B.d) AES

B.96 AES (Advanced Encryption Standard)

AES is a symmetric block cipher with

- plaintext space P = ciphertext space $C = \{0,1\}^{128}$
- key space
  - $K = \{0,1\}^{128}$ (usual case) or
  - $K = \{0,1\}^{192}$ or
  - $K = \{0,1\}^{256}$
- Depending on the size of K the AES is a round-based block cipher with (cf. B.99)
  - 10 rounds or
  - 12 rounds or
  - 14 rounds
- AES is not a Feistel cipher.

B.97 AES (History)

- In 1997 NIST (National Institute for Standards and Technology) initiated a competition to find a successor of DES.
- Requirements
  - Security, especially resistance against linear and differential attacks
  - Efficiency (hardware and software implementations)
  - Scalability
  - Royalty freeness

B.97 AES (History)

- 1st Round (1998):
  - 15 algorithms were submitted
  - main aspect: security
  - 5 algorithms “survived” the first round
- 2nd Round
  - Main aspect: Efficiency on various platforms
- Winner of the competition: Rijndael (designers: V. Rijmen, J. Daemen,)

B.98 Remark

Note: Cryptanalysts from all over the world analyzed the submitted AES candidates. Security and implementation aspects were discussed on many crypto conferences.

B.99 Scalability

- The AES consists of Nr rounds and uses a 32*Nk bit key
- Admissible pairs: (Nr, Nk) =
  - (10,4) (usual case)
  - (12,6)
  - (14,8)

Note: Rijndael additionally considered the cases $P = C = \{0,1\}^{192}$ and $P = C = \{0,1\}^{256}$. These options have not been standardized.

B.100 State Space

- plaintext block: $(s_{00}, s_{10}, s_{20}, s_{30}, s_{01}, s_{11}, \ldots, s_{33}) \in (\{0,1\}^8)^{16} \cong \{0,1\}^{128}$. (The s_{ij} denote bytes.)
- The plaintext block is transformed into the state

<u>state</u>

$$\begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix}$$

B.100 (continued)

- The plaintext bytes fill the state array, column by column (direction: top - down), beginning with the leftmost column.
- After encryption the (final) state is transformed into a ciphertext block.

<u>Decryption:</u> ciphertext block $\rightarrow$ state $\rightarrow$ plaintext block

B.101 AES (coarse structure)

plaintext block (128 bit = 16 Byte) $\rightarrow$ state

AddRoundKey(state,RoundKey_0*) [[* non-standard notation]]

```
For i =1 to Nr-1 do {
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, RoundKey_i*)
}
SubBytes(state)                          ⎫
ShiftRows(state)                         ⎬ final round
AddRoundKey(state, RoundKey_Nr*)         ⎭
```

state $\rightarrow$ ciphertext block

B.102 Remark

(i) The AES cipher consists of four ‘basic’ transformations. These transformations operate on the state.

(ii) The final round is different from the others. (The MixColumns(.) operation is missing.)

(iii) AES is a byte-oriented cipher. Each state byte s_{ij} is interpreted as an element in the finite field $GF(2^8)$

B.103 A Reminder: Finite Fields

- For any integer n>1 Z_n :={0,…,n-1} is a ring (equipped with the addition and multiplication modulo n).
- In general Z_n is not a field.
- Example: $2 \in Z_4$ has no multiplicative inverse modulo 4.
- If p is prime Z_p ={0,1,…,p-1} is a field.
- Example: Z_2, Z_{17}, Z_{101} are fields.

Note: The definition of a group, a ring and a field can be found in any elementary algebra book.

B.103 (continued)

Fact:

(i) To any prime p and any positive integer k there exists a finite field with p^k elements.

(ii) All fields with p^k elements are isomorphic.

(iii) Any finite field contains $p'^{k'}$ elements where p' is a prime and k' a positive integer.

Notation: In the following $GF(p^k)$ stands for a finite field with p^k elements. For p prime we alternatively use the notations Z_p and GF(p).

