DR. STEFAN M. RITTER

# Advanced Encryption Standard AES

Schwierigkeitsgrad:

Im Jahr 2001 wurde der weit verbreitete symmetrische Verschlüsselungsalgorithmus DES von seinem Nachfolger AES als offizieller Standard in den USA abgelöst. AES gilt als sehr sichere und performante Chiffre, seine elegante mathematische Struktur gibt jedoch auch Anlass zur Kritik.

**IN DIESEM ARTIKEL ERFAHREN SIE…**

Wie AES als Nachfolger von DES entstanden ist;

Wie AES funktioniert;

Wie der Stand der Sicherheit bei AES ist.

**WAS SIE VORHER WISSEN/KÖNNEN SOLLTEN…**

Grundverständnis für mathematische Zusammenhänge;

Interesse an algorithmischen Strukturen;

Interesse an kryptografischen Fragestellungen.

Die symmetrische Blockchiffre Rijndael der belgischen Entwickler Vincent Rijmen und Joan Daemen besitzt als neuer offizieller Standard AES (Advanced Encryption Standard) der NIST (US National Institute for Security Technologies) ([1]) eine weltweit herausragende Stellung unter den Chiffrealgorithmen. Anwendungen reichen von der WPA2-Verschlüsselung beim WLAN über SSH und IPSEC bis hin zu PGP, zudem hat die NSA in den USA im Jahr 2003 AES für die Sicherung von Daten der Stufe top secret freigegeben ([2]). Letzteres ist umso bemerkenswerter, als es sich bei Rijndael um eine nicht-amerikanische Chiffre handelt.

Anders als bei seinem Vorgänger DES (Data Encryption Standard) mit seinen Diskussionen um die Beteiligung der NSA geschah die Festlegung von Rijndael als neuer Standard AES in einem ausgesprochen transparenten Prozess, der vom NIST ausgerufen und durchgeführt wurde. Viele internationale Kryptographen hatten dabei Gelegenheit, sich die möglichen Kandidaten genau anzuschauen. Den Ausschlag für die Entscheidung zugunsten von Rijndael gab insbesondere die Kombination von Einfachheit, Sicherheit und sehr guter Performance auf unterschiedlichen Anwendungssystemen.

Seit Festlegung des Standards unterliegt AES weiteren intensiven Untersuchungen seiner Sicherheitsaspekte, und obwohl er nach wie vor als sehr sicher angesehen wird, gibt es auch kritische Stimmen.

Grund genug, sich mit AES genauer auseinanderzusetzen.

## Ein wenig Geschichte

1976 wurde der DES-Algorithmus von der US-Regierung als nationaler Standard für Verschlüsselung festgelegt und ist seitdem weltweit verbreitet in Gebrauch (gewesen). In den 1990er Jahren jedoch wurden mehr und mehr Angriffsmöglichkeiten gegen DES untersucht und schließlich galt die vom DES verwendete Schlüssellänge von 56 Bits als nicht mehr sicher und tragfähig.

RSA Security schrieb 1997 bis 1999 mehrere so genannte DES Challenges aus, bei denen DES-Schlüssel mittels der Bruteforce Methode gebrochen werden sollten, also einem schlichten Ausprobieren aller möglichen Schlüssel, bis der richtige gefunden ist. Die erste Challenge wurde 1997 vom DESCHALL-Projekt gewonnen, das dazu 96 Tage benötigte.

Der große Durchbruch gelang mit Deep Crack, einer EFF (Electronic Frontier Foundation) Cracker-Maschine, die einen vorgegebenen DES-Schlüssel 1998 in 56 Stunden brach. Der Code-Brecher COPACOBANA [3] ist seit März 2007 in der Lage, jeden DES-Schlüssel im Mittel in 6,4 Tagen zu brechen, mit einer maximalen Dauer von 12,8 Tagen.

DES ist jedoch auch heute noch in aktivem Gebrauch, vor allem in Form von 3DES (Triple DES), bei dem der DES-Algorithmus dreifach

hintereinander ausgeführt wird (genauer gesagt in der Reihenfolge DES,$DES^{-1}$,DES). Typische Anwendungsgebiete sind zum Beispiel Chipkarten für Banksysteme.

Mit 3DES sind Schlüssellängen von 112 oder 168 Bits möglich. 3DES gilt als sicherer Algorithmus, wesentliche Nachteile sind jedoch die längere Rechenzeit im Vergleich zu DES sowie die Beschränkung der Blockgröße auf 64 Bits.

Folgerichtig eröffnete das NIST im Januar 1997 ein Verfahren, um einen Nachfolger von DES zu finden, der den Namen AES erhalten sollte. Ziel war es, einen Algorithmus mit mindestens der gleichen Stärke wie 3DES zu finden, der jedoch gleichzeitig die zum Teil bereits angedeuteten Einschränkungen von 3DES nicht besitzen sollte. Drei wesentliche, allgemeine Kriterien wurden an den neuen Algorithmus gestellt:

- Sicherheit,
- Kosten,
- Algorithmus-Eigenschaften/Implementationseigenschaften.

