

The Advanced Encryption Standard: Rijndael

K. Cartrysse and J.C.A. van der Lubbe

Supplement to the books

”Basic methods of cryptography”

and

”Basismethoden cryptografie”

October 2004

Contents

1	Introduction	2
2	Mathematical tools	2
2.1	Fields and polynomial arithmetic	2
2.2	Rijndael and $GF(2^8)$	6
2.2.1	The field $GF(2^8)$	6
2.2.2	Polynomials with coefficients in $GF(2^8)$	6
3	Overview of Rijndael	8
4	AddRoundKey	10
5	SubBytes	10
6	ShiftRows	13
7	MixColumns	13
8	Key schedule	15
9	Decryption	18
10	Some words on the security of Rijndael	19
11	References	20

1. Introduction

In 1997 the National Institute of Standards and Technology (NIST) of the United States put out a call for proposals for a new symmetric algorithm, that will be called the Advanced Encryption Standard (AES). The algorithm DES was since 1976 the standard for symmetric algorithms, and a replacement was necessary. In 1998 15 candidate algorithms were accepted, and after one year of research 5 of these candidates were announced as finalists:

- MARS (IBM corporation)
- RC6 (RSA Laboratories)
- Rijndael (J. Daemen & V. Rijmen)
- Serpent (E. Biham et al.)
- Twofish (B. Schneier et al.)

On all of these algorithms extensive research has been done to find attacks or weaknesses. According to NIST, all 5 finalists appear to offer adequate security. Also much research has been done to test the performs of these 5 algorithms in both software and hardware. In 2000 NIST announced that Rijndael was chosen as the successor of DES, the AES. The combination of security, performance, efficiency, implementability and flexibility made Rijndael an appropriate selection for the AES.

The candidates for the AES algorithm had to fulfill certain design criteria. First, of course the algorithm should be a symmetric algorithm and it must be resistant against all known attacks. Furthermore, the AES must be efficient in performance and memory for different platforms. The design must be simple, and it should be able to handle different key lengths (128, 192 and 256 bits). The block length of the cipher should be 128 bits.

This chapter gives a description of the Rijndael algorithm.

2. Mathematical tools

This section gives a short introduction to the mathematics that are used in Rijndael. Rijndael uses a finite field of the form $GF(2^8)$, and here the basics of (finite) fields are presented such that the Rijndael algorithm can be understood. GF stands for Galois Field and is an other name for finite field.

2.1. Fields and polynomial arithmetic

The computations done in the Rijndael algorithm are mainly done in the finite field $GF(2^8)$. A field is a commutative ring in which all non-zero elements have multiplicative inverses. First these terms are explained by the following definitions:

Definition 1 (Ring($R, +, \times$)) A ring $(R, +, \times)$ consists of a set R with two binary operations arbitrarily denoted $+$ (addition) and \times (multiplication) on R , satisfying the following axioms.

1. $(R, +)$ is an abelian group with identity denoted 0 .
2. The operation \times is associative. That is, $a \times (b \times c) = (a \times b) \times c$ for all $a, b, c \in R$.
3. There is a multiplicative identity denoted 1 , with $1 \neq 0$, such that $1 \times a = a \times 1 = a$ for all $a \in R$.
4. The operation \times is distributive over $+$. That is, $a \times (b + c) = (a \times b) + (a \times c)$ and $(b + c) \times a = (b \times a) + (c \times a)$ for all $a, b, c \in R$.

The ring is a commutative ring if $a \times b = b \times a$ for all $a, b \in R$.

Definition 2 A group $(G, *)$ consists of a set G with a binary operation $*$ on G satisfying the following three axioms.

1. The group operation is associative. That is $a*(b*c) = (a*b)*c$ for all $a, b, c \in G$.
2. There is an element $1 \in G$, called the identity element of a , such that $a * 1 = 1 * a = a$ for all $a \in G$
3. For each $a \in G$ there exists an element $a^{-1} \in G$, called the inverse of a , such that $a * a^{-1} = a^{-1} * a = 1$.

A group G is abelian (or commutative) if, furthermore,

4. $a * b = b * a$ for all $a, b \in G$.

The abelian group that is used in the definition of a Ring, is the abelian group with addition as operation. For reasons of completeness here a definition of an inverse is given when the operation multiplication is used. A *multiplicative inverse* of element a is an element b , such that $a \times b = 1$. For example the set of integers \mathbb{Z} with the operations addition and multiplication is a commutative ring. Furthermore the set $\mathbb{Z}_n : \{0, 1, \dots, n-1\}$ with addition and multiplication performed *modulo* n is a commutative ring. For an explanation on computations modulo n , we refer to the book "Basic methods of cryptography" chapter 6.

In Rijndael finite fields are used, where finite means there are a limited number of elements in the field. $GF(p)$ has p elements. For example $GF(7)$ is a finite field with 7 elements.