B.103 (continued)

- GF(2)[X] denotes the ring of polynomials over GF(2).
- Example: X^4+1, $X^2+X \in$ GF(2)[X]
- A polynomial p(X) with deg(p(X)) $\geq$ 1 is called irreducible in GF(2)[X] if it cannot be expressed as a product of two non-constant polynomials.

Example:

(i) $X^2+X = X (X + 1)$ is not irreducible in GF(2)[X]

(ii) X^2+X+1 is irreducible in GF(2)[X]

B.103 (continued)

- The AES cipher considers the polynomial

  $m(X) := X^8 + X^4 + X^3 + X + 1 \in GF(2)[X]$

  This polynomial is irreducible in GF(2)[X].
- $< m(X) > := \{ p(X)m(X) | \; p(X) \in GF(2)[X] \}$
- Fact: The factor ring $GF(2)[X] / < m(X) >$ is a field. More precisely, it is (isomorphic to) $GF(2^8)$. That is,

  $GF(2^8) \cong \{ p(X) + < m(X) > \; | \; p(X) \in GF(2)[X] \}$.

B.103 (continued)

Reminder: For concrete computations modulo n we use the set of representatives $Z_n = \{0,1,\ldots,n-1\}$.

Similarly, for computations in $GF(2^8)$ we use the set of representatives

$R:=\{p(X) \in GF(2)[X] \mid \deg(p(X)) < \deg(m(X))=8\}$
Polynomials are added and multiplied modulo m(X).

A more detailed treatment: blackboard

B.104 Example

- $X^8 \equiv X^4 + X^3 + X + 1 \pmod{m(X)}$
- Let $a := X^6+X^4+X^1+1$ and $b := X^2+X^1+1$
- Then $a+b = X^6+X^4+X^1+1+X^2+X^1+1 = X^6+X^4+X^2$

  (The corresponding coefficients are added modulo 2.)
- $a*b = (X^6+X^4+X^1+1)(X^2+X^1+1)$

  $= (X^8+X^6+X^5+X^2) + (X^7+X^5+X^2+X^1) + (X^6+X^4+X^1+1)$

  $= X^8+X^7+X^4+1 \equiv X^4+X^3+X+1 + X^7+X^4+1$

  $= X^7 + X^3 + X \pmod{m(X)}$

B.105 Miscellaneous

- We identify a byte **b** = ($b_7, b_6, \ldots, b_0$) with the polynomial $b_7X^7 + b_6X^6 + \ldots + b_0$
- Bytes are added and multiplied according to the laws in the field $GF(2^8)$.
- In hexadecimal notation the byte ($b_7, b_6, \ldots, b_0$) reads ($8{*}b_7 + 4{*}b_6 + 2{*}b_5 + b_4$, $8{*}b_3 + 4{*}b_2 + 2{*}b_1 + b_0$).
- <u>Example:</u> In hexadecimal notation (11010011) reads D3.

B.106 Next Steps

Study the basic transformations

- SubBytes(state)
- ShiftRows(state)
- MixColumns(state)
- AddRoundKey(state, RoundKey)

B.107 SubBytes

- *SubBytes(.)* maps an element $t \in GF(2^8)$ to $S(t)$ where $S: GF(2^8) \rightarrow GF(2^8)$ denotes a fixed non-GF(2)-linear bijective mapping.
- More precisely,

  $S(t)=At^{-1}+c \qquad \text{for } t \neq 0.$

  $S(0)=c$
- In particular,
  - t^{-1} denotes the inverse of t in $GF(2^8)$, viewed as a 8-bit vector
  - A is a fixed (8x8) matrix over GF(2)
  - c is a fixed vector in $GF(2)^8$

B.107 (continued)

$$A := \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \qquad c := \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix}$$

The computation of $At^{-1}+c$ demands an inversion, a matrix-vector multiplication and a vector addition over GF(2).

B.108 Remark

- AES implementations neither invert bytes nor perform matrix-vector multiplication since this was too costly.
- Instead, the values of S are stored, and SubBytes(.) needs only one table-lookup.
- The SubBytes(.) transformation is called S-box.

