

03 - AES

Kryptologie LAB

Luc Spachmann

Friedrich-Schiller-Universität Jena

12. November 2021

- Blockchiffre mit 128 Bit Blöcken
- Schlüssel auch 128, 192, 256 Bit
- Wir betrachten nur 128 Bit Schlüssel
- Aus Schlüssel werden 11 Rundenschlüssel erzeugt

Algorithmus (Verschlüsselung)

Require: 128 Bit Klartextblock

Require: 11 Rundenschlüssel K

```
1: AddRoundKey( $K[0]$ )
2: for  $i=1\ldots 9$  do
3:   SubBytes
4:   ShiftRows
5:   MixColumns
6:   AddRoundKey( $K[i]$ )
7: end for
8: SubBytes
9: ShiftRows
10: AddRoundKey( $K[10]$ )
```

Initialisierung

- Block wird gespeichert als 4×4 Byte Matrix
- Initialisierung erst über Spalten dann Zeilen
- Entschlüsselung umgekehrte Reihenfolge (abgeänderte Funktionen)
- Schlüsseladdition über XOR

x_0	x_4	x_8	x_{12}
x_1	x_5	x_9	x_{13}
x_2	x_6	x_{10}	x_{14}
x_3	x_7	x_{11}	x_{15}

- Lokale Substitution auf jedem Byte
- Substitutionen gegeben durch Tabelle (moodle)
- Byte als Zahl interpretieren
- Verschiedene S Boxen für Ver- und Entschlüsselung

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	\Rightarrow	$s(a_{0,0})$	$s(a_{0,1})$	$s(a_{0,2})$	$s(a_{0,3})$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$		$s(a_{1,0})$	$s(a_{1,1})$	$s(a_{1,2})$	$s(a_{1,3})$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$		$s(a_{2,0})$	$s(a_{2,1})$	$s(a_{2,2})$	$s(a_{2,3})$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$		$s(a_{3,0})$	$s(a_{3,1})$	$s(a_{3,2})$	$s(a_{3,3})$

- Zyklische Permutation jeder Zeile nach links
- Zeile i wird um i verschoben (Zählung 0-beginnend)
- Für Entschlüsselung Verschiebung nach rechts

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

 \Rightarrow

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

MixColumns

- Spaltenweise Substitution
- Wird als Matrixmultiplikation im Galoiskörper $GF(2^8)$ beschrieben

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

- Zur Entschlüsselung:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \rightarrow \begin{pmatrix} E & B & D & 9 \\ 9 & E & B & D \\ D & 9 & E & B \\ B & D & 9 & E \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Rechenregeln im Galoiskörper $GF(2^8)$

- Formal werden Bytes als Polynome 7. Grades aufgefasst
- Koeffizienten aus \mathbb{Z}_2 (entsprechen Bits)
- $(a_7, \dots, a_0) \hat{=} a_7x^7 + \dots + a_0$
- Addition entspricht einem Bitweisen XOR
- Multiplikation ist die Polynommultiplikation modulo

$$x^8 + x^4 + x^3 + x + 1$$

- Zur Implementierung:
 - Implementierung einer Verdopplungsfunktion (XTime)
 - Multiplikation über wiederholtes Verdoppeln und Addieren (Russische Bauernmultiplikation)

Verdopplungsfunktion im Detail

Require: $a = (a_7, \dots, a_0)$

- 1: $t = a \ll 1$
- 2: **if** $a_7 \neq 0$ **then**
- 3: $t = t \oplus 1b$
- 4: **end if**
- 5: **return** t

- Schreibt ein Programm zum Ver- und Entschlüsseln eines AES-Blocks (also 128 Bit)
- Eingabe:
 - 128 Bit Text
 - 11 Rundenschlüssel
- Ausgabe: Ver-/Entschlüsselter Text.
- **Wichtig:** Schreibt alle Funktionen selber und ohne Bibliotheken!
- Keine Lookup Tables für MixColumns verwenden.