equations using the Möbius transform of these functions.

Now, the goal of our approach is to be able to perform a combinatorial analysis on the files of equations thus obtained.

The use of a Boolean equation system of low degree is motivated by the fact that its solution is likely to be easier than the existing equation model. We are currently developing a new, combinatorial approach in Boolean equation system solving which seems to be promising.

A. Equations for the first round

$$b'_1 = k_1 \oplus 1 \oplus x_{15}x_{16} \oplus x_{14} \oplus x_{14}x_{16} \oplus x_{13} \oplus x_{13}x_{15} \oplus x_{13}x_{15}x_{16} \oplus x_4 \oplus x_3x_4 \oplus x_2x_4 \oplus x_2x_3 \oplus x_2x_3x_4 \oplus x_1x_3 \oplus x_1x_3x_4 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus 1 \oplus k_{16} \oplus k_{14} \oplus k_{14}k_{15} \oplus k_{14}k_{15}k_{16} \oplus k_{13} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{15} \oplus k_1$$

$$b'_2 = k_2 \oplus 1 \oplus x_{16} \oplus x_{15} \oplus x_{15}x_{16} \oplus x_{14}x_{16} \oplus x_{14}x_{15} \oplus x_{13}x_{16} \oplus x_{13}x_{15} \oplus x_{13}x_{14} \oplus x_{13}x_{14}x_{16} \oplus x_{13}x_{14}x_{15} \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_2x_3 \oplus x_1 \oplus x_1x_4 \oplus x_1x_3x_4 \oplus x_1x_2 \oplus x_1x_2x_4 \oplus x_1x_2x_3 \oplus 1 \oplus k_{15}k_{16} \oplus k_{14} \oplus k_{14}k_{16} \oplus k_{13} \oplus k_{13}k_{15} \oplus k_{13}k_{15}k_{16} \oplus k_2$$

$$b'_3 = k_3 \oplus 1 \oplus x_{16} \oplus x_{15} \oplus x_{14} \oplus x_{14}x_{16} \oplus x_{14}x_{15} \oplus x_{14}x_{15}x_{16} \oplus x_{13}x_{16} \oplus x_{13}x_{15}x_{16} \oplus x_{13}x_{14} \oplus x_{13}x_{14}x_{15} \oplus x_3x_4 \oplus x_2 \oplus x_2x_3x_4 \oplus x_1x_3 \oplus x_1x_3x_4 \oplus x_1x_2x_4 \oplus 1 \oplus k_{16} \oplus k_{15} \oplus k_{15}k_{16} \oplus k_{14}k_{16} \oplus k_{14}k_{15} \oplus k_{13}k_{16} \oplus k_{13}k_{15} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{16} \oplus k_{13}k_{14}k_{15} \oplus k_3$$

$$b'_4 = k_4 \oplus x_{16} \oplus x_{14} \oplus x_{14}x_{15} \oplus x_{14}x_{15}x_{16} \oplus x_{13} \oplus x_{13}x_{14} \oplus x_{13}x_{14}x_{15} \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_2x_4 \oplus x_2x_3 \oplus x_2x_3x_4 \oplus x_1x_4 \oplus x_1x_3x_4 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus 1 \oplus k_{15} \oplus k_{14}k_{16} \oplus k_{13} \oplus k_{13}k_{16} \oplus k_{13}k_{15}k_{16} \oplus k_4$$

$$b'_5 = k_5 \oplus 1 \oplus x_{16} \oplus x_{15}x_{16} \oplus x_{14}x_{16} \oplus x_{14}x_{15} \oplus x_{14}x_{15}x_{16} \oplus x_{13}x_{15} \oplus x_{13}x_{15}x_{16} \oplus x_{13}x_{14} \oplus x_{13}x_{14}x_{15} \oplus x_3x_4 \oplus x_2 \oplus x_2x_4 \oplus x_1 \oplus x_1x_3 \oplus x_1x_3x_4 \oplus 1 \oplus k_{16} \oplus k_{14} \oplus k_{14}k_{15} \oplus k_{14}k_{15}k_{16} \oplus k_{13} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{15} \oplus k_5 \oplus k_1$$

$$b'_6 = k_6 \oplus 1 \oplus x_{16} \oplus x_{15} \oplus x_{14} \oplus x_{14}x_{15} \oplus x_{13} \oplus x_{13}x_{16} \oplus x_{13}x_{15}x_{16} \oplus x_{13}x_{14} \oplus x_{13}x_{14}x_{16} \oplus x_{13}x_{14}x_{15} \oplus x_4 \oplus x_3 \oplus x_3x_4 \oplus x_2x_4 \oplus x_2x_3 \oplus x_1x_4 \oplus x_1x_3 \oplus x_1x_2 \oplus x_1x_2x_4 \oplus x_1x_2x_3 \oplus 1 \oplus k_{15}k_{16} \oplus k_{14} \oplus k_{14}k_{16} \oplus k_{13} \oplus k_{13}k_{15} \oplus k_{13}k_{15}k_{16} \oplus k_6 \oplus k_2$$