Dazu gehörten die Forderungen, dass der Algorithmus gegen die bekannten kryptografischen Angriffe widerstandsfähig ist, eine Blockgröße von 128 Bits sowie Schlüssellängen von 128, 192 und 256 Bits unterstützt. Wichtig war zudem, eine möglichst effiziente Implementierung sowohl in Hard- als auch in Software-Lösungen zu bieten, was sich in niedrigem Speicherbedarf sowie schnellen Laufzeiten niederschlagen sollte. Auch durfte es keine patentrechtlichen Ansprüche geben.

Im Laufe der Evaluierung wurden von fünfzehn ernsthaften Kandidaten im August 1999 die folgenden fünf zur finalen Runde ausgewählt: Mars, RC6, Rijndael, Serpent und Twofish. Unter den Entwicklern dieser Algorithmen befanden sich dabei beispielsweise Größen wie Bruce Schneier, Eli Biham und Ronald Rivest. Die Wahl fiel schließlich mit Rijndael auf einen Algorithmus der belgischen Kryptographen Vincent Rijmen und Joan Daemen.

Als Randnotiz mag dabei interessant sein, dass der Beitrag der Deutschen Telekom, Magenta, im Prinzip bereits während der Präsentation wegen massiver Sicherheitsbedenken durchfiel.

Rijndael wurde schließlich von der NIST 2001 als neuer offizieller Standard verabschiedet (FIPS 197, [1]).

## Der AES-Algorithmus

Rijndael ist eine symmetrische Blockchiffre und unterstützt frei kombinierbare Block- und Schlüssellängen von jeweils 128, 196 oder 256 Bits. AES hingegen schreibt im NIST-Standard eine Blockgröße von 128 Bits sowie Schlüssellängen von 128, 192 oder 256 Bits vor, reduziert Rijndael also auf eine feste Blockgröße.

AES verwendet einige grundlegende mathematische Operationen, die in mehren Runden immer wiederholt werden. Im Gegensatz zu DES ist AES jedoch keine Feistelchiffre, i.e. in jeder Runde wird immer der gesamte Inputblock gleichzeitig verarbeitet. In Abhängigkeit von der Schlüssellänge variiert auch die Anzahl der durchzuführenden Runden:

Grundlegende Eigenschaften von AES sind:

- AES ist kein Feistel-Algorithmus;
- AES besitzt eine einfache, elegante mathematische Struktur und verwendet eine Arithmetik über dem endlichen Körper GF($2^8$);
- die Rundenfunktion besteht aus vier Transformationen, davon drei Substitutionen und eine Permutation;
- die einzelnen Transformationen sind leicht invertierbar (-› Entschlüsselung);
- die Rundenschlüssel-Addition besteht aus einem Bit-weisen XOR;
- Verschlüsselungs- und Entschlüsselungsalgorithmus sind nicht identisch; bei der Entschlüsselung werden die invertierten Transformationen in umgekehrter Reihenfolge durchgeführt.;
- Vor der ersten Runde wird eine AddRoundKey-Transformation durchgeführt;
- die letzte Runde besteht nur aus drei Transformationen.

**Tabelle 1.** *AES*

| Bezeichnung | Schlüssellänge | Rundenanzahl | Blocklänge |
|---|---|---|---|
| AES-128 | 128 Bits | 10 | 128 Bits |
| AES-192 | 192 Bits | 12 | 128 Bits |
| AES-256 | 256 Bits | 14 | 128 Bits |

Die weitere Darstellung folgt bei der Nomenklatur im Wesentlichen der offiziellen Darstellung aus [1].

Die grundlegende Einheit bei der AES-Verarbeitung ist das Byte, das seinerseits aus acht Bits besteht. Ein solches Byte kann in verschiedenen Weisen dargestellt werden:

- binär: $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, mit $b_i$ aus `{0,1}`
- hexadezimal: $\{h_1h_0\}$, mit $h_i$ aus `{0,...,f}`
- polynomial: $b(x) = b_7x^7 + b_6x^6 + ... + b_1x + b_0$

Beispiel: $\{011001011\} = \{4b\} = x^6 + x^3 + x + 1$

Bei der hexadezimalen Schreibweise gilt hier zu beachten, dass jeweils die erste und zweite Hälfte der Bit-Folge in einen Hexadezimalwert umgewandelt wird, in dem vorigen Beispiel also 0100 zu `4` und 1011 zu `b`, was dann in dieser Notation den Wert `{4b}` ergibt. Dies wird später noch wichtig sein.

Bei der Darstellung als Polynom wird `b(x)` als Element in einer speziellen mathematischen Struktur, dem endlichen Körper GF($2^8$) mit $2^8$ Elementen, aufgefasst. `b(x)` wird als Polynom vom Höchstgrad sieben realisiert, um die acht Bits eines Bytes darstellen zu können.

Intern verwaltet AES den zu verarbeitenden Input-Block als zweidimensionales Array, bezeichnet als State Matrix oder State Array. Diese Matrix besteht aus vier Reihen und vier Spalten, wobei jeder Eintrag in der Matrix jeweils aus einem Byte besteht. Also insgesamt 16 Bytes, was einer Blocklänge von 128 Bits entspricht. Die einzelnen Einträge der Matrix werden typischerweise mit `s[r,c]` oder $S_{r,c}$ referenziert, wobei r die Zeile (Werte 0 bis 3) und c die Spalte (Werte 0 bis 3) bezeichnet.