The representation of the field elements used in Rijndael is the polynomial representation. Each element is represented by a polynomial. When a finite field $GF(p^n)$ is used, this means that the coefficients of the elements are modulo p and an irreducible polynomial $f(x)$ is chosen of degree n . All computations in this field are done modulo $f(x)$. An irreducible polynomial of degree n is a polynomial that does not factor (except trivial factor 1) into smaller polynomials from $GF(p^n)$. In a finite field multiple irreducible polynomials may exist. To understand the concept of these irreducible polynomials, you can use them in the same way as you use prime numbers. Just as with prime numbers there is no straight forward way to generate irreducible polynomials. It is possible to calculate how many irreducible polynomials exist in a finite field, but to generate one a polynomial is taken and then tested whether this is an irreducible polynomial. This is a similar process as the generation of large prime numbers, where primality tests are used to give a degree of certainty that a number is prime. We know that prime numbers are always odd (except for the number 2), for irreducible polynomials we know that the coefficient of x^0 is always 1. The field used in Rijndael is $GF(2^8)$.

Let's look at an example of a finite field represented by polynomials. Consider $GF(2^3)$ with the irreducible polynomial $f(x)$ of degree $n = 3$:

$$f(x) = x^3 + x + 1. \quad (1)$$

All elements in $GF(2^3)$ are polynomials with degree 2 or smaller. The calculations on coefficients of the polynomials are performed modulo $p = \text{modulo } 2$ (e.g. all coefficients can only take the values 0 and 1), while the computations on the polynomials are done modulo $f(x)$. Then $GF(2^3)$ exists out of the elements $\{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$. In this field the operations addition and multiplication are defined. To add two polynomials in the field $GF(2^3)$ an EXOR operation is done on the coefficients because $p = 2$. For example consider $g(x) = x^2 + x + 1$ and $h(x) = x^2 + 1$ then $k(x) = g(x) + h(x) = x$.

Multiplication is also defined for $GF(2^3)$. We wish to multiply $g(x)$ and $h(x)$:

$$\begin{aligned} k(x) = g(x) \cdot h(x) \bmod f(x) &= (x^2 + x + 1)(x^2 + 1) \bmod f(x) \\ &= x^4 + x^3 + x + 1 \bmod f(x). \end{aligned}$$

Now the modulus calculation must be performed as $k(x)$ is not an element in $GF(2^3)$. Just as with numbers the modulus operation is equal to calculating the remainder of the division. Division with polynomials can be done using long division:

$$\begin{array}{r} x^3 + x + 1 \quad / \quad x^4 + x^3 + x + 1 \setminus x + 1 \\ \underline{x^4 + x^2 + x} \\ x^3 + x^2 + 1 \end{array}$$

$$\frac{x^3 + x + 1}{x^2 + x}$$

Therefore, $x^4 + x^3 + x + 1 \bmod (x^3 + x + 1) = x^2 + x$. This can be easily checked by computing:

$$(x + 1)(x^3 + x + 1) + (x^2 + x)$$

and this is equal to $x^4 + x^3 + x + 1 \bmod (x^3 + x + 1)$.

It is also possible to divide elements within the field, but this is a little more complicated. To compute $\frac{h(x)}{g(x)} \bmod f(x)$, this is equal to $h(x) \cdot g^{-1}(x) \bmod f(x)$. When $f(x)$ is an irreducible polynomial the $g^{-1}(x)$ will exist. To calculate the inverse in case numbers are used, the extended euclidean algorithm can be used. The same can be done for polynomials. We wish to calculate the inverse of element $g(x)$. We use the extended euclidean algorithm for polynomials, which is equivalent to the extended euclidean algorithm used for numbers as is described on page 138 of "Basic methods of cryptography":

$$\begin{aligned} r(0) &= a(1) \cdot r(1) + r(2) \\ r(1) &= a(2) \cdot r(2) + r(3) \\ r(2) &= a(3) \cdot r(3) + r(4) \\ &\vdots \\ r(k-2) &= a(k-1) \cdot r(k-1) + r(k) \end{aligned}$$

Then $r(k)$ must be expressed in terms of $r(0)$ and $r(1)$, such that $r(k) = u \cdot r(0) + v \cdot r(1)$, then v is equal to the inverse of $g(x)$, because if $r(k) = 1$ then $1 = u \cdot r(0) + v \cdot r(1) = u \cdot f(x) + v \cdot g(x)$. From this it follows that $v = g^{-1}(x) \bmod f(x)$.

