Chapter 6

AES

The cipher AES ("Advanced Encryption Standard") is the successor of the obsolete DES. It was adopted after a thorough competitive selection procedure in 2001. The winner of the competiton was the Belgian algorithm Rijndael, henceforth called AES, sparing English speaking people the plight of correct pronunciation, and neglecting a small difference in the specifications: Rijndael contains some extended parameter options that are not standardized for AES.

AES is a multiple cipher with several rounds but not a FEISTEL cipher, not even an SP-network in the proper sense. The kernel map is based on an S-box that essentially is the multiplicative inversion in the finite field $\mathbb{F}_{256}$. For a comprehensive analysis of the nonlinear properties of this S-box see Appendix D.

The inventors Joan DAEMEN and Vincent RIJMEN themselves published a book that provides a very comprehensive und comprehensible description of the method:

> Joan Daemen, Vincent Rijmen, The Design of Rijndael. AES – The Advanced Encryption Standard. Springer-Verlag, Berlin 2002. ISBN 3-540-42580-2.

(Note that Joan is a Flemish version of John.)

In this text we only give an introduction into the overall scheme and the kernel map.

6.1 The Structure of AES

Overview

Figures 6.1 and 6.2 give an overall impression of the scheme of AES and illustrate how the construction principles derived in former sections show up in AES.

- The block length is $n = 128$, the key length, $l = 128$, 192, or 256, the number of rounds, $r = 10$, 12 oder 14.
- As a first step of each round, and as a last step after the last round, a partial key is added to the current bitblock, making a total of $r + 1$ partial keys.
- The 128-bit "partial keys" $k^{(i)}$ are not partial keys in the proper sense but are extracted from the real key k by a somewhat involved procedure called "key expansion". In particular they are not stochastically independent.
- At the begin of each round, after adding the round key, the current 128-bit block is decomposed into 16 partial blocks each with 8 bits. The S-box $S : \mathbb{F}_2^8 \longrightarrow \mathbb{F}_2^8$ is applied to each of these 16 blocks. The linear potential of the S-box is $\frac{1}{64}$, see Appendix D.
- The "diffusion" step consists of a permutation followed by a linear map. This step is slightly more complex than the standard for a pure SP-network as in Section 2.4.

A further remark on the key expansion: The selection of the "round keys" $k^{(i)}$ conceal the "real" key. Cryptanalytic approaches attack the "effective" key consisting of the collection of the round keys $k^{(i)}$. This is adequate for breaking the cipher—the real key is not needed. However the complexity of the key expansion might prevent exploiting a dependency of the different round keys, for instance in the case where the round keys are overlapping partial blocks of the "real" key.

Representation of the Bitblocks

AES operates on octets (8-bit bytes), that is, on the $\mathbb{F}_2$ vector space $\mathbb{F}_2^8$. Since the structure of this vector space as a field with 256 elements plays a crucial role in several steps of the algorithm identifying this vector space with the field $\mathbb{F}_{256}$ suggests itself. The exact identification map is given in Section 6.2.

AES operates on bitblocks of lengths that are multiples of 32. Plaintexts, ciphertexts, an intermediate results have 128 bits, the key length is 128, 192, or 256.