Abbildung 1 zeigt, wie der ursprüngliche Input-Block in die State Matrix abgebildet und schließlich nach Abschluss der r Runden des Algorithmus den Output-Block ergibt. Dieses Verfahren ist bei Ver- und Entschlüsselung identisch.

Die State Matrix lässt sich noch anders auffassen, indem man die vier

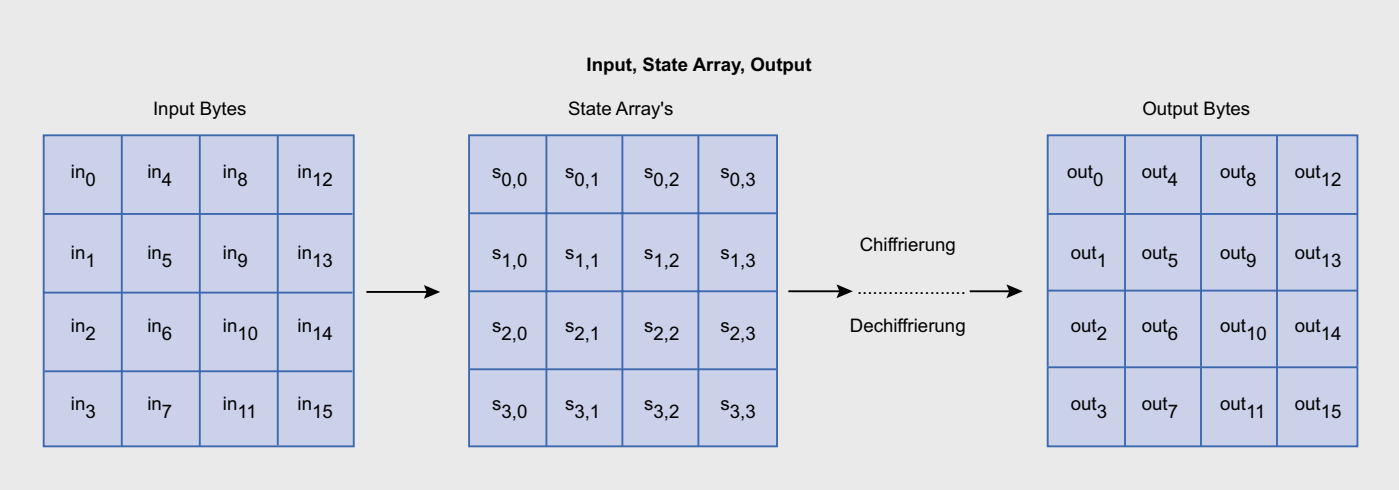


**Abbildung 1.** *Input-Block, State Matrix, Output-Block*

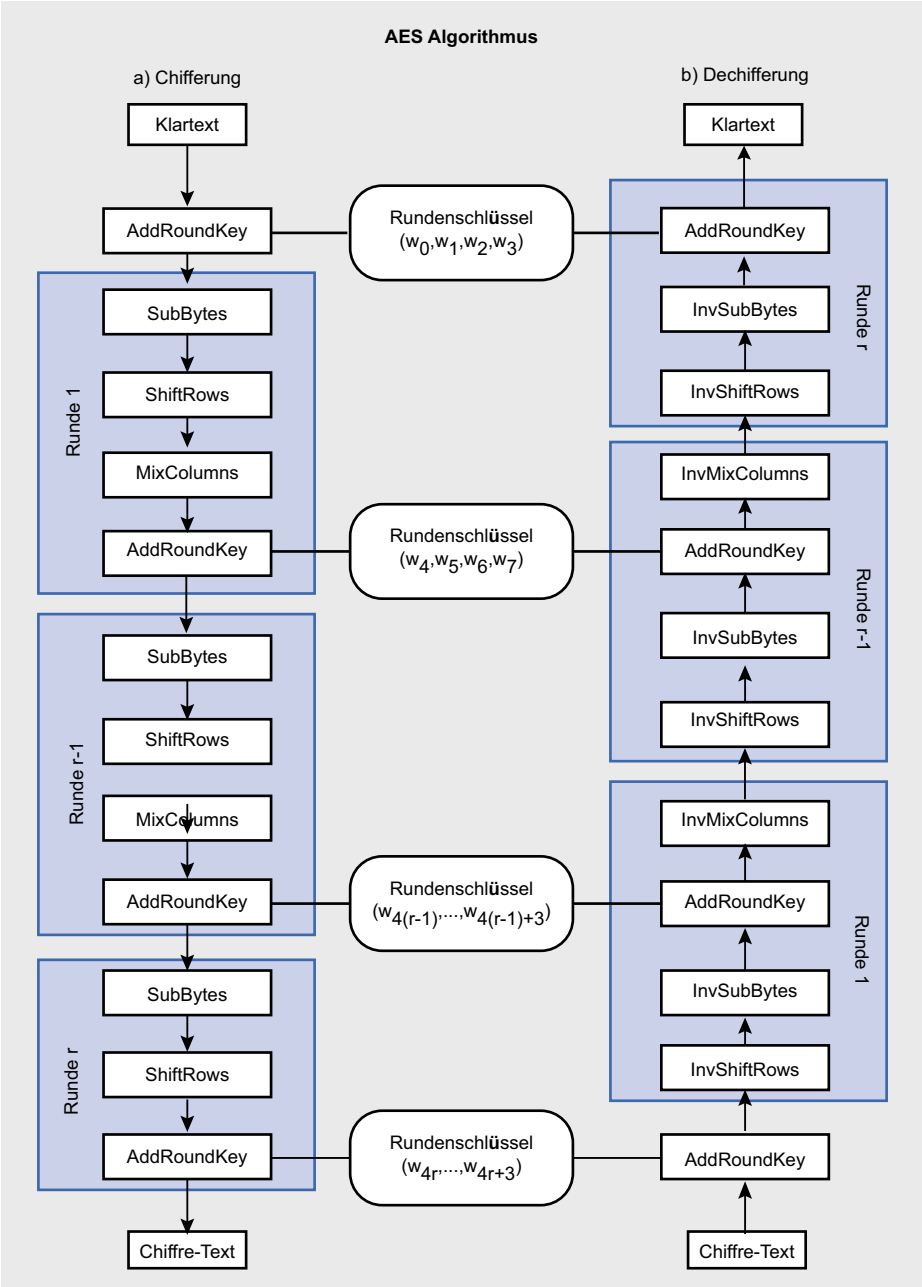


**Abbildung 2.** *AES-Algorithmus*

Bytes jeder Spalte als ein 32-Bit-Wort auffasst. Dann besteht die State Matrix aus einem eindimensionalen Array mit vier 32-Bit-Wörtern als Einträgen. Die einzelnen Wörter werden mit $w_0,\ldots,w_3$ bezeichnet, wobei dann zum Beispiel $w_0 = s_{0,0}s_{1,0}s_{2,0}s_{3,0}$ ist.

Da Ver- und Entschlüsselungsalgorithmus nicht identisch sind, soll zunächst die Verschlüsselung vorgestellt werden.

## AES-Chiffre

Der Ablauf bei der Chiffrierung ist in Abbildung 2a dargestellt. Zu Beginn des Algorithmus wird, wie in Abbildung 1 gezeigt, der Input-Block (128 Bits) in die State Matrix kopiert. Dann wird eine Rundenschlüssel-Addition durchgeführt, woraufhin sich die Anwendung von r Rundenfunktionen anschließt.

Die von AES zur Verschlüsselung verwendete Rundenfunktion besteht aus vier grundlegenden Transformationen:

- S-Box Byte-Substitution (`SubBytes`),
- Permutation der Inputblock-Zeilen (`ShiftRows`),
- Substitution der Inputblock-Spalten (`MixColumns`),
- Addition eines Rundenschlüssels (`AddRoundKey`).

Die letzte Runde beinhaltet dabei als einzige nicht die `MixColumns`-Transformation.

Da in jeder Runde ein anderer Schlüssel (Rundenschlüssel) verwendet wird, werden aus dem initialen Chiffre-Schlüssel die übrigen Rundenschlüssel erzeugt, also bei `r` Runden insgesamt `r+1`