We show the above by using an example. Consider again the field $GF(2^3)$ with irreducible polynomial $f(x) = x^3 + x + 1$ and we will compute the inverse of $g(x) = x^2$

$$\begin{aligned} x^3 + x + 1 &= (x)x^2 + (x + 1) &\implies r(2) &= r(0) + xr(1) \\ x &= x(x + 1) + x &\implies r(3) &= r(1) + xr(2) \\ & &\implies r(3) &= xr(0) + (1 + x^2)r(1) \\ x + 1 &= (1)x + 1 &\implies r(4) &= r(2) + r(3) \\ & &r(4) &= (1 + x)r(0) + (1 + x + x^2)r(1) \end{aligned}$$

Hence the inverse of $x^2 \bmod f(x)$ is $x^2 + x + 1$. Long divisions can be used to obtain the values from each intermediate step. To check whether the result is correct the multiplication of x^2 and $(x^2 + x + 1)$ can be performed modulo $x^3 + x + 1$. The result should be equal to 1. using the multiplicative inverse it is then possible to compute the division of $\frac{h(x)}{g(x)} \bmod f(x)$ by performing the multiplication $h(x)g^{-1}(x) \bmod f(x)$.

2.2. Rijndael and $GF(2^8)$

2.2.1. The field $GF(2^8)$

Rijndael uses the finite field $GF(2^8)$. The irreducible polynomial that is used is:

$$f(x) = x^8 + x^4 + x^3 + x + 1. \quad (2)$$

First, something must be said about the notation used to describe the algorithm. A byte b , consisting of bits $b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$ is considered as a polynomial with coefficients modulo 2. The polynomial will look like:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0. \quad (3)$$

The notation used to describe Rijndael are hexadecimal numbers. These numbers must be converted first to binary numbers such that the coefficients of the polynomial can be determined.

For example the hexadecimal number 'D4' can binary be represented as '11010100'. This represents the polynomial

$$x^7 + x^6 + x^4 + x^2.$$

In the previous section addition was explained within $GF(p^n)$, with $GF(2^3)$ as an example. The sum of two polynomials is the sum modulo 2 of the coefficients of the two terms. For example: ' $D4$ ' + ' $E0$ ' = ' 34 '. In polynomial representation this looks as follows:

$$\begin{aligned} 'D4' &= x^7 + x^6 + x^4 + x^2 \\ 'E0' &= x^7 + x^6 + x^5 \\ 'D4 + 'E0' &= x^5 + x^4 + x^2 = '34', \end{aligned}$$

which is a simple EXOR operation on the byte level.

Using an equivalent approach as above also multiplication and division can be done in $GF(2^8)$. It is important to remember that when hexadecimal numbers are used, they must be seen as a polynomial.

2.2.2. Polynomials with coefficients in $GF(2^8)$

In the previous sections all the polynomials had coefficients modulo 2. However, in Rijndael sometimes a 4-byte vector is considered as a polynomial with coefficients in $GF(2^8)$. Again addition and multiplication are defined.

The addition of two vectors is a simple bitwise EXOR operation, as the addition in $GF(2^8)$ is a bitwise EXOR. For example, consider two polynomials over $GF(2^8)$ (each coefficient is one byte):

$$\begin{aligned} a(x) &= a_3x^3 + a_2x^2 + a_1x + a_0 \\ b(x) &= b_3x^3 + b_2x^2 + b_1x + b_0, \end{aligned}$$

then

$$\begin{aligned} c(x) &= a(x) + b(x) \\ &= (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0). \end{aligned}$$

Multiplication is more complicated as the coefficients are not in $GF(2)$ anymore (as in the previous paragraphs), but in $GF(2^8)$. Consider again the polynomials $a(x)$ and $b(x)$ with coefficients in $GF(2^8)$.

The product $c(x) = a(x) \cdot b(x)$ is:

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0.$$

Where the coefficients are given by:

$$\begin{aligned} c_0 &= a_0 \cdot b_0 \\ c_1 &= a_1 \cdot b_0 \oplus a_0 \cdot b_1 \\ c_2 &= a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2 \\ c_3 &= a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3 \\ c_4 &= a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3 \\ c_5 &= a_3 \cdot b_2 \oplus a_2 \cdot b_3 \\ c_6 &= a_3 \cdot b_3 \end{aligned}$$

The polynomial $c(x)$ does not fit in a 4-byte vector anymore, therefore it is reduced to a 4-byte vector by calculating $c(x) \bmod M(x)$, where $M(x) = x^4 + 1$. Then the modular product of $a(x)$ and $b(x)$ is given by:

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0.$$

The coefficients of $d(x)$ can be obtained by taking the remainder of a long division between $c(x)$ and $M(x)$. Then, the coefficients of $d(x)$ are:

$$\begin{aligned} d_0 &= a_0 \cdot b_0 \oplus a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3 \\ d_1 &= a_1 \cdot b_0 \oplus a_0 \cdot b_1 \oplus a_3 \cdot b_2 \oplus a_2 \cdot b_3 \\ d_2 &= a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2 \oplus a_3 \cdot b_3 \\ d_3 &= a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3 \end{aligned}$$

This can be written as a matrix multiplication:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (4)$$

3. Overview of Rijndael

First an overview of the Rijndael algorithm is given. Figure 1 shows the different phases of the Rijndael algorithm. It starts with an initial round followed by a number of standard rounds and it ends with the final round. Only four different operations are necessary to compute these rounds and a key schedule. Each of these operations are described separately in the next chapters.

