Advanced Encryption Standard (AES)

Dozent: **Prof. Dr. Michael Eichberg**

Version: 2024-02-13

Basierend auf: *Cryptography and Network Security - Principles and Practice, 8th Edition, William Stallings*

Arithmetik endlicher Körper (Rekapitulation)

- Ein Körper ist eine Menge, in der wir Addition, Subtraktion, Multiplikation und Division durchführen können, ohne die Menge zu verlassen.
- Die Division ist mit der folgenden Regel definiert: $a/b = a(b^{-1})$.

Beispiel

Ein endlicher Körper (mit einer endlichen Anzahl von Elementen) ist die Menge Z_p, die aus allen ganzen Zahlen $\{0, 1, \ldots, p-1\}$ besteht, wobei p eine Primzahl ist und in der modulo p gerechnet wird.

Arithmetik endlicher Körper (Rekapitulation)

- For convenience and for implementation efficiency, we would like to work with integers that fit exactly into a given number of bits with no wasted bit patterns.
  - Integers in the range 0 through 2^n–1, which fit into an n-bit word.
- If one of the operations used in the algorithm is division, then we need to work in arithmetic defined over a field.
  - Division requires that each nonzero element has a multiplicative inverse.
- The set of such integers, Z_{2^n}, using modular arithmetic, is not a field!
  - For example, the integer 2 has no multiplicative inverse in Z_{2^n}, that is, there is no integer b, such that $2b \ mod \ 2^n = 1$
- A finite field containing 2^n elements is referred to as $GF(2^n)$.

Every polynomial in $GF(2^n)$ can be represented by an n-bit number.

Finite Field Arithmetic for AES

- In the Advanced Encryption Standard (AES) all operations are performed on 8-bit bytes
- The arithmetic operations of addition, multiplication, and division are performed over the finite field $GF(2^8)$

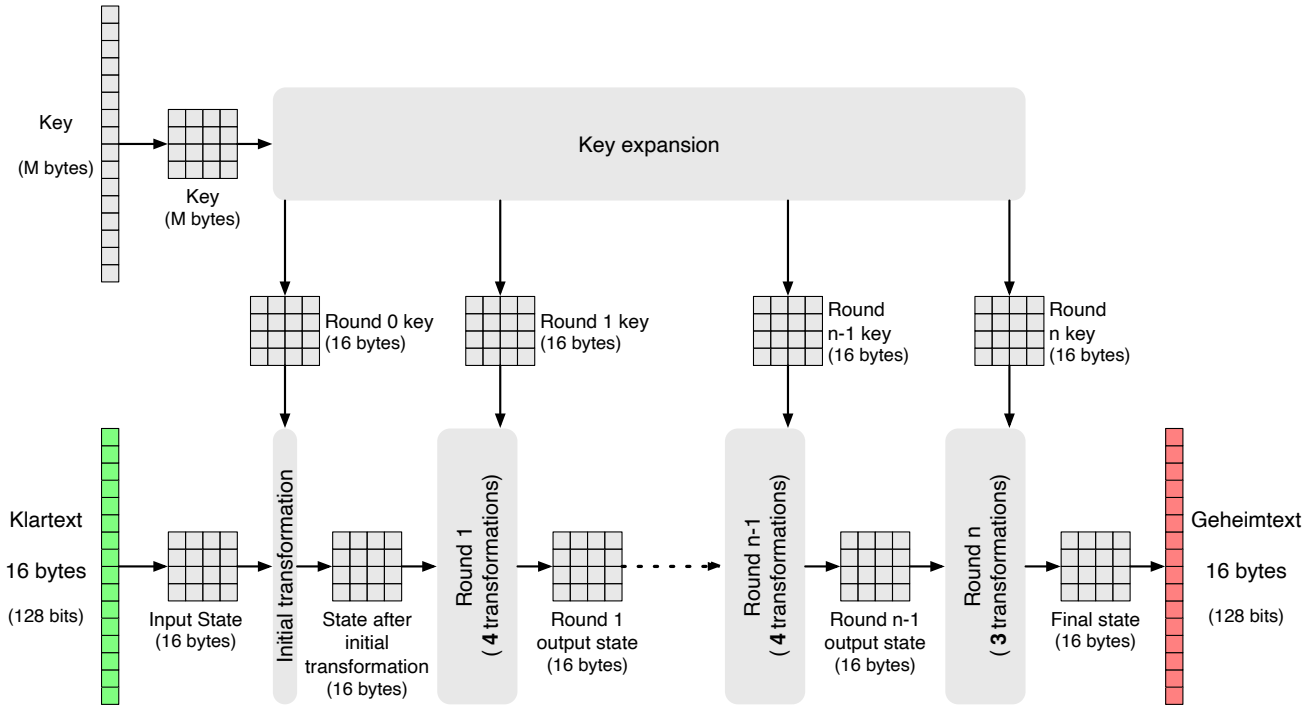
  AES uses the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$

AES Key Elements

- AES uses a fixed block size of 128 bits.
- AES operates on a 4x4 column-major order array of 16 bytes/128 bits: $b_0, b_1, \ldots, b_{15}$ termed the state:

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

AES Encryption Process



AES Parameters

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

AES - Ver- und Entschlüsselungsprozess

(Key Size 128bits)

AES Detailed Structure

- Processes the entire data block as a single matrix during each round using substitutions and permutation.
- The key that is provided as input is expanded into an array of forty-four 32-bit words, $w[i]$ if 128 bits are used for the keysize.
- The cipher begins and ends with an AddRoundKey stage.
- Can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on.
- Each stage is easily reversible.
- The decryption algorithm makes use of the expanded key in reverse order, however the decryption algorithm is not identical to the encryption algorithm.
- State is the same for both encryption and decryption.
- Final round of both encryption and decryption consists of only three stages.

AES Uses Four Different Stages

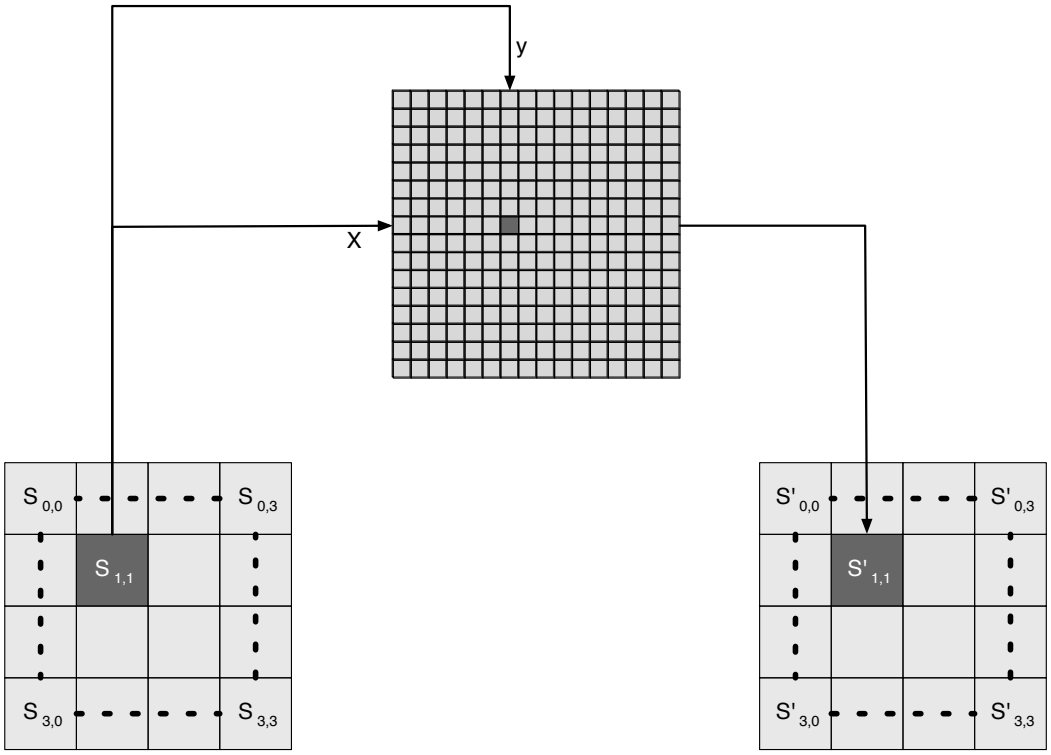
Substitute bytes: uses an S-box to perform a byte-by-byte substitution of the block

ShiftRows: is a simple permutation.

MixColumns: is a substitution that makes use of arithmetic over $GF(2^8)$.

AddRoundKey: is a simple bitwise XOR of the current block with a portion of the expanded key.

AES Substitute byte transformation



AES S-box

x^y	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
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

Jedes einzelne Byte des Zustands (*State*) wird auf folgende Weise auf ein neues Byte abgebildet: Die äußersten linken 4 Bits des Bytes werden als Zeilenwert und die äußersten rechten 4 Bits als Spaltenwert verwendet. Diese beiden Werte dienen als Indizes in der S-Box.