Schlüssel. Die `AddRoundKey`-Operation besteht dabei aus einem einfachen XOR der einzelnen Bits.

### SubBytes

Die einzige nicht-lineare Transformation ist die Substituierung der einzelnen Bytes des Input-Blocks mittels einer so genannten S-Box. Anders als bei DES, wo es acht S-Boxen gibt, verwendet AES eine S-Box sowie deren Inverse. Diese S-Box ist eine 16x16 Matrix, deren Einträge aus einer Permutation aller möglichen Bytes bestehen (siehe Abbildung 3a).

Bei der SubBytes-Transformation geschieht folgendes: Jedes Byte der State Matrix wird in einen linken und einen rechten Teil zerlegt.

Die vier linken Bits werden dann als Zeilenindex, die vier rechten Bits als Spaltenindex in der S-Box verwendet, um den neuen Wert des Input-Bytes zu finden. Durch diesen Wert wird dann das ursprüngliche Byte ersetzt. Ein Beispiel möge dies verdeutlichen:

Angenommen, das Input-Byte $s_{1,1}$ habe den Wert `{11010011}`, was in der erwähnten Hexadezimal-Schreibweise `{d3}` entspricht. Nimmt man nun `d` als Zeilenindex und `3` als Spaltenindex, so ergibt sich in der S-Box aus Abbildung 3 der Wert `{66}`. Das ursprüngliche Byte `{d3}` wird also durch `{66}` ersetzt.

Dieser Vorgang wird für alle 16 Bytes der State Matrix durchgeführt, wodurch sich ein neuer Zustand der Matrix ergibt. Abbildung 4a zeigt den Vorgang nochmals schematisch.

Mathematisch gesehen besteht die Transformation aus der Verkettung zweier Transformationen, der multiplikativen Inversenbildung über dem endlichen Körper GF($2^8$)sowie einer affinen Transformation über dem Körper GF(2).

Die S-Box ist von den Entwicklern so gestaltet, dass sie gegen lineare und differentielle Kryptanalysen nicht anfällig ist, und bildet damit ein Herzstück des Algorithmus'.