$$b'_7 = k_7 \oplus 1 \oplus x_{15}x_{16} \oplus x_{14} \oplus x_{14}x_{15}x_{16} \oplus x_{13}x_{15} \oplus x_{13}x_{15}x_{16} \oplus x_{13}x_{14}x_{16} \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_2x_4 \oplus x_2x_3 \oplus x_2x_3x_4 \oplus x_1x_4 \oplus x_1x_3x_4 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus 1 \oplus k_{16} \oplus k_{15} \oplus k_{15}k_{16} \oplus k_{14}k_{16} \oplus k_{14}k_{15} \oplus k_{13}k_{16} \oplus k_{13}k_{15} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{16} \oplus k_{13}k_{14}k_{15} \oplus k_7 \oplus k_3$$

$$b'_8 = k_8 \oplus x_{16} \oplus x_{15} \oplus x_{14} \oplus x_{14}x_{16} \oplus x_{14}x_{15} \oplus x_{14}x_{15}x_{16} \oplus x_{13}x_{16} \oplus x_{13}x_{15}x_{16} \oplus x_{13}x_{14} \oplus x_{13}x_{14}x_{15} \oplus x_4 \oplus x_2 \oplus x_2x_3 \oplus x_2x_3x_4 \oplus x_1 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus 1 \oplus k_{15} \oplus k_{14}k_{16} \oplus k_{13} \oplus k_{13}k_{16} \oplus k_{13}k_{15}k_{16} \oplus k_8 \oplus k_4$$

$$b'_9 = k_9 \oplus 1 \oplus x_{12} \oplus x_{11}x_{12} \oplus x_{10}x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{11}x_{12} \oplus x_9x_{11} \oplus x_9x_{11}x_{12} \oplus x_9x_{10} \oplus x_9x_{10}x_{11} \oplus x_7x_8 \oplus x_6 \oplus x_6x_8 \oplus x_5 \oplus x_5x_7 \oplus x_5x_7x_8 \oplus 1 \oplus k_{16} \oplus k_{14} \oplus k_{14}k_{15} \oplus k_{14}k_{15}k_{16} \oplus k_{13} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{15} \oplus k_9 \oplus k_5 \oplus k_1$$

$$b'_{10} = k_{10} \oplus 1 \oplus x_{12} \oplus x_{11} \oplus x_{10} \oplus x_{10}x_{11} \oplus x_9 \oplus x_9x_{12} \oplus x_9x_{11}x_{12} \oplus x_9x_{10} \oplus x_9x_{10}x_{12} \oplus x_9x_{10}x_{11} \oplus x_8 \oplus x_7 \oplus x_7x_8 \oplus x_6x_8 \oplus x_6x_7 \oplus x_5x_8 \oplus x_5x_7 \oplus x_5x_6 \oplus x_5x_6x_8 \oplus x_5x_6x_7 \oplus 1 \oplus k_{15}k_{16} \oplus k_{14} \oplus k_{14}k_{16} \oplus k_{13} \oplus k_{13}k_{15} \oplus k_{13}k_{15}k_{16} \oplus k_{10} \oplus k_6 \oplus k_2$$

$$b'_{11} = k_{11} \oplus 1 \oplus x_{11}x_{12} \oplus x_{10} \oplus x_{10}x_{11}x_{12} \oplus x_9x_{11} \oplus x_9x_{11}x_{12} \oplus x_9x_{10}x_{12} \oplus x_8 \oplus x_7 \oplus x_6 \oplus x_6x_8 \oplus x_6x_7 \oplus x_6x_7x_8 \oplus x_5x_8 \oplus x_5x_7x_8 \oplus x_5x_6 \oplus x_5x_6x_7 \oplus 1 \oplus k_{16} \oplus k_{15} \oplus k_{15}k_{16} \oplus k_{14}k_{16} \oplus k_{14}k_{15} \oplus k_{13}k_{16} \oplus k_{13}k_{15} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{16} \oplus k_{13}k_{14}k_{15} \oplus k_{11} \oplus k_7 \oplus k_3$$

$$b'_{12} = k_{12} \oplus x_{12} \oplus x_{11} \oplus x_{10} \oplus x_{10}x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{11}x_{12} \oplus x_9x_{12} \oplus x_9x_{11}x_{12} \oplus x_9x_{10} \oplus x_9x_{10}x_{11} \oplus x_8 \oplus x_6 \oplus x_6x_7 \oplus x_6x_7x_8 \oplus x_5 \oplus x_5x_6 \oplus x_5x_6x_7 \oplus 1 \oplus k_{15} \oplus k_{14}k_{16} \oplus k_{13} \oplus k_{13}k_{16} \oplus k_{13}k_{15}k_{16} \oplus k_{12} \oplus k_8 \oplus k_4$$