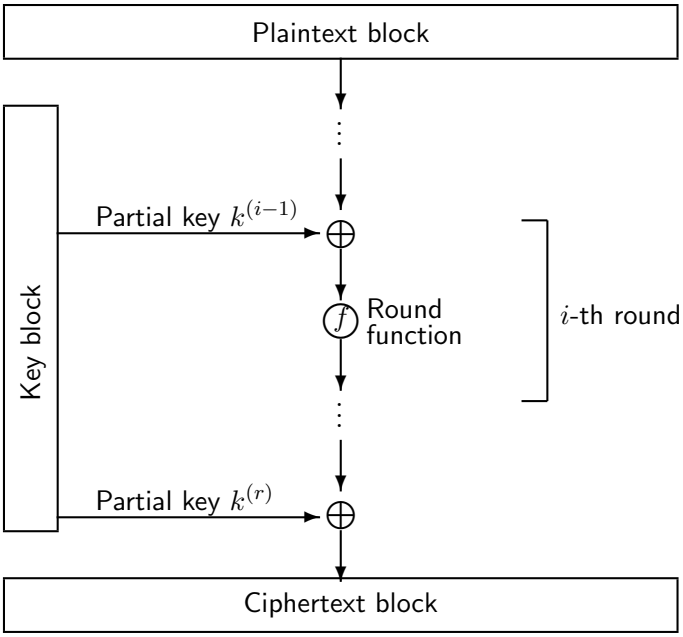


Figure 6.1: The overall scheme of AES

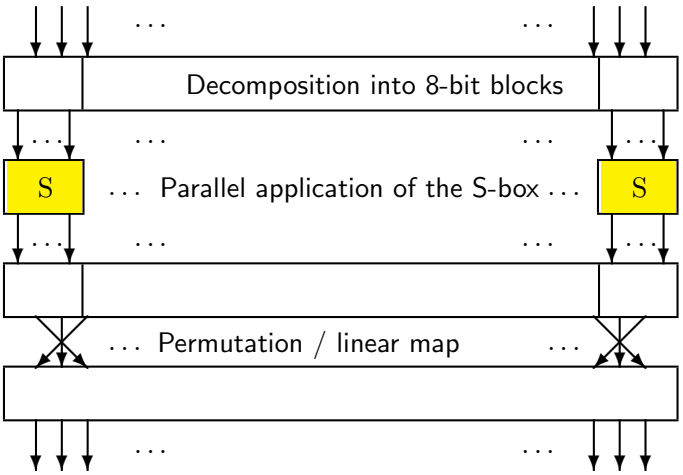


Figure 6.2: The round function f of AES

Remark The only difference between Rijndael and AES is that the specification of Rijndael allows block and key lengths of 128, 160, 192, 224, 256 each. The AES standardization procedure makes no assertions about the security of Rijndael for the additional sizes of these parameters.

Interpret the 32-bit blocks as columns with 4 octets, that is, as elements of the 4-dimensional $\mathbb{F}_{256}$ vector space $\mathbb{F}_{256}^4$. So each block of length $n = 32 \cdot N_b$ is an N_b-tuple of such columns, or a $4 \times N_b$-matrix

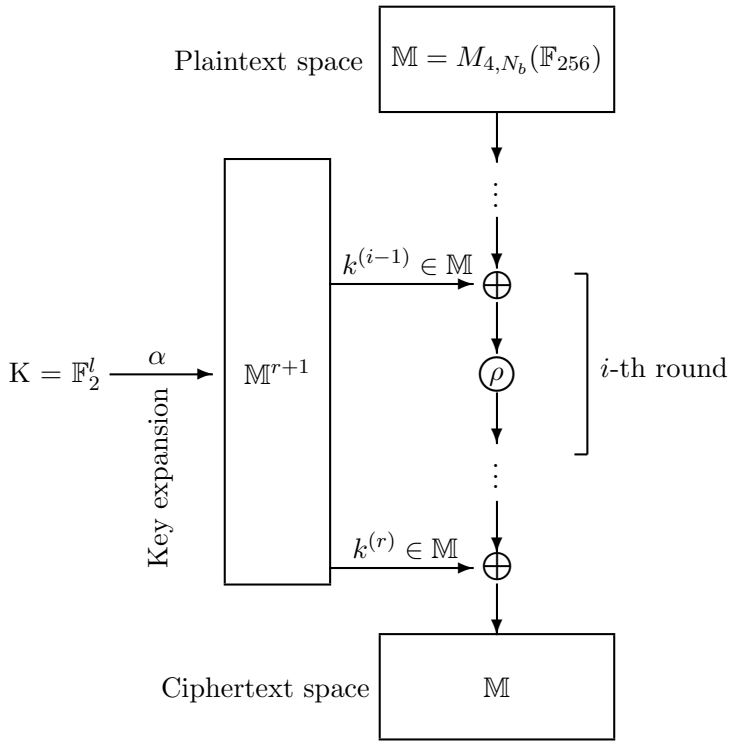
$$a = \begin{pmatrix} a_{0,0} & \dots & a_{0,N_b-1} \\ \vdots & & \vdots \\ a_{3,0} & \dots & a_{3,N_b-1} \end{pmatrix} \in M_{4,N_b}(\mathbb{F}_{256}) =: \mathbb{M}.$$