### ShiftRows

Durch die ShiftRows-Transformation werden die Bytes der letzten drei Zeilen der State Matrix zyklisch vertauscht, und zwar mit unterschiedlichen Offsets. Die Bytes der zweiten Zeile werden dabei um ein Feld, die Bytes der dritten Zeile um zwei Felder, und die Bytes der vierten Zeile um drei Felder nach links verschoben. Abbildung 4c zeigt dies schematisch.

Dadurch werden die vier Bytes einer State-Spalte auf vier verschiedene Spalten verteilt, wodurch eine Diffusion der einzelnen Wörter erreicht wird. ShiftRows führt also eine Permutation der Input-Daten aus.

### MixColumns

Die MixColumns-Transformation arbeitet im Gegensatz zu ShiftRows auf den einzelnen Spalten der State Matrix, wobei jedes einzelne Byte einer Spalte auf einen neuen Wert geändert wird, welcher von allen vier Bytes dieser Spalte abhängig ist. Die Spalten werden dabei mit einer fest vorgegebenen Matrix (Abbildung 4e) multipliziert, wodurch sich die neue Spalte ergibt. Mathematisch gesehen werden die einzelnen Spalten der State Matrix als Polynome vom Höchstgrad 3 über GF($2^8$) aufgefasst, mit einer Polynommultiplikation modulo $(x^4+1)$. Dabei sind dann die Koeffizienten jedoch keine Bits mehr, sondern Bytes.

**AES: S-Box**

| Zeile x \ Spalte 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 |

**AES: Inverse S-Box**

| Zeile x \ Spalte 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 | 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 |

**Abbildung 3.** *AES-S-Box*

Ziel der MixColumns-Transformation ist die Schaffung von Diffusion innerhalb der Wörter (Spalten) und Bytes.

### AddRoundKey

Bei der AddRoundKey-Transformation wird ein Rundenschlüssel per XOR zur State Matrix addiert. Genauer gefasst: Jeder Rundenschlüssel (128 Bits) besteht aus vier 4-Byte-Wörtern, die zu den vier Spalten der State Matrix addiert werden. Abbildung 2 zeigt, dass bei r Runden insgesamt `r+1` Schlüssel benötigt werden, die in einem speziellen Verfahren (siehe nächster Abschnitt) erzeugt werden. Diese `r+1` Schlüssel bestehen wiederum insgesamt aus `4(r+1)` Wörtern.

Abbildung 4b zeigt schematisch den AddRoundKey-Vorgang mit Schlüsselwörtern $w_{4j}, w_{4j+1}, w_{4j+2}, w_{4j+3}$

## Schlüsselalgorithmus (Key Expansion)

Nachdem mehrfach von Rundenschlüsseln die Rede war, stellt sich die Frage, wie diese 4(r+1) Schlüsselwörter generiert werden, denn zu Beginn gibt es nur einen initialen Schlüssel der Länge 128, 196 oder 256 Bits, was vier, sechs oder acht 4-Byte-Wörtern entspricht (im Folgenden mit `Nk` bezeichnet, also `Nk = 4,6,8`). Dieser initiale Schlüssel wird als *Chiffre-Schlüssel* bezeichnet, nicht zu verwechseln mit den einzelnen Rundenschlüsseln.

Diese `r+1` Rundenschlüssel (genauer gesagt: deren vier Wörter) lassen sich in einem Array w anordnen, wobei dann $w_i$ das i-te Wort in diesem Array bedeutet $(0 <= i < 4(r+1))$.

Da jeder Rundenschlüssel aus 128 Bits, also vier Wörtern zusammengesetzt ist, besteht zum Beispiel der Schlüssel der Runde 1 aus den Wörtern $w_4$ bis $w_7$ und der letzte Rundenschlüssel aus den Wörtern $w_{4r}$ bis $w_{4r+3}$.

Man beachte zum Verständnis, dass gemäß Abbildung 2 vor der ersten Runde bereits ein AddRoundKey stattfindet, weshalb die ersten vier Schlüsselwörter ( $w_0$ bis $w_3$ ) bereits vor der ersten Runde verbraucht sind.