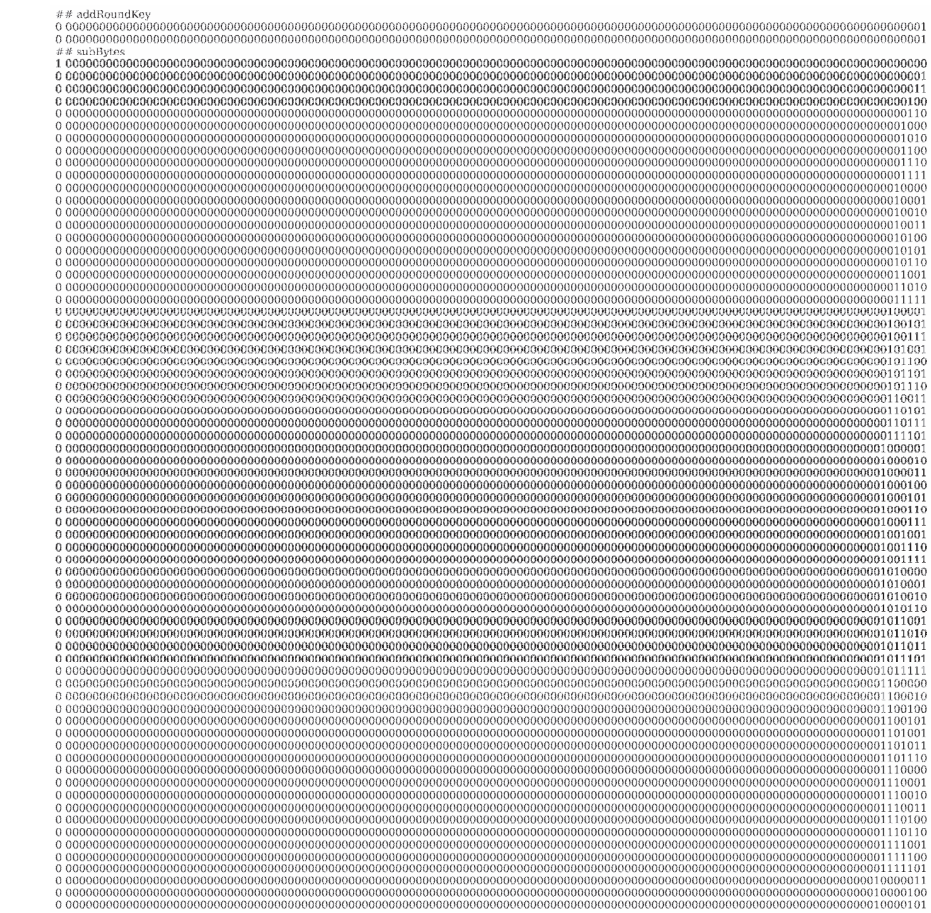
$$b'_{13} = k_{13} \oplus 1 \oplus x_{11}x_{12} \oplus x_{10} \oplus x_{10}x_{12} \oplus x_9 \oplus x_9x_{11} \oplus x_9x_{11}x_{12} \oplus x_8 \oplus x_7x_8 \oplus x_6x_8 \oplus x_6x_7 \oplus x_6x_7x_8 \oplus x_5x_7 \oplus x_5x_7x_8 \oplus x_5x_6 \oplus x_5x_6x_7 \oplus 1 \oplus k_{16} \oplus k_{14} \oplus k_{14}k_{15} \oplus k_{14}k_{15}k_{16} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{15} \oplus k_9 \oplus k_5 \oplus k_1$$

$$b'_{14} = k_{14} \oplus 1 \oplus x_{12} \oplus x_{11} \oplus x_{11}x_{12} \oplus x_{10}x_{12} \oplus x_{10}x_{11} \oplus x_9x_{12} \oplus x_9x_{11} \oplus x_9x_{10} \oplus x_9x_{10}x_{12} \oplus x_9x_{10}x_{11} \oplus x_8 \oplus x_7 \oplus x_6 \oplus x_6x_7 \oplus x_5 \oplus x_5x_8 \oplus x_5x_7x_8 \oplus x_5x_6 \oplus x_5x_6x_8 \oplus x_5x_6x_7 \oplus 1 \oplus k_{15}k_{16} \oplus k_{14}k_{16} \oplus k_{13} \oplus k_{13}k_{15} \oplus k_{13}k_{15}k_{16} \oplus k_{10} \oplus k_6 \oplus k_2$$