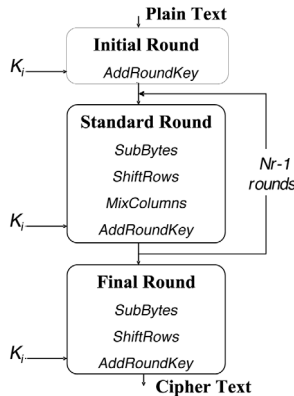


Figure 1: Overview of Rijndael

It is possible in Rijndael to use different keylengths according to the security level that is required for the application. Rijndael is defined as a block cipher with key lengths of 128, 192 or 256 bits. The possible input block lengths are 128, 192 or 256 for the Rijndael algorithm. The AES algorithm is exactly the same as the Rijndael algorithm, but it only defines one block length of 128 bits.

The Rijndael algorithm is such that each bit is dependent on all bits from 2 rounds ago, e.g. full diffusion is provided. The number of rounds that must be run is dependent on the key length, see table 3.

In the description of the Rijndael cipher the intermediate cipher result will be called the *State*. Matrix notations can be used to represent the state. The matrix structure is

Table 1: Number of rounds (a word is 32 bits)

	Key length (words)	Number of rounds (N_r)
AES-128	4	10
AES-192	6	12
AES-256	8	14

such that there are always 4 rows and the number of columns is variable depending on the number of bits chosen for block length and key length. A key of 192 bits for example is a (4, 6) matrix with one byte in each element:

$$\begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} & k_{0,4} & k_{0,5} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} & k_{1,4} & k_{1,5} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} & k_{2,4} & k_{2,5} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} & k_{3,4} & k_{3,5} \end{bmatrix}$$

A block length of 128 bits is represented in a (4, 4) matrix:

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

The number of columns in the input block is called N_b , which is equal to the block length divided by 32. The parameter N_k is used to denote the number of columns in the key. It is possible to combine all block lengths with all different key lengths.

For example, consider the following input and key:

Input: 32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 07 34
 Key: 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C

This is represented as:

$$Input = \begin{bmatrix} 32 & 88 & 31 & E0 \\ 43 & 5A & 31 & 37 \\ F6 & 30 & 98 & 07 \\ A8 & 8D & A2 & 34 \end{bmatrix} \quad Key = \begin{bmatrix} 2B & 28 & AB & 09 \\ 7E & AE & F7 & CF \\ 15 & D2 & 15 & 4F \\ 16 & A6 & 88 & 3C \end{bmatrix}$$

where $N_k = N_b = 4$.

As with other block ciphers, Rijndael can be used in several modes, such as ECB, CBC, and CFB. The next sections describe each phase of the algorithm separately.

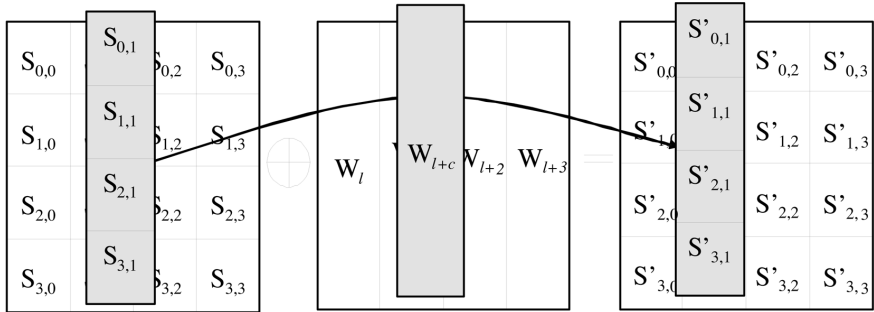


Figure 2: AddRoundKey.

4. AddRoundKey

The AddRoundKey (figure 2) operation is a simple EXOR operation between the *State* and the *RoundKey*. The *RoundKey* is derived from the *Cipherkey* by means of the key schedule as is described in section 5. The *State* and *RoundKey* are of the same size and to obtain the next *State* an EXOR operation is done per element:

$$s'(i, j) = s(i, j) \oplus w(i, j). \quad (5)$$

where s is the current *State*, s' the next *State* and w the round key.