AES Inverse S-box

x^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
7	DO	2C	1E	8F	CA	3F	OF	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	FO	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	FI	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	CO	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	OD	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	BO	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

Beispiel

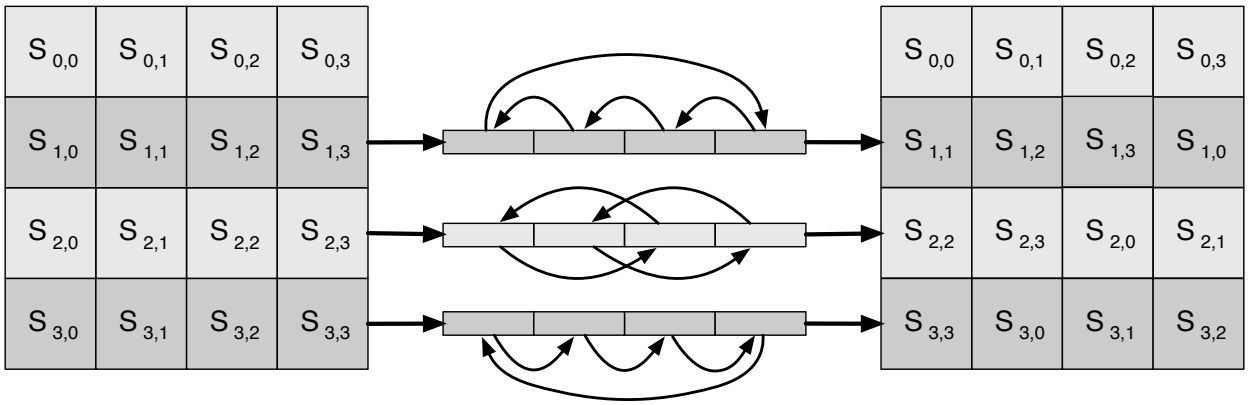
Der (Hex)Wert `0xA3` (`x=A` und `y=3`) wird von der S-Box auf den (Hex)Wert `0x0A` abgebildet.

Die inverse S-Box bildet den Wert `0x0A` (`x=0` und `y=A`) wieder auf den ursprünglichen Wert ab.

S-Box Design Grundlagen

- Die S-Box ist so konzipiert, dass sie gegen bekannte kryptoanalytische Angriffe resistent ist.
- Die Rijndael-Entwickler suchten nach einem Design, das eine geringe Korrelation zwischen Eingabe- und Ausgabebits aufweist und die Eigenschaft hat, dass die Ausgabe keine lineare mathematische Funktion der Eingabe ist.
- Die Nichtlinearität ist auf die Verwendung der multiplikativen Inversen bei der Konstruktion der S-Box zurückzuführen.

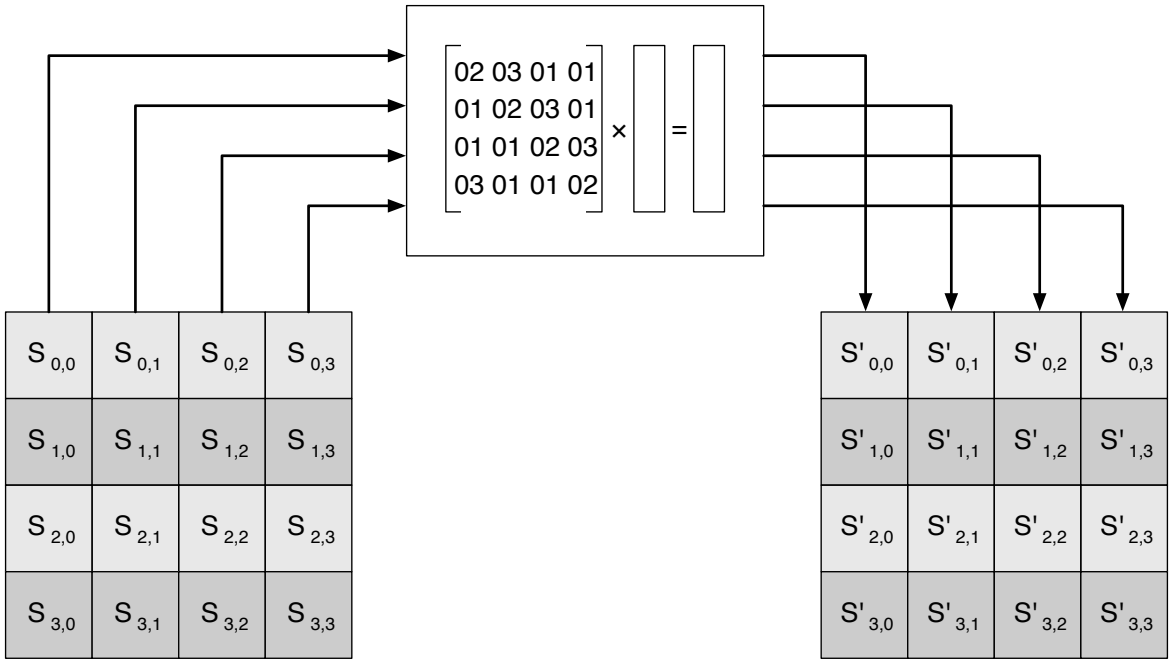
Shift Row Transformation



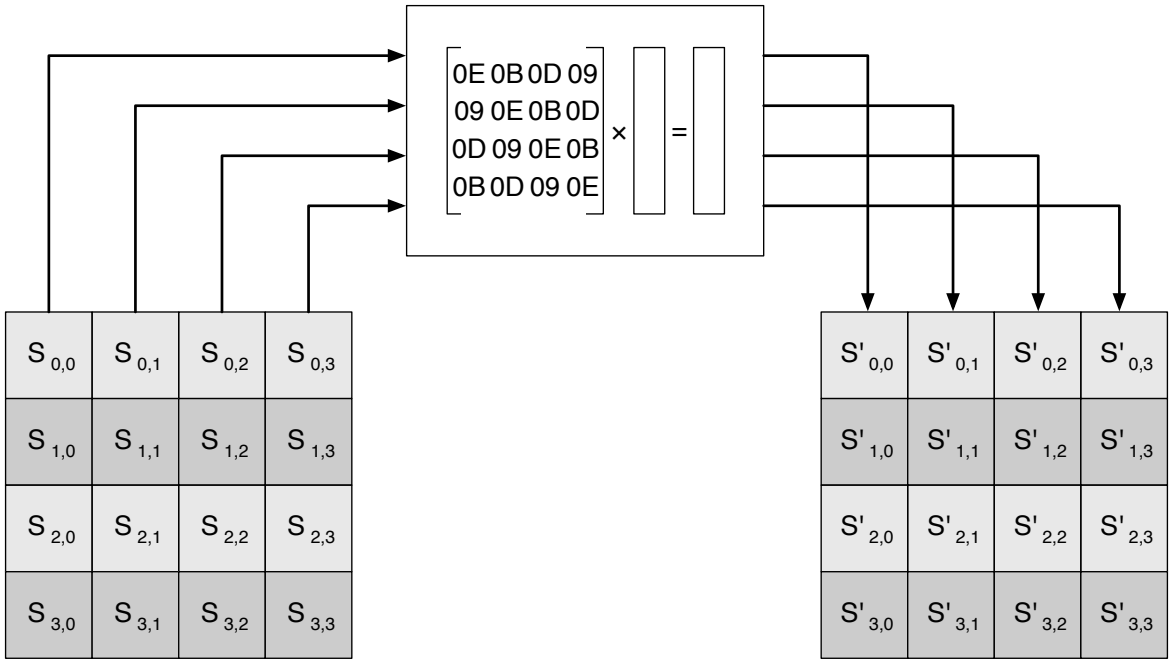
Shift Row Transformation - Begründung

- Wesentlicher als es auf den ersten Blick scheint!
- Der Zustand (*State*) wird ebenso wie die Chiffrierein- und -ausgabe als Array aus vier 4-Byte-Spalten behandelt.
- Bei der Verschlüsselung werden die ersten 4 Bytes des Klartextes in die erste Spalte vom Zustands (*State*) kopiert, und so weiter.
- Der Rundenschlüssel wird spaltenweise auf den Zustand (*State*) angewendet.
- Bei einer Zeilenverschiebung wird also ein einzelnes Byte von einer Spalte in eine andere verschoben, was einem linearen Abstand von einem Vielfachen von 4 Byte entspricht.
- Die Transformation sorgt dafür, dass die 4 Bytes einer Spalte auf vier verschiedene Spalten verteilt werden.

Mix Column Transformation



Inverse Mix Column Transformation



Mix Colum Transformation - Beispiel

Gegeben

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Ergebnis

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Beispiel für die Berechnung von $S'_{0,0}$:

$$\begin{array}{lccccccccc} S'_{0,0} = & 02 \times S_{0,0} & \oplus & 03 \times S_{1,0} & \oplus & 01 \times S_{2,0} & \oplus & 01 \times S_{3,0} & \\ & (02 \times 87) & \oplus & (03 \times 6E) & \oplus & (46) & \oplus & (A6) & = 47. \end{array}$$

Hilfsrechnungen

$$\begin{array}{rcll} (02 \times 87) & = & (0000\,1110) \oplus (0001\,1011) = & (0001\,0101) \\ (03 \times 6E) = 6E \oplus (02 \times 6E) & = & (0110\,1110) \oplus (1101\,1100) = & (1011\,0010) \\ 46 & = & & (0100\,0110) \\ A6 & = & & (1010\,0110) \\ & & & \overline{(0100\,0111)} \end{array}$$

Warnung

$(03 \times 6E) = 6E \oplus (02 \times 6E)$ und **ist nicht** $6E \oplus 6E \oplus 6E$, da wir hier Polynomarithmetik in $GF(2^8)$ nutzen und 03 dem Polynom: $x + 1$ entspricht.

Mix Column Transformation - Begründung

- Die Koeffizienten einer Matrix, die auf einem linearen Code mit maximalem Abstand zwischen den Codewörtern basiert, gewährleisten eine gute Mischung zwischen den Bytes jeder Spalte.
- Die *Mix Column Transformation* (🏴 *Mischspaltentransformation*) kombiniert mit der *Shift Row Transformation* (🏴 *Zeilenverschiebungstransformation*) stellt sicher, dass nach einigen Runden alle Ausgangsbits von allen Eingangsbits abhängen.

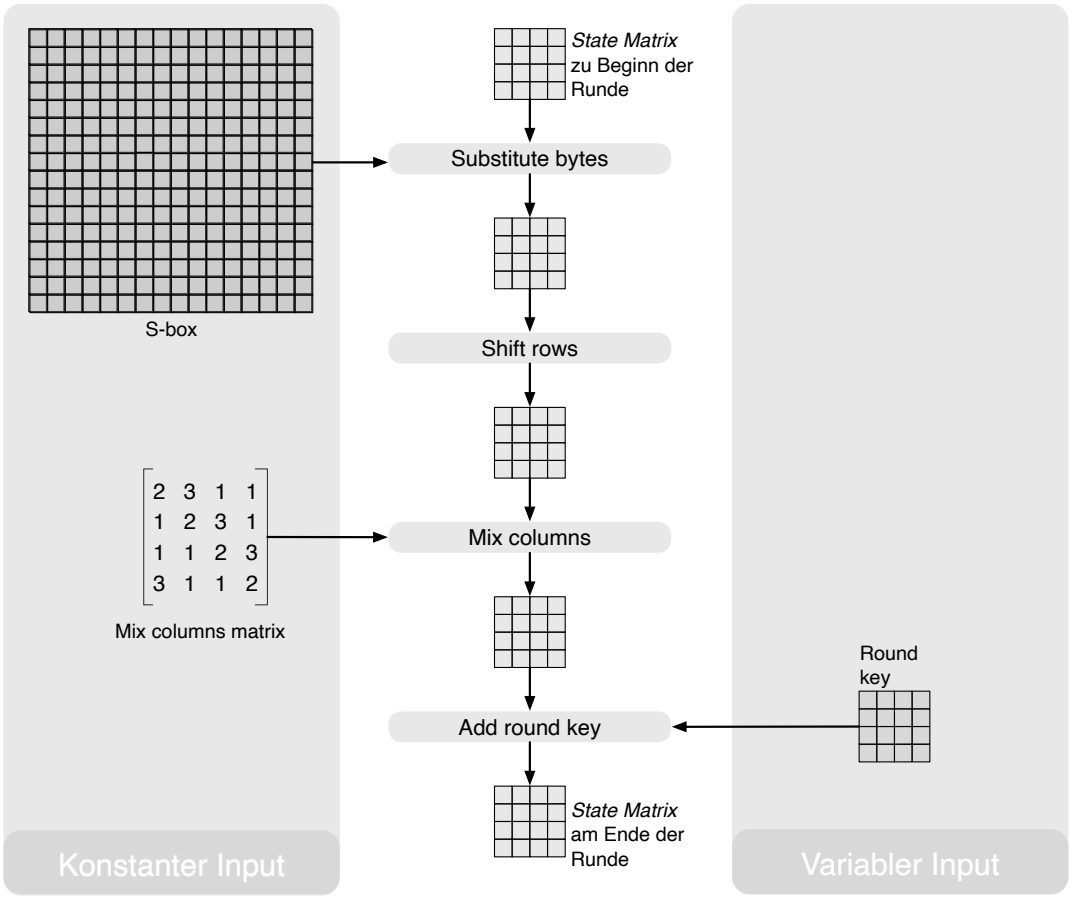
AddRoundKey Transformation

- Die 128 Bits des Zustands (*State*) werden bitweise mit den 128 Bits des Rundenschlüssels XOR-verknüpft.
- Die Operation wird als spaltenweise Operation zwischen den 4 Bytes einer Spalte des Zustands (*State*) und einem Wort des runden Schlüssels betrachtet.
- *Kann auch als eine Operation auf Byte-Ebene betrachtet werden*.

Designbebegründung

- Sie ist so einfach wie möglich und betrifft jedes Bit des Staates.
- Die Komplexität der runden Schlüsselexpansion plus die Komplexität der anderen Stufen von AES sorgen für Sicherheit!

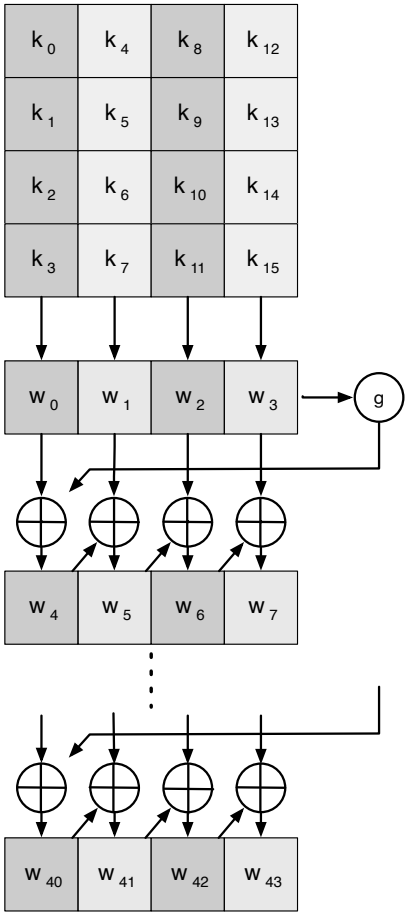
Eingabe für eine einzelne AES-Verschlüsselungsrunde

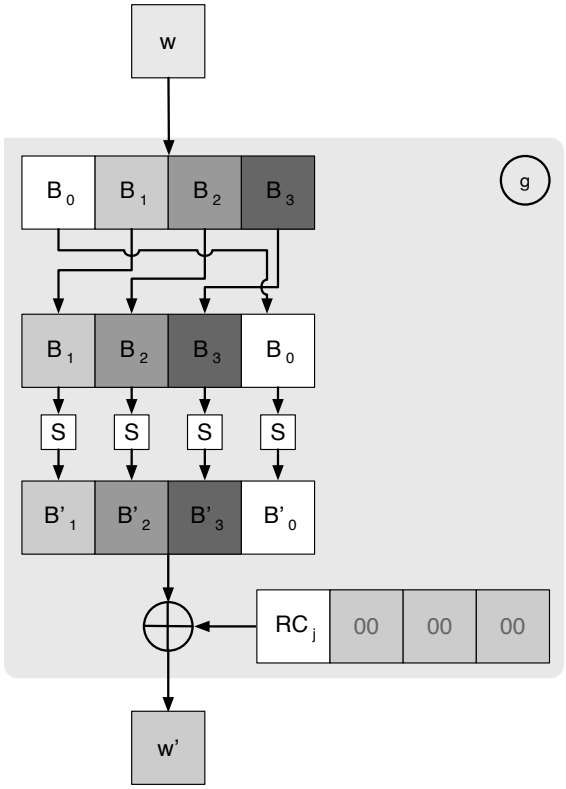


AES Schlüsselexpansion

- Takes as input a four-word (16 byte) key and produces a linear array of 44 words (176) bytes.
- This is sufficient to provide a four-word round key for the initial *AddRoundKey* stage and each of the 10 rounds of the cipher.
- Key is copied into the first four words of the expanded key.
- The remainder of the expanded key is filled in four words at a time.
- Each added word $w[i]$ depends on the immediately preceding word, $w[i–1]$, and the word four positions back, $w[i–4]$
- In three out of four cases a simple XOR is used.
- For a word whose position in the w array is a multiple of 4, a more complex function g is used.