B.109 ShiftRows

- *The ShiftRows(.)* transformation shifts the rows of the state cyclically to the left. To be precise
  - Row 0 is not shifted
  - Row 1 is shifted cyclically by 1 position to the left
  - Row 2 is shifted cyclically by 2 positions to the left
  - Row 3 is shifted cyclically by 3 positions to the left

B.110 MixColumns

- *MixColumns(state)* is given by a matrix-matrix multiplication in $GF(2^8)$:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 02 & 01 & 01 \end{pmatrix} \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix}$$

Note: The matrix entries 01, 02 and 03 (hexadecimal notation) correspond to the polynomials 1, X and X+1, respectively.

B.111 AddRoundKey

- *AddRoundKey*(state, RoundKey) computes the next state by adding RoundKey (interpreted as a 4x4 matrix over GF(2^8)) to the state.

Note: AddRoundKey(.,.) implies a bitwise XOR addition.

B.112 Key Scheduling

- A non-linear feedback shift register on 32-bit words is used to compute the (Nr+1) round keys from the encryption key *K*.
- Each round key is as large as the state (i.e., it consists of 128 bits.)

B.112 (continued)

Definitions:

- word: $w=(b_0,b_1,b_2,b_3)$ (data type, consists of 4 Bytes)
- $\mathrm{SubWord}(w):=(\mathrm{SubBytes}(b_0), \mathrm{SubBytes}(b_1), \mathrm{SubBytes}(b_2), \mathrm{SubBytes}(b_3))$
- $\mathrm{RotWord}((b_0,b_1,b_2,b_3)):= (b_1,b_2,b_3,b_0)$
- $\mathrm{Rcon}(n)$: $((02)^{n-1},(00),(00),(00))$

  The first byte equals $X^{n-1} \pmod{m(X)} \in GF(2^8)$ (hexadecimal notation).

Note: On the next slide we concentrate on the case Nk=4, i.e. on 128 bit keys. The other key lengths are treated similarly.

B.112 (continued) [128-bit keys]

```
for j:=0 to 3 do w[j] := jth key word
j := 4
while (j < 4 * 11) {
    temp = w[j-1]
    if (j ≡ 0 (mod 4))
        temp = SubWord(RotWord(temp)) ⊕ Rcon(j/4)
    else temp = SubWord(temp)
    w[j] = w[j-4] ⊕ temp
    j := j + 1
}
```

B.112 (continued)

first round key: (w[0], w[1], w[2], w[3])

second round key: (w[4], w[5], w[6], w[7])

...

last round key: (w[40], w[41], w[42], w[43])

Note: When AddRoundKey(.,.) is called the i^{th} time the word w[4*i+j] is added to the j^{th} column of the state.

B.113 Decryption

Decryption:

- The order of the basic transformations has to be reversed.
- Each basic transformation is replaced by its inverse.
- The order of the round keys is reversed.

- AddRoundKey(.,RoundKey) is self-inverse.
- The inverse transformations of SubBytes(.), ShiftRows(.), MixColumns(.) are called InvSubBytes(.), InvShiftRows(.), InvMixColumns(.).

B.113 (continued)

```
ciphertext block (128 bit = 16 Byte) → state
AddRoundKey(state,RoundKey)
For Nr-1 downto 1 do {
    InvShiftRows(state)
    InvSubBytes(state)
    AddRoundKey(state, RoundKey)
    InvMixColumns(state)
}
InvShiftRow(state)
InvSubBytes(state)
AddRoundKey(state, RoundKey)
state → plaintext block
```

B.114 Equivalent Decryption Algorithm

- The transformations SubBytes(.) and ShiftRows(.) commute.
- MixColumns(.) and hence InvMixColumns(.) are linear transformations.
- $\rightarrow$ The inverse operations may be reordered (see next slide).