Example. Consider the following *State* s and *RoundKey* w :

$$s = \begin{bmatrix} 32 & 88 & 31 & E0 \\ 43 & 5A & 31 & 37 \\ F6 & 30 & 98 & 07 \\ A8 & 8D & A2 & 34 \end{bmatrix} \quad w = \begin{bmatrix} 2B & 28 & AB & 09 \\ 7E & AE & F7 & CF \\ 15 & D2 & 15 & 4F \\ 16 & A6 & 88 & 3C \end{bmatrix}$$

Then the next *State* s' is:

$$s' = \begin{bmatrix} 19 & A0 & 9A & E9 \\ 3D & F4 & C6 & F8 \\ E3 & E2 & 8D & 48 \\ BE & 2B & 2A & 08 \end{bmatrix}$$

5. SubBytes

The operation SubBytes is similar to the S-boxes used in the DES-algorithm. Rijndael has only one S-box. The design criteria for the S-box are such that it is resistant

Table 2: The AES S-box.

		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
x	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

against the known differential and linear cryptanalysis and attack using algebraic manipulations. The S-box is shown in table 2. The x-coordinate represents the first digit of the hexadecimal number and y the second, hence for '08': x=0 and y=8.

Example. Consider the SubBytes operation on the *State*:

$$s = \begin{bmatrix} 19 & A0 & 9A & E9 \\ 3D & F4 & C6 & F8 \\ E3 & E2 & 8D & 48 \\ BE & 2B & 2A & 08 \end{bmatrix} \text{ gives } s' = \begin{bmatrix} D4 & E0 & B8 & 1E \\ 27 & BF & B4 & 41 \\ 11 & 98 & 5D & 52 \\ AE & F1 & E5 & 30 \end{bmatrix}$$

Unlike the design of the S-boxes in the DES algorithm, which is kept secret, the design of the S-box is public. The SubBytes transformation is a non-linear byte substitution, operating on each of the *State* bytes independently. The S-box is invertible and is constructed by the composition of two transformations:

1. Of each element the multiplicative inverse in $GF(2^8)$ is computed, where the representation '00' is mapped onto itself.
2. Then, an affine transformation (over $GF(2)$) is applied. An affine cipher is a cipher of the following form:

$$e_k(x) = ax + b \text{ mod } n,$$

$$\begin{aligned}
(x^8 + x^4 + x^3 + x + 1) &= \\
(x^3 + x^2)(x^5 + x^4 + x^3 + x^2 + 1) + (x^2 + x + 1) &\implies r(2) = r(0) + (x^3 + x^2)r(1) \\
(x^5 + x^4 + x^3 + x^2 + 1) = (x^3 + 1)(x^2 + x + 1) + x &\implies r(3) = (x^3 + 1)r(0) + \\
&\quad (x^6 + x^5 + x^3 + x^2 + 1)r(1) \\
(x^2 + x + 1) = (x + 1)(x) + 1 &\implies r(4) = (x^4 + x^3 + x)r(0) + \\
&\quad (x^7 + x^5 + x^4 + x^3 + x + 1)r(1)
\end{aligned}$$

where the key $k = (a, b)$. The affine cipher in Rijndael is:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (6)$$

The first step of this generation of the S-box, the computation of the inverse, is the transformation that causes the non-linearity in Rijndael, which is an important aspect for the security of Rijndael.

Example. Here it is shown by example how an element of the S-box can be computed. We will calculate the S-box value when the input is $a = '3D'$. Written in polynomial representation, this is:

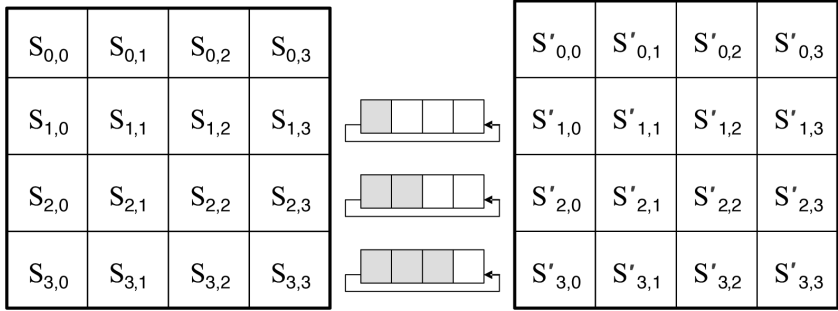
$$a = '3D' \equiv x^5 + x^4 + x^3 + x^2 + 1 \quad (7)$$

First the multiplicative inverse of a is calculated:

hence the inverse of a is: $(x^7 + x^5 + x^4 + x^3 + x + 1)$. In binary representation this is $x_7x_6x_5x_4x_3x_2x_1x_0 = 10111011$, this is the input to the affine transformation. The following matrix computation must then be performed:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

and '00100111' is equal to '27', which corresponds to the S-box entry for '3D'.

Figure 3: ShiftRows for $Nb = 4$.