Analogously let the key length be $l = 32 \cdot N_k$; however we don't use the corresponding matrices but expand the keys and then decompose them into round keys $k^{(i)}$.

> The specification of Rijndael uses $N_b, N_k \in \{4, 5, 6, 7, 8\}$, the specification of AES, $N_b = 4$, $N_k \in \{4, 6, 8\}$.

The Iteration Scheme

An AES encryption runs through r rounds. Each round consists of the (binary) addition of a round key and the kernel map ρ. In the last round the kernel map is slightly abbreviated—more on this later on. After the last round one more partial key is added. Therefore the key expansion has to produce $r + 1$ partial keys, each consisting of $32N_b$ bits, from the total of l key bits. The kernel map is invertible, and is the same for all rounds except the last one.



The number of rounds for Rijndael is specified as follows:

	N_b				
N_k	4	5	6	7	8
4	10	11	12	13	14
5	11	11	12	13	14
6	12	12	12	13	14
7	13	13	13	13	14
8	14	14	14	14	14

For AES with key lengths 128, 192, or 256 the number of rounds is 10, 12, or 14, tagged green in the table.

The Kernel Map

Each round of AES (or Rijndael) consists of four steps

1. `AddRoundKey`, the addition of the round key, see Figure 6.3.
2. `SubBytes`, a substitution that consists of parallel application of the S-box on each octet, see Figure 6.4,
3. `ShiftRows`, a permutation of each matrix row, see Figure 6.5,

4. `MixColums`, a linear map of each matrix column to itself, see Figure 6.6,

The kernel map ρ consists of steps 2 to 4. In the last round step 4, `MixColums`, is omitted. In this way the inverse map—decryption—has the same structure. The cryptographic strength is not affected by this omission.

Figures 6.3 to 6.6 are taken from the Wikipedia article on AES (for the case $N_b = N_k = 4$).