AES Schlüsselexpansion - Visualisiert





AES *Round Key* Berechnung

$$\begin{aligned} r_i &= (r_{c_i}, 00, 00, 00) \\ r_{c_1} &= 01 \\ r_{c_{i+1}} &= xtime(r_{c_i}) \end{aligned}$$

$xtime$ Function

$$y_7y_6y_5y_5y_4y_3y_2y_1y_0 = xtime(x_7x_6x_5x_5x_4x_3x_2x_1x_0) \qquad (x_i, y_i \in \{0,1\})$$

$$y_7y_6y_5y_5y_4y_3y_2y_1y_0 = \begin{cases} x_6x_5x_5x_4x_3x_2x_1x_00, & if x_7 = 0 \\ x_6x_5x_5x_4x_3x_2x_1x_00 \oplus 00011011, & if x_7 = 1 \end{cases}$$

Die *Round Key* Werte sind:

$r_{c_1} = 01, r_{c_2} = 02, r_{c_3} = 04, r_{c_4} = 08, r_{c_5} = 10$

$r_{c_6} = 20, r_{c_7} = 40, r_{c_8} = 80, r_{c_9} = 1B = 00011011, r_{c_{10}} = 36$

AES Schlüsselexpansion - Beispiel (Runde 1)

Gegeben sei: $w[0] = (54, 68, 61, 74)$

$w[1] = (73, 20, 6D, 79)$

$w[2] = (20, 4B, 75, 6E)$

$w[3] = (67, 20, 46, 75)$

- $g(w[3])$:
  - zirkuläre Linksverschiebung von $w[3]$: $(20, 46, 75, 67)$
  - Bytesubstitution mit Hilfe der s-box: $(B7, 5A, 9D, 85)$
  - Addition der Rundenkonstante $(01, 00, 00, 00) \Rightarrow g(w[3]) = (B6, 5A, 9D, 85)$
- $w[4] = w[0] \oplus g(w[3]) = (E2, 32, FC, F1)$
- $w[5] = w[4] \oplus w[1] = (91, 12, 91, 88)$
- $w[6] = w[5] \oplus w[2] = (B1, 59, E4, E6)$
- $w[7] = w[6] \oplus w[3] = (D6, 79, A2, 93)$
- First roundkey is: $w[4]||w[5]||w[6]||w[7]$

AES Key Expansion - Rationale

- The Rijndael developers designed the expansion key algorithm to be resistant to known cryptanalytic attacks
- Inclusion of a round-dependent round constant eliminates the symmetry between the ways in which round keys are generated in different rounds

The specific criteria that were used are:

- Knowledge of a part of the cipher key or round key does not enable calculation of many other round-key bits
- An invertible transformation
- Speed on a wide range of processors
- Usage of round constants to eliminate symmetries
- Diffusion of cipher key differences into the round keys
- Enough nonlinearity to prohibit the full determination of round key differences from cipher key differences only
- Simplicity of description

Lawineneffekt in AES: Änderung im Klartext

Round		# unterschiedlicher Bits
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d6890175070485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36ald891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500f£59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

Lawineneffekt in AES: Änderung im Schlüssel

Runde		# unterschiedlicher Bits
	0123456789abcdeffedcba9876543210 0123456789abcdeffedcba9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fcle8e8ca4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 Occb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

Equivalent Inverse Cipher

AES decryption cipher is not identical to the encryption cipher.

- The sequence of transformations differs although the form of the key schedules is the same.
- Has the disadvantage that two separate software or firmware modules are needed for applications that require both encryption and decryption.

Two separate changes are needed to bring the decryption structure in line with the encryption structure:

1. The first two stages of the decryption round need to be interchanged.
2. The second two stages of the decryption round need to be interchanged.

Interchanging *InvShiftRows* and *InvSubBytes*

- *InvShiftRows* affects the sequence of bytes in State but does not alter byte contents and does not depend on byte contents to perform its transformation
- *InvSubBytes* affects the contents of bytes in State but does not alter byte sequence and does not depend on byte sequence to perform its transformation

Important

Thus, these two operations commute and can be interchanged.

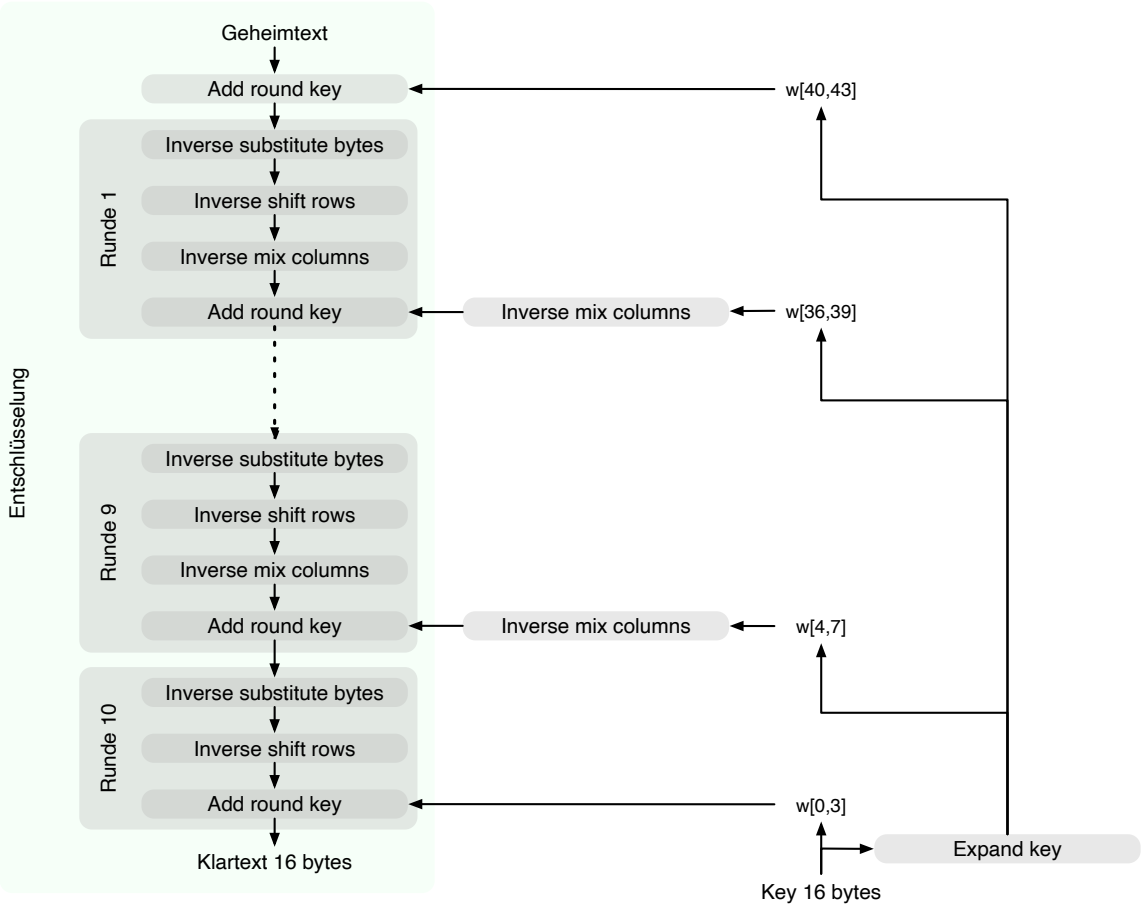
Interchanging *AddRoundKey* and *InvMixColumns*

- The transformations *AddRoundKey* and *InvMixColumns* do not alter the sequence of bytes in State.
- If we view the key as a sequence of words, then both AddRoundKey and InvMixColumns operate on State one column at a time.
- These two operations are linear with respect to the column input.

  That is, for a given State S_i and a given round key w_j:

$$InvMixColumns(S_i \oplus w_j) = InvMixColumns(S_i) \oplus InvMixColumns(w_j)$$

Äquivalente Inverse Chiffre



Implementation Aspects

AES can be implemented very efficiently on an 8-bit processor:

AddRoundKey: is a bytewise XOR operation.

ShiftRows: is a simple byte-shifting operation.

SubBytes: operates at the byte level and only requires a table of 256 bytes.

MixColumns: requires matrix multiplication in the field $GF(2^8)$, which means that all operations are carried out on bytes.

Implementation Aspects

Can be efficiently implemented on a 32-bit processor:

- Redefine steps to use 32-bit words
- Can precompute 4 tables of 256-words
- Then each column in each round can be computed using 4 table lookups + 4 XORs
- At a cost of 4Kb to store tables
- Designers believe this very efficient implementation was a key factor in its selection as the AES cipher