$$b'_{15} = k_{15} \oplus 1 \oplus x_{12} \oplus x_{11} \oplus x_{10} \oplus x_{10}x_{12} \oplus x_{10}x_{11} \oplus x_{10}x_{11}x_{12} \oplus x_9x_{12} \oplus x_9x_{11}x_{12} \oplus x_9x_{10} \oplus x_9x_{10}x_{11} \oplus x_7x_8 \oplus x_6 \oplus x_6x_7x_8 \oplus x_5x_7 \oplus x_5x_7x_8 \oplus x_5x_6x_8 \oplus 1 \oplus k_{16} \oplus k_{15}k_{16} \oplus k_{14}k_{16} \oplus k_{14}k_{15} \oplus k_{13}k_{16} \oplus k_{13}k_{15} \oplus k_{13}k_{14} \oplus k_{13}k_{14}k_{16} \oplus k_{13}k_{14}k_{15} \oplus k_{11} \oplus k_7 \oplus k_3$$

$$b'_{16} = k_{16} \oplus x_{12} \oplus x_{10} \oplus x_{10}x_{11} \oplus x_{10}x_{11}x_{12} \oplus x_9 \oplus x_9x_{10} \oplus x_9x_{10}x_{11} \oplus x_8 \oplus x_7 \oplus x_6 \oplus x_6x_8 \oplus x_6x_7 \oplus x_6x_7x_8 \oplus x_5x_8 \oplus x_5x_7x_8 \oplus x_5x_6 \oplus x_5x_6x_7 \oplus 1 \oplus k_{16} \oplus k_{15} \oplus k_{14}k_{16} \oplus k_{13} \oplus k_{13}k_{16} \oplus k_{13}k_{15}k_{16} \oplus k_{12} \oplus k_8 \oplus k_4$$

B. File of the low level bit block describing the first round of the AES



```
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010001010
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010001100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010001101
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010001110
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010001111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010010001
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010010011
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010010100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010010101
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010010110
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010010111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010011000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010011111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010100000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010100100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010100101
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010100111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010101000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010101011
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010101100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010101110
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010101111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010110001
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010110010
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010110100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010110101
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010110111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010111001
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010111010
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010111100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010111101
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011000000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011000100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011000110
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011000111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011001000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011001010
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011001011
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011001100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011001111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011010001
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011010101
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011010110
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011011001
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011011011
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011011110
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011011111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011100000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011100010
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011100011
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011100110
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011100111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011101000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011101001
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011101101
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011110011
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011110100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011110110
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011110111
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111010
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111100
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111101
## shiftRows
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000
## mixColumns
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000000000000000000000000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000000000000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000100000000
0 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000
## end
```

References

[1] E. F. Assmus, On the Reed-Muller Codes, Vol. 106, 1992.

[2] Carlos Cid, Sean Murphy and Matthew Robshaw, Small scale variants of the advanced encryption standard, Fast Software Encryption 2005, LCNS 3557 (2005), 145-162.

[3] Carlos Cid, Sean Murphy and Matthew Robshaw, Algebraic Aspects of the Advanced Encryption Standard, Springer, 2006.

[4] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations, Vol. 2501, 2002.

[5] Éric Filiol, Reconstruction Techniques in Cryptology and in Coding Theory, Ph.D. Thesis, École Polytechnique, 2005.

[6] FIPS, Advanced Encryption Standard, 2001.

[7] Richard Wesley Hamming, Error Detecting and Error Correcting Codes, Vol, 29, 1950.

[8] Paul McCarty, Introduction to Arithmetical Functions, Springer Verlag, 1986.

[9] Sean Murphy and Matthew Robshaw, Essential Algebraic Structure Within the AES, Volume LCNS 2442, 2002.

[10] Raphael Chung-Wei Phan, Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students, Vol 26, 2002.

[11] Vincent Rijmen, Joan Daemen, Bart Preneel and Antoon Bosselaers, The cipher shark, https://www.cosic.esat.kuleuven.be/publications/article-55.pdf, 1996.

[12] Bassem Sakkour, Étude et amélioration du décodage des codes de Reed-Muller d'ordre deux, 2007.

[13] Claude Elwood Shannon, A Mathematical Theory of Communication, Vol. 27, 1948.

[14] Claude Elwood Shannon, Communication theory of secrecy systems, Bell System Technical Journal, 28, 1949.

[15] Wikipedia, Cryptanalysis, 2011.