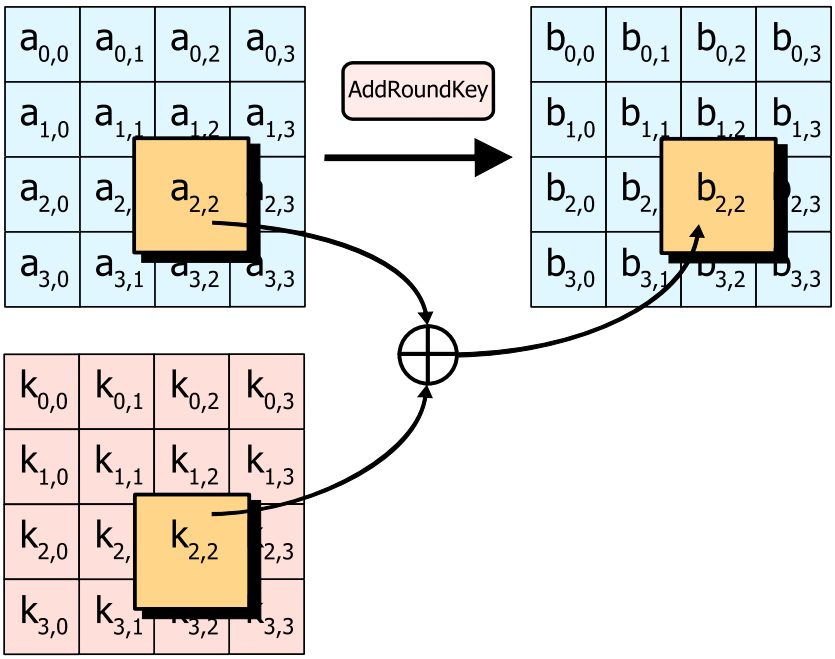


Figure 6.3: The operation of `AddRoundKey`

The Substitution Step `SubBytes`

Apply the S-box S_{RD} separately in parallel to all octets of the current state, that is, to all entries of the current matrix $b \in \mathbb{M}$. This S-box is the composition $S_{RD} = f \circ g$ of two maps, the inversion g in $\mathbb{F}_{256}$,

$$g\colon \mathbb{F}_{256} \longrightarrow \mathbb{F}_{256}, \quad g(x) = \begin{cases} x^{-1} & \text{for } x \neq 0, \\ 0 & \text{for } x = 0, \end{cases}$$

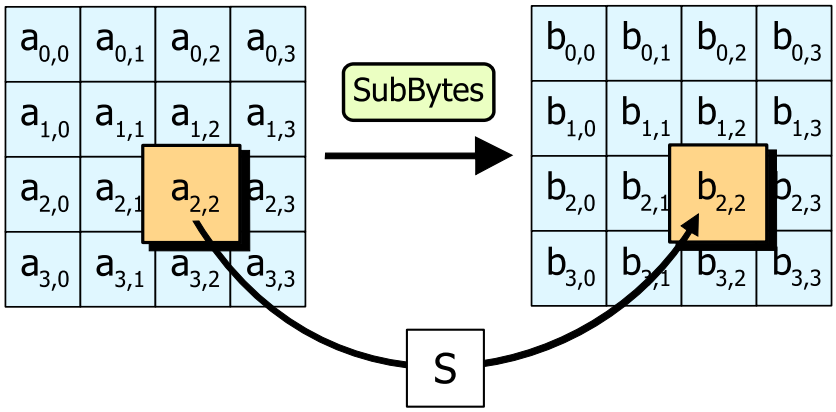


Figure 6.4: The operation of `SubBytes`

(whose nonlinearity properties we know already well), and the affine map

$$f\colon \mathbb{F}_2^8 \longrightarrow \mathbb{F}_2^8,\quad \begin{pmatrix} x_7\\ x_6\\ x_5\\ x_4\\ x_3\\ x_2\\ x_1\\ x_0 \end{pmatrix} \mapsto \begin{pmatrix} 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1\\ 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1 \end{pmatrix} \begin{pmatrix} x_7\\ x_6\\ x_5\\ x_4\\ x_3\\ x_2\\ x_1\\ x_0 \end{pmatrix} + \begin{pmatrix} 0\\ 1\\ 1\\ 0\\ 0\\ 0\\ 1\\ 1 \end{pmatrix}.$$

This construction ensures that

- S_{RD} has no fixed point, that is, $\mathrm{S}_{\mathrm{RD}}(b) \neq b$ for all $b \in \mathbb{F}_{256}$,
- S_{RD} has no "anti fixed point", that is, $\mathrm{S}_{\mathrm{RD}}(b) \neq \bar{b}$ for all $b \in \mathbb{F}_{256}$;

where $\bar{b} = (1-b_7, \ldots, 1-b_0)$ is the Boolean complement, that is, the bitwise logical negation.

The drawback of this modification of the inversion map is that the involutory property of g gets lost; therefore the decryption algorithm needs the implementation of another S-box $\mathrm{S}_{\mathrm{RD}}^{-1}$.