6. ShiftRows

In ShiftRows (figure 3), the rows of *State* are cyclically shifted with different offsets. Row 1 is shifted over c_1 bytes, row 2 over c_2 bytes, and row 3 over c_3 bytes. The values of c_1 , c_2 , and c_3 depend on the block length Nb :

Nb	c_1	c_2	c_3
4	1	2	3
6	1	2	3
8	1	3	4

Example. The ShiftRows operation on *State* is:

$$s = \begin{bmatrix} D4 & E0 & B8 & 1E \\ 27 & BF & B4 & 41 \\ 11 & 98 & 5D & 52 \\ AE & F1 & E5 & 30 \end{bmatrix} \text{ gives } s' = \begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix}$$

7. MixColumns

The MixColumn transformation is an operation on the different columns. Figure 4 shows the operation.

To calculate the MixColumn transformation the columns of the current state are considered as polynomials over $GF(2^8)$, e.g. the coefficients of the polynomial are elements of $GF(2^8)$. Each column (each polynomial) is multiplied by the polynomial $a(x) \bmod (x^4 + 1)$:

$$a(x) = 03x^3 + 01x^2 + 01x + 02. \quad (8)$$

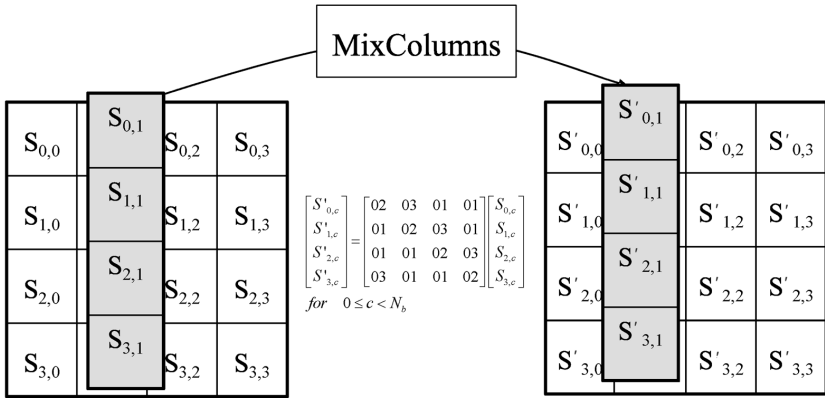


Figure 4: MixColumns.

Using the results of paragraph 2.2.2, it is possible to write this as a matrix multiplication, where $b = b_3b_2b_1b_0$ is a column of *State*:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{9}$$

Example. *State* is given by:

$$s = \begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix}$$

Here it is shown for one element how the next *State* can be computed. To compute one element of the next *state*, one column of the current *State* is taken as input. The value of $s'(1, 1)$ is only dependent of the first column of s . This column can be written as a vector A and can be represented by polynomials:

$$A = \begin{bmatrix} D4 \\ E0 \\ B8 \\ 1E \end{bmatrix} \equiv \begin{bmatrix} x^7 + x^6 + x^4 + x^2 \\ x^7 + x^5 + x^4 + x^3 + x^2 + x + 1 \\ x^6 + x^4 + x^3 + x^2 + 1 \\ x^5 + x^4 \end{bmatrix}$$

This vector A must be used to compute vector D of equation (9). Of course the elements of the matrix in (9) must first be written as polynomials. The matrix multiplication will then look like:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix} \begin{bmatrix} x^7+x^6+x^4+x^2 \\ x^7+x^5+x^4+x^3+x^2+x+1 \\ x^6+x^4+x^3+x^2+1 \\ x^5+x^4 \end{bmatrix}$$

To compute only the first element of the new state, the first row of the matrix must be multiplied by vector A . This results in the following:

$$\begin{aligned} s'(1,1) &= x(x^7+x^6+x^4+x^2) \oplus (x+1)(x^7+x^5+x^4+x^3+x^2+x+1) \\ &\quad \oplus 1(x^6+x^4+x^3+x^2+1) \oplus 1(x^5+x^4) = x^2. \end{aligned}$$

If the outcome were of a larger degree than 7, long division must be performed to calculate the value modulus $f(x)$. The binary representation of x^2 is '00000100', which is '04' in hexadecimal numbers.

The MixColumns operation for the entire *State* is then:

$$s' = \begin{bmatrix} 04 & E0 & 48 & 28 \\ 66 & CB & F8 & 06 \\ 81 & 19 & D3 & 26 \\ E5 & 9A & 7A & 4C \end{bmatrix}$$

8. Key schedule

The *RoundKeys* are derived from the *CipherKey* by means of a key schedule (figure 5).

The number of *RoundKeys* necessary to encrypt one block of information depends on the block length and key length as this determines the number of rounds. For a block length of 128 bits, 11 *RoundKeys* (1 for initial round, 9 for standard rounds and 1 for the final round) are needed. The keys are generated recursively. Again the *CipherKey* is described in a matrix (in case of 128 bits):

$$K = \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}$$

The i -th column of K is denoted by W_i . The key schedule is basically a method to extend K with more columns (we will call the extended version W). A distinction for

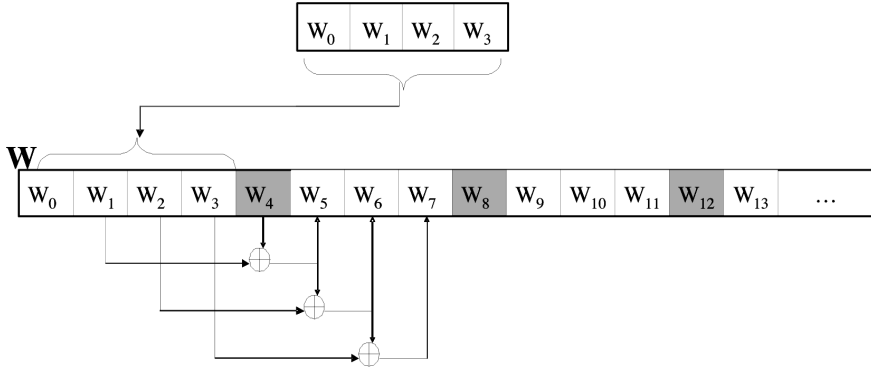


Figure 5: Key schedule.

the key expansion scheme must be made between the cases $N_k \leq 6$ and $N_k > 6$. When $N_k \leq 6$, the key is expanded as follows:

$$W_i = \begin{cases} W_{i-N_k} \oplus \text{SubWord}(S_1(W_{i-1})) \oplus \text{rcon}(\frac{i}{N_k}) & \text{if } i \bmod N_k = 0 \\ W_{i-N_k} \oplus W_{i-1} & \text{if } i \bmod N_k \neq 0 \end{cases}$$

The function $S_1(W_{i-1})$ is a cyclic shift of the elements in W_{i-1} . If W_{i-1} is represented as $[a, b, c, d]$, then $S_1(W_{i-1})$ is given by $[b, c, d, a]$. The function SubWord is a SubBytes operation on each element of the vector separately. $\text{rcon}(\frac{i}{N_k})$ is a vector, that is defined as $\text{rcon}(i) = [x^{i-1}, '00', '00', '00']$, with x^{i-1} being powers of x in the field $GF(2^8)$.

When $N_k > 6$, a small change occurs in the key expansion scheme:

$$W_i = \begin{cases} W_{i-N_k} \oplus \text{SubWord}(S_1(W_{i-1})) \oplus \text{rcon}(\frac{i}{N_k}) & \text{if } i \bmod N_k = 0 \\ W_{i-N_k} \oplus \text{SubWord}(W_{i-1}) & \text{if } i \bmod N_k = 4 \\ W_{i-N_k} \oplus W_{i-1} & \text{elsewhere} \end{cases}$$

When $i - 4$ is a multiple of N_k then SubWord is applied to W_{i-1} prior to the XOR operation. From the matrix W , the *RoundKeys* can be easily extracted. The first N_b columns of W form the key for the initial round and the second N_b columns the key for the first standard round, etc... (see figure 6).

Example. When the block length is 128 ($N_b = 4$), and a key is chosen of 128 bits ($N_k = 4$), the encryption will need 11 *RoundKeys*. Let the key be given by:

Key: 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C

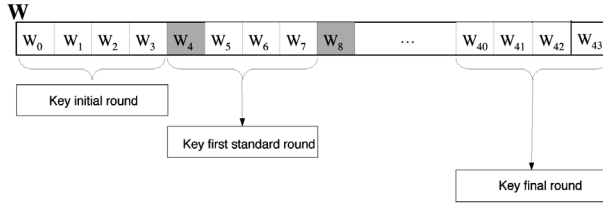


Figure 6: Extraction of RoundKeys from W in case of a block length and key length of 128 bits.

In matrix representation this looks as follows:

$$\text{Key } K = \begin{bmatrix} 2B & 28 & AB & 09 \\ 7E & AE & F7 & CF \\ 15 & D2 & 15 & 4F \\ 16 & A6 & 88 & 3C \end{bmatrix}$$

In order to obtain a sufficient amount of *RoundKeys* this matrix must be extended by 40 columns (as one *RoundKey* exists of 4 columns). In the key expansion scheme, the elements W_i for $0 \leq i \leq 3$ are simply the i -th columns of K . For the remaining W_i the key schedule is followed. Here it is shown how W_4 can be computed. First W_3 must be shifted cyclically, this results in:

$$\begin{bmatrix} CF \\ 4F \\ 3C \\ 09 \end{bmatrix}$$

The *SubWord* operation gives:

$$\begin{bmatrix} 8A \\ 84 \\ EB \\ 01 \end{bmatrix}$$

The $rcon(\frac{i}{N_k})$ is given by $[x^0, '00', '00', '00'] = ['01', '00', '00', '00']$. The fifth column of W , W_4 , becomes:

$$W_4 = \begin{bmatrix} 2B \\ 7E \\ 15 \\ 16 \end{bmatrix} \oplus \begin{bmatrix} 8A \\ 84 \\ EB \\ 01 \end{bmatrix} \oplus \begin{bmatrix} 01 \\ 00 \\ 00 \\ 00 \end{bmatrix} = \begin{bmatrix} A0 \\ FA \\ FE \\ 17 \end{bmatrix}$$

This process of generating W_i must be repeated until W contains 44 columns. Then W will look like (not all columns are given):

$$W = \begin{bmatrix} 2B & 28 & AB & 09 & A0 & 88 & 23 & 2A & F2 & 7A & 59 & 73 & 3D & \dots & B6 \\ 7E & AE & F7 & CF & FA & 54 & A3 & 6C & C2 & 96 & 35 & 59 & 80 & \dots & 63 \\ 15 & D2 & 15 & 4F & FE & 2C & 39 & 76 & 95 & B9 & 80 & F6 & 47 & \dots & 0C \\ 16 & A6 & 88 & 3C & 17 & B1 & 39 & 05 & F2 & 43 & 7A & 7F & 7D & \dots & A6 \end{bmatrix}$$

9. Decryption

In the DES algorithm encipherment and decipherment consists of the same operations, only the order of the subkeys is different. In Rijndael, this is not the case. Each operation that is used for encryption must be inverted to make it possible to decrypt a message. In figure 7 the order of these operations are shown.

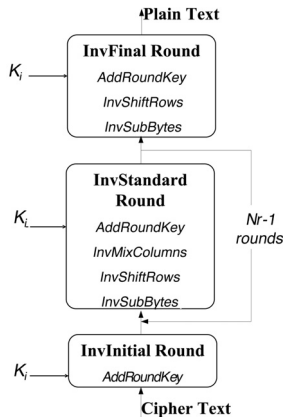


Figure 7: Decryption.

Here we give a short description of each of these inverse operation.

InvSubBytes InvSubBytes is a similar operation as the SubBytes operation, only the inverse of the S-box used for encryption is used, see table 3.

InvShiftRows The InvShiftRows operation is equal to the ShiftRows operation, only the shift is to the right instead of to the left.

Table 3: The Inverse S-box.

								y									
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
x	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

InvMixColumns To invert the MixColumns operation, the matrix used in Mixcolumns must be inverted. The InvMixColumns operation then becomes:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (10)$$

Key schedule for decryption For decryption the key schedule is a little different than the one for encryption. The following steps must be taken to expand the key for decryption:

1. Apply the Key Schedule scheme for encryption.
2. Apply InvMixColumns to all RoundKeys except the first and the last one.

10. Some words on the security of Rijndael

The structure of Rijndael differs very much from the structure of DES. During each round in DES only half of the bits are used and therefore changed. In Rijndael all the bits are used in each round, which causes a high diffusion in a small number or

rounds. For Rijndael it can be shown that full diffusion is already achieved after two rounds, e.g. each of the 128 bits after two rounds depends on all of the 128 input bits.

The SubBytes transformation is the one that causes the cipher to be non-linear. The S-box is constructed by using the non-linear transformation x^{-1} , e.g. computing the inverse, in $GF(2^8)$. This simple expression allows algebraic manipulations that can be used to mount attacks such as interpolation. Therefore, to the mapping an affine transformation is added. The entire construction of the S-box is also kept simple and explicit, to avoid any suspicions of trapdoors built into the algorithm, as has always been the case for DES.

The ShiftRows operation is added after two attacks (e.g. truncated differentials and the Square attack) were developed against the predecessor of Rijndael, Square. This ShiftRows operation makes the Rijndael algorithm resistant against these attacks.

The MixColumn transformation causes diffusion among the bytes. Changing one input byte in the MixColumn operation results in a change of all four output bytes.

In the Key schedule the S-box is used which causes a non-linear mixing of the keys. The key schedule is designed such that it is resistant against attacks where the cryptanalyst knows part of the key and tries to obtain the remaining bits. A second important design criterium is that there should not be two different cipher keys that have a large set of RoundKeys in common. Each round is different due to the usage of the round constants. This eliminates symmetries in the encryption process.

Over the years it has been shown that there are attacks for Rijndael available that can attack the cipher faster than brute force up to six rounds. Therefore it is chosen that the cipher should have at least 10 rounds. In the future it is possible to extend the number of rounds.

11. References

The following references were used to write this handout.

- J. Daemen and V. Rijmen. *AES Proposal: Rijndael*. 1999.
- W. Trappe and L.C. Washington. *Introduction to cryptography with coding theory*. Prentice Hall, 2002.
- A.J. Menezes, P.C. van Oorschot and S.A. Vanstone *Handbook of applied cryptography*. CRC press, 1996.
- National Institute of Standards and Technology. "Announcing the Advanced Encryption Standard (AES)", November 2001, <http://csrc.nist.gov/CryptoToolkit/aes/>