B.114 (continued)

```
ciphertext block (128 bit = 16 Byte) → state
AddRoundKey(state,RoundKey_(Nr-1))

For i =Nr-1 downto 1 do {
    InvSubBytes(state)
    InvShiftRows(state)
    InvMixColumns(state)
    AddRoundKey(state, InvMixColumn(RoundKey_i)*)
}                                   [[* non-standard notation]]
InvSubBytes(state)
InvShiftRows(state)
AddRoundKey(state, RoundKey_0)
state → ciphertext block
```

B.115 Remark

- A comprehensive justification of the AES design criteria is beyond the scope of this course.
- However, we consider the question what happened if one of these basic transformations would be left out, or equivalently, substituted by the identity mapping at all its occurrences.

B.116 Consequences of Missing AddRoundKey(.,.)

- The ciphertext would not depend on a key. There existed only one encryption transformation.
- Logically, this was equivalent to a key space containing only one key.

B.117 Consequences of Missing MixColumns(.)

- The AES encryption split into four independent ‘small’ encryption algorithms (each affecting one row of the state).
- That is, AES was an encryption algorithm with plaintext and ciphertext block length of 32 bits. We already know that this is too small.

B.118 Consequences of Missing ShiftRows(.)

- The AES encryption split into four independent ‘small’ encryption algorithms (each affecting one column of the state).
- That is, AES was an encryption algorithm with plaintext and ciphertext block length of 32 bits. We already know that this is too small.

B.119 Consequences of Missing SubBytes(.)

- The transformations ShiftRows(.), MixColumns(.): $GF(2^8)^{16} \rightarrow GF(2^8)^{16}$ are $GF(2^8)$-linear
- The transformation AddRoundKey(.,.) adds the round key to the state.
- Under this condition the key scheduling transformation: $GF(2^8)^{16} \rightarrow GF(2^8)^{16*11}$ was $GF(2^8)$-affine.

<u>Conclusion:</u> If the S-box transformation was replaced by the identity mapping it was very easy to recover an AES key by a known-plaintext attack (why?). In fact it required no more than elementary linear algebra.

B.120 Vulnerability against Attacks

- The resistance against known types of attacks (e.g. linear and differential attacks) was one important design criterion in the AES competition.
- Compared to DES, for instance, the AES has a rich algebraic structure. In 2002 and 2003 several algebraic attacks were proposed. It was suggested to consider specific systems of non-linear equations over GF(2) or GF(2^8), resp., whose solution recover an AES key.

B.121 (continued)

- Some researchers predicted that these attacks were considerably more efficient than exhaustive key search. This lowered the confidence in the AES in the public.
- These conjectures could not be confirmed in the following years. Some assertions were shown to be definitely false.

B.122 Background

- Any mapping f: $\{0,1\}^n \rightarrow \{0,1\}$ can be expressed by a polynomial over GF(2).

<u>Example:</u>

(i) n=2, f(0,0) = f(0,1) = f(1,0) = 0, f(1,1) = 1

Then $f(x_1,x_2) := x_1 x_2$

(ii) n=2, f(0,0) = f(1,0) = 1, f(0,1) = f(1,1) = 0

Then $f(x_1,x_2) := 1 - x_2$

B.123 Impact on Block Ciphers

- Principally, each block cipher Enc $:\{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ could be represented by n nonlinear polynomials in n+m binary variables. (Each variable corresponds to a particular plaintext, resp. to a particular key bit.)
- With this polynomial representation a known plaintext attack is equivalent to solving a system of non-linear equations (unknown key k).

B.123 (continued)

Problem: For reasonable parameters n and m the polynomials consist of a gigantic number of monomials.

Note: The number of monomials can be reduced by introducing additional variables and formulating additional equations.

- Due to its algebraic structure for the AES cipher systems of non-linear equations with a moderate number of terms were found.

B.123 (continued)

- In 2002 Murphy & Robshaw worked out a system of 4800 quadratic and 3008 linear equations over $GF(2^8)$ in 4608 variables. A part of its solution gives the AES key.
- Unlike for linear equations solving systems of non-linear equations over finite fields is difficult. No universal algorithm is known that works efficiently for arbitrary systems of non-linear equations.

Note: Until now algebraic attacks did not yield more efficient attacks than exhaustive key search.

B.124 Remark

For more sophisticated security analysis and (further) attacks on the AES (resp. on the AES with a reduced number of rounds) the interested reader is referred to the literature.