The Row Permutation `ShiftRows`

Each of the four rows of the current state—a $4 \times N_b$-matrix—gets cyclically shifted to the left by an individual amount; row i by C_i positions. The C_i depend on the block length in the following way:

N_b	C_0	C_1	C_2	C_3
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

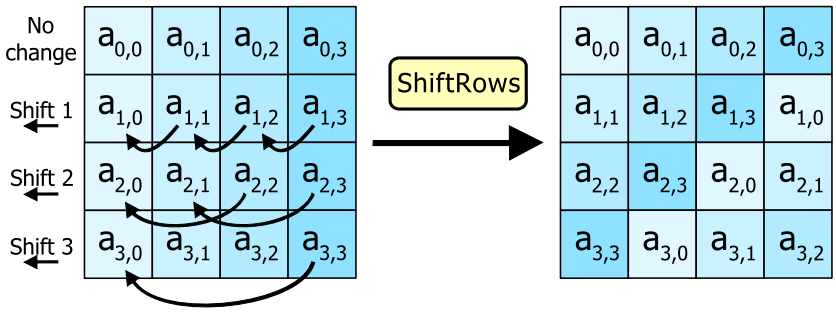


Figure 6.5: The operation of `ShiftRows`

The Linear Transformation of the Columns, `MixColumns`

The current state is multipled with a fixed 4×4-matrix column by column from the left; in other words, the state matrix $\in \mathbb{M}$ is multiplied from the left to generate some diffusion. This is the $\mathbb{F}_{256}$-linear map

$$\mu \colon \mathbb{F}_{256}^4 \longrightarrow \mathbb{F}_{256}^4,$$

specified by the matrix

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

whose entries must be interpreted as octets in hexadecimal representation, for example $03 = (00000011) \in \mathbb{F}_2^8$.

Decryption

The complete AES (or Rijndael) encryption is the composition

$$F_k = \kappa_r \circ \underbrace{\tau \circ \sigma \circ \kappa_{r-1}}_{i=r} \circ \ldots \circ \underbrace{[\mu \circ \tau \circ \sigma \circ \kappa_{i-1}]}_{i=1,\ldots,r-1} \circ \ldots \circ \kappa_0$$

of maps $\mathbb{M} \longrightarrow \mathbb{M}$. Here κ_i is the addition of the i-th round key, $\tau =$ `ShiftRows`, $\sigma =$ `SubBytes`, and $\mu =$ `MixColumns`, and the kernel map is $\rho = \mu \circ \tau \circ \sigma$.

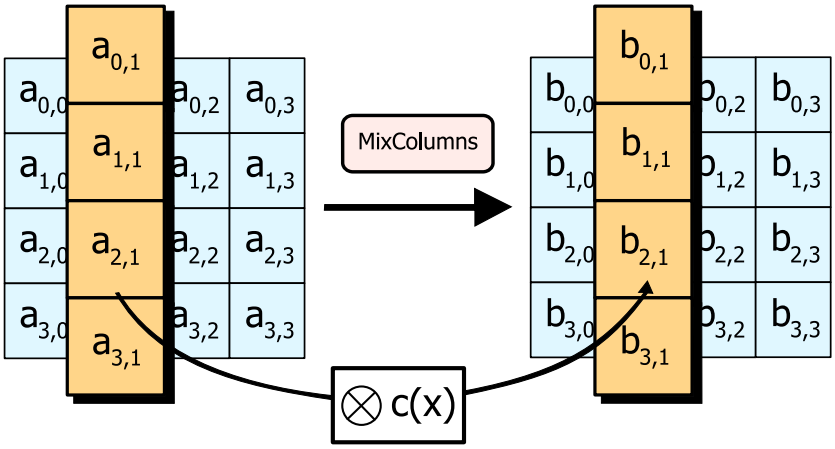


Figure 6.6: The operation of `MixColumns`

For decryption we need the inverse map that looks as follows:

$$F_k^{-1} = \kappa_0^{-1} \circ \ldots \circ [\kappa_{i-1}^{-1} \circ \sigma^{-1} \circ \tau^{-1} \circ \mu^{-1}] \circ \ldots \circ \kappa_{r-1}^{-1} \circ \sigma^{-1} \circ \tau^{-1} \circ \kappa_r^{-1}.$$

The promised structural analogy with F_k follows after some transformations:

- We have $\kappa_i^{-1} = \kappa_i$ for all i.
- Because σ acts on the matrix entries, in the same way on each entry, we have $\tau \circ \sigma = \sigma \circ \tau$.
- Finally $\kappa_i \circ \mu(x) = \mu(x) + k^{(i)} = \mu(x + \mu^{-1}(k^{(i)})) = \mu \circ \tilde{\kappa}_i(x)$ since μ is linear. Hence $\mu^{-1} \circ \kappa_i = \tilde{\kappa}_i \circ \mu^{-1}$, where $\tilde{\kappa}_i$ is the binary addition of $\mu^{-1}(k^{(i)})$. Use this for $i = 1, \ldots, r$.

This gives

$$F_k^{-1} = \kappa_0 \circ \tau^{-1} \circ \sigma^{-1} \circ \tilde{\kappa}_1 \circ \ldots \circ [\mu^{-1} \circ \tau^{-1} \circ \sigma^{-1} \circ \tilde{\kappa}_i] \circ \ldots \circ \kappa_r.$$

Hence the decryption algorithm is composed in the same way as the encryption algorithm with the following modifications:

- a modified key expansion: $\tilde{\kappa}_i$ instead of κ_i,
- the reverse order of the partial keys,
- `MixColumns`: μ replaced by the inverse linear map μ^{-1},
- `Shiftrows`: τ replaced by a right shift,
- S-box S_{RD} replaced by the inverse map $\mathrm{S}_{\mathrm{RD}}^{-1}$.

Key Expansion

We wont describe the key expansion in detail, but only mention that it involves cyclic shifts of bytes inside blocks of lengths 4, the S-box S_{RD}, and the addition of fixed constants.

6.2 The Arithmetic of the Base Field

For the description of AES we identify the 8-dimensional $\mathbb{F}_2$ vector space $\mathbb{F}_2^8$ and the field $\mathbb{F}_{256}$. We specify the exact mapping in the following subsections.

Algebraic Representation of the Base Field

The simplest construction of a finite field, see Appendix A, is as a factor ring of the polynomial ring $\mathbb{F}_p[X]$ over its prime field $\mathbb{F}_p$ by a principal ideal that is generated by an irreducible polynomial $h \in \mathbb{F}_p[X]$. The ideal $h\mathbb{F}_p[X]$ is prime, hence

$$K := \mathbb{F}_p[X]/h\mathbb{F}_p[X]$$

is a finite field and has degree (= dimension) $n = \deg h$ over $\mathbb{F}_p$. For the identification of K with the vector space $\mathbb{F}_p^n$ we identify the residue classes of the powers of X with the n unit vectors. So setting $x = X \bmod h$ we identify:

$$x^0 = 1 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad x^1 = x = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix}, \quad \ldots, \quad x^{n-1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

If $h = X^n + a_1X^{n-1} + \cdots + a_{n-1}X + a_n$ (monic without loss of generality), then from $h \bmod h = 0$ we get

$$x^n = -a_1x^{n-1} - \cdots - a_{n-1}x - a_n$$

in K. Moreover this equation shows how to express the residue class of an arbitrary polynomial f by the canonical basis $1, x, \ldots, x^{n-1}$. Algorithmically this amounts to the remainder of a polynomial division "f divided by h".

For AES we use the polynomial

$$h = X^8 + X^4 + X^3 + X + 1 \in \mathbb{F}_2[X].$$

Multiplication Table

The multiplication table for the basis $(1, x, \ldots, x^{n-1})$ follows from the relation defined by h. In $\mathbb{F}_{256}$ (for AES) we have

$$x^2 \cdot x^7 = x^9 = x \cdot x^8 = x \cdot (x^4 + x^3 + x + 1) = x^5 + x^4 + x^2 + x.$$

Efficient inversion

The implementation of AES uses a complete value table of the S-box. This is efficient for we have to specify only 256 values.