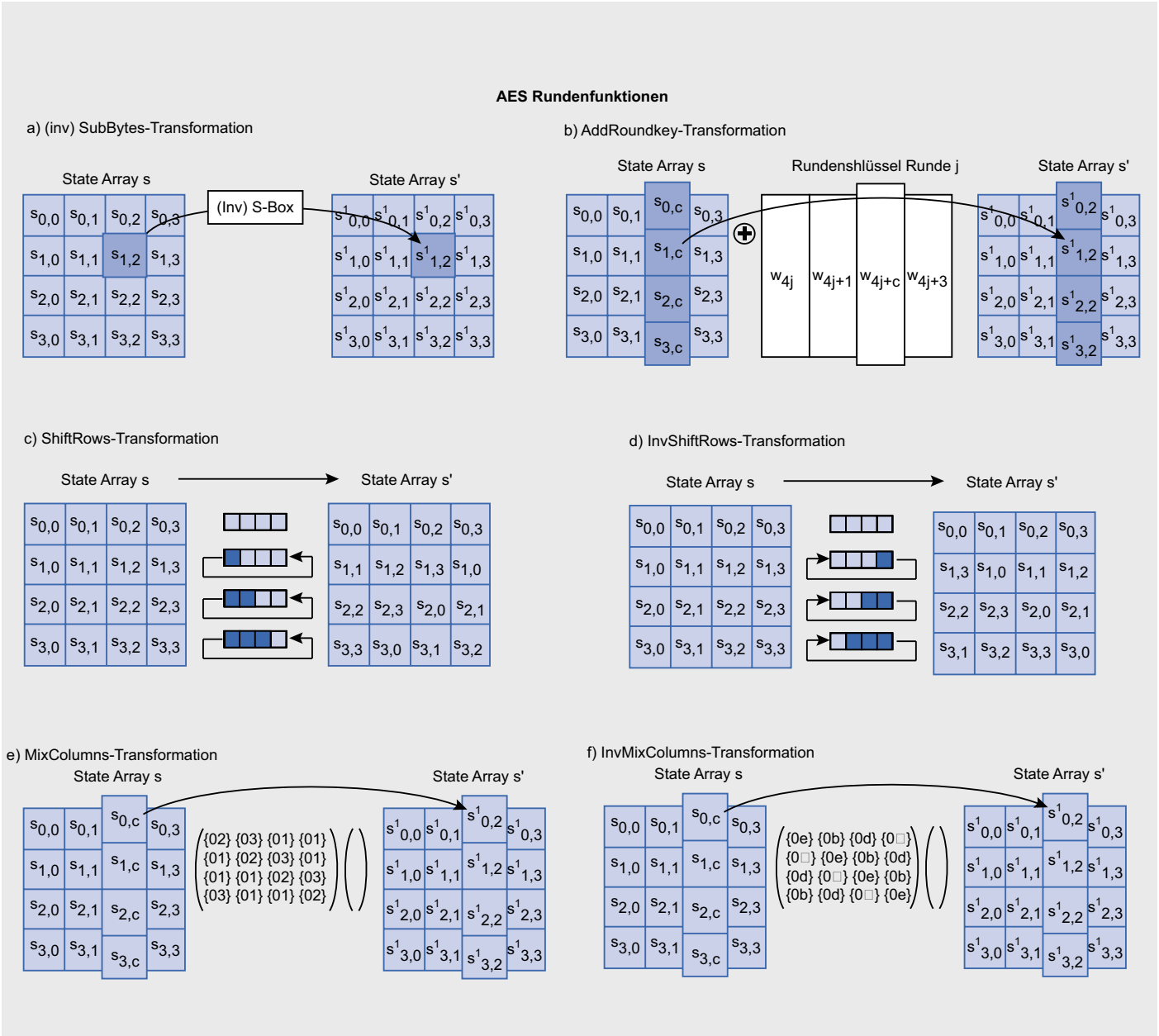


**Abbildung 4.** *AES-Rundenfunktionen*

Die Schlüsselgenerierung (Key Expansion) geschieht folgendermaßen:

Die ersten Nk Wörter des Schlüsselarrays w werden mit dem Chiffre-Schlüssel gefüllt. Jedes weitere Schlüsselwort wi entsteht aus dem XOR der Schlüsselwörter $w_{i-1}$ und $w_{i-NK}$ :

$$w_i = w_{i-1} \text{ XOR } w_{i-Nk}$$

Wörter $w_i$, deren Index i ein Vielfaches von `Nk` ist, werden jedoch anders behandelt, indem $w_{i-1}$ vor dem XOR mit $w_{i-NK}$ einer speziellen Transformation unterworfen wird. Diese Transformation, hier mit f bezeichnet, besteht aus einer zyklischen Verschiebung der Bytes `(RotWord)`, gefolgt von einer S-Box-Substitution `(SubWord)`. Anschließend wird noch eine rundenabhängige Konstante per `XOR` hinzuaddiert `(Rcon[i])`: $f(w_{4_r}) = SubWord(RotWord(w_{i-1}))XOR\ Rcon[i]$ und

$$w_i = f(w_{4r})XOR\ w_{i-Nk}$$

Im Falle einer Chiffre-Schlüssellänge von 256 Bits (`Nk = 8`) ist dieser Algorithmus leicht geändert, indem in dem Fall, dass der Index (`i-4`) ein Vielfaches von `Nk` ist, die Transformation f nur aus der S-Box-Substitution `(SubWord)` besteht.

Um dies etwas deutlicher zu machen, zeigt Abbildung 5 die initiale Schlüsselbefüllung des Schlüsselarrays am Beispiel einer Chiffre-Schlüssellänge von 128 Bits (entspricht `Nk = 4`). Dort sieht man, dass in die Berechnung von $w_4$ die Transformation f eingeht, da der Index 4 ein Vielfaches von `Nk = 4` ist.

## Umkehrung der AES-Chiffre

Die einzelnen Rundentransformationen bei der Verschlüsselung lassen sich invertieren und ergeben bei umgekehrter Ausführung den Entschlüsselungsalgorithmus (siehe Abbildung 2b). Zu beachten ist dabei, dass die Rundenschlüssel ebenfalls in umgekehrter Reihenfolge verwendet werden.

Die vier Transformationen der Rundenfunktion lauten also

- InvShiftRows,
- InvSubBytes,
- AddRoundKey,
- InvMixColumns.

Die Inverse der AddRoundKey-Transformation ist mit AddRoundKey identisch, da es sich dabei um ein einfaches XOR handelt.

Die Durchführung von InvShiftRows und InvMixColumns zeigen Abbildung 4d bzw. 4f.. InvShiftRows führt in Umkehrung zu ShiftRows in der zweiten Zeile eine Verschiebung um ein Byte nach rechts, in der dritten Zeile um zwei Bytes und in der vierten Zeile um drei Bytes durch. InvMixColumns führt die gleiche Operation wie MixColumns durch, jedoch mit der inversen Matrix.

InvSubBytes verwendet die inverse S-Box (siehe Abbildung 3b). Analog zu den Beschreibungen von SubBytes wird zum Beispiel das Byte `{66}` durch `{d3}` ersetzt.

Da die Entschlüsselung nicht mit der Verschlüsselung identisch ist, bedeutet dies in der Praxis, dass für die beiden Modi verschiedene Module bereitgestellt werden müssen, was ein wenig unpraktisch ist. Daher spezifiziert der Standard für AES eine zweite Möglichkeit der Entschlüsselung, genannt Equivalent Inverse Cipher. Hierbei ist die Reihenfolge der Transformationen in den Runden bei Ver- und Entschlüsselung die gleiche, die einzige dazu notwendige Änderung ist eine Anpassung in der Rundenschlüsselverwendung. Vor der Anwendung von AddRoundKey muss nämlich zusätzlich die Transformation InvMixColumns auf den Rundenschlüssel angewandt werden.

## Sicherheit von AES

Um allgemein die Sicherheit eines Chiffre-Algorithmus beurteilen zu können, ist zunächst wichtig, die möglichen Angriffe zu klassifizieren. Drei Arten von Angriffen sind dabei relevant:

- Brute-Force Angriffe,
- Kryptanalyse,
- Side-Channel Angriffe.

Side-Channel-Angriffe richten sich nicht direkt gegen den Algorithmus, sondern gegen die physische Implementation, Beispiele sind *timing attacks* oder *power attacks*. Bei ersterem wird versucht über die Messung von Unterschieden in der Laufzeit der Ver-/Entschlüsselung Informationen über den Schlüssel zu finden. Power attacks dagegen untersuchen den Stromverbrauch bei den Berechnungen. In [4] werden solche Angriffe gegen AES mittels Informationspreisgabe des CPU Cache beschrieben.

Eine Brute-Force-Attacke zielt ab auf die verwendete Schlüssellänge, i.e. es werden schlichtweg alle möglichen Schlüssel ausprobiert, bis der richtige gefunden ist. Bei einer Schlüssellänge von n Bits gibt es $2^n$ mögliche Schlüssel, so dass im Mittel $2^{n-1}$ Versuche erforderlich sind. Eine Schlüssellänge von mindestens 128 Bits gilt derzeit

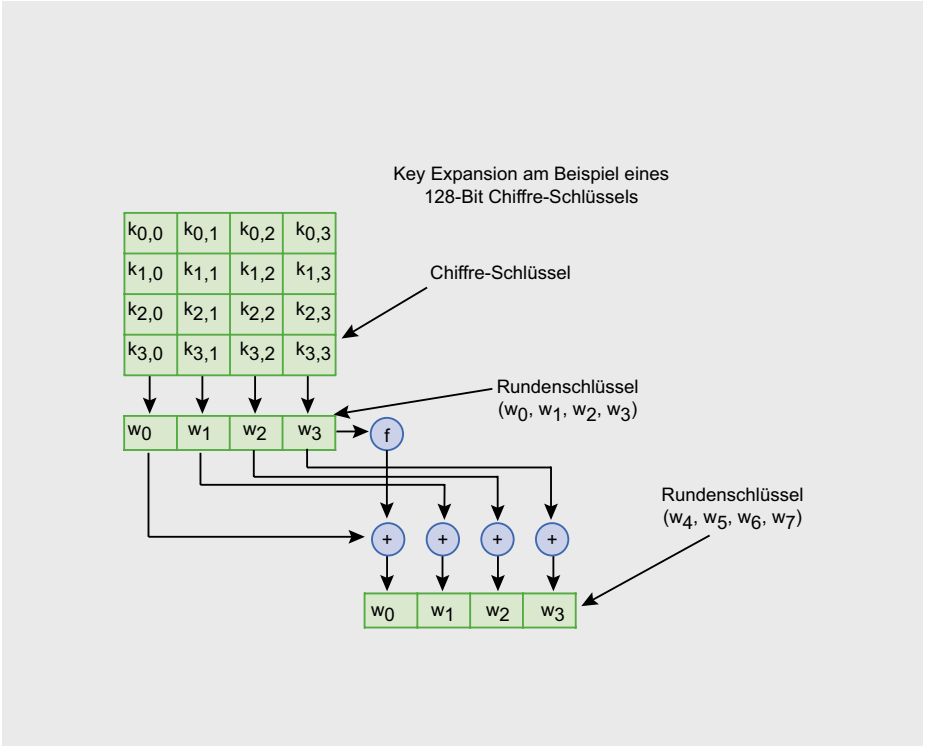


**Abbildung 5.** *AES Key Expansion*

## Begriffe rund um AES

**Block-Chiffre**
Eine Block-Chiffre transformiert einen n-Bit Klartextblock in einen n-Bit Chiffretextblock unter Verwendung eines Schlüssels k. Beispiele sind DES, 3DES oder AES. Eine *iterierte* Block-Chiffre ist eine Block-Chiffre, bei der die Transformation in mehreren Runden hintereinander ausgeführt wird. In jeder dieser Runden wird ein unterschiedlicher Schlüssel verwendet.

**Feistel-Chiffre**
Bezeichnet eine allgemeine Struktur, in der Block-Chiffren realisiert werden können. Vor jeder Runde wird der Blocktext in eine linke und eine rechte Hälfte geteilt. Die rechte Hälfte wird mit einer, von der Chiffre abhängigen Funktion verändert, wobei Teile des Schlüssels eingehen. Das Ergebnis wird mit XOR mit der linken Blockhälfte verknüpft. Dieses Resultat ist dann die rechte Blockhälfte für die nächste Runde, die alte rechte Blockhälfte wird die neue linke. Beispiel für Feistel-Chiffre sind DES, 3DES oder Twofish. AES ist keine(!) Feistel-Chiffre.

**Chiffre-Schlüssel**
Der initiale Schlüssel, der zur Generierung der restlichen Rundenschlüssel verwendet wird. Mögliche Längen bei AES sind 128, 192 oder 256 Bits.

**Rundenschlüssel**
Die Schlüssel, die in den einzelnen Runden verwendet werden und aus dem Chiffre-Schlüssel generiert werden. Die Länge beträgt bei AES 128 Bits.

**State Matrix/Array**
Bezeichnung für den zu verarbeitenden Datenblock (sowie seine Zwischenzustände) aus 128 Bits, aufgefasst als 4x4-Matrix mit Byte-Einträgen bzw. als Array mit vier Einträgen, die jeweils aus 32-Bit Wörtern bestehen.

**Nk, Nr, Nb**
`Nk` bezeichnet die Anzahl der 32-Bit Wörter des Chiffre-Schlüssels (4, 6 oder 8). `Nr` beschreibt die Anzahl der Runden (10, 12 oder 14) und `Nb` steht für die Anzahl der Spalten der State Matrix, die für den AES-Standard immer 4 beträgt.

**Wörter**
Vier Bytes werden zu einem 32-Bit-Wort zusammengefasst.

**AddRoundKey**
Diese Transformation addiert (XOR) einen Rundenschlüssel zu einem Inputblock.

**SubBytes**
Mittels einer S-Box wird eine Byte-Substitution durchgeführt. Die Umkehrung heißt `InvSubBytes`.

**ShiftRows**
Zyklische Vertauschung der Bytes der Zeilen der State Matrix. Die Umkehrung heißt `InvShiftRows`.

**MixColumns**
Transformation der Spalten der State Matrix durch Multiplikation mit einem speziellen, fixen Polynom vierten Grades. Die Umkehrung heißt `InvMixColumns`.

**SubWord**
Bei der Rundenschlüsselgenerierung wendet diese Transformation die S-Box auf jedes Byte eines 4-Byte-Wortes an.

**RotWord**
Bei der Rundenschlüsselgenerierung führt RotWord für ein Wort eine Byteverschiebung um ein Byte nach links durch.

**Rcon**
Eine rundenabhängige Konstante, die bei der Rundenschlüsselgenerierung verwendet wird.
Schlüsselalgorithmus (Key Expansion)
Algorithmus zur Erzeugung der Rundenschlüssel.

**Endliche Körper**
In der Mathematik bezeichnen Körper algebraische Strukturen mit besonderen Eigenschaften. So ist innerhalb eines Köpers Addition, Subtraktion, Multiplikation und Division möglich, ohne die Menge der Elemente zu verlassen. Das bedeutet zum Beispiel, dass Addition zweier Körperelemente wieder ein Element des Körpers ergibt.

Endliche Körper sind Körper mit endlich vielen Elementen, wobei die Anzahl der Elemente eines endlichen Körpers stets die Potenz einer Primzahl ist ($p^n$). In der Kryptografie wichtige endliche Körper sind $F_2$ und $F_{256}$, auch bezeichnet als GF(2) bzw. GF($2^8$). GF(2) entspricht dabei der Menge `{0,1}` mit der bekannten XOR-Verknüpfung als Addition.

**Polynomdarstellung in GF($2^8$)**
Ein Byte lässt sich mit seinen acht Bits $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ als Element im endlichen Körper GF($2^8$) auffassen, und zwar in Form eines Polynoms vom Grad kleiner 8:
$b(x) = b_7x^7 + b_6x^6 + ... + b_1x + b_0$ mit Koeffizienten $b_i$ aus `{0,1}`.

Die Addition in GF($2^8$) stellt die bekannte XOR-Verknüpfung der Koeffizienten dar. Die Multiplikation in GF($2^8$) ist realisiert als Polynommultiplikation modulo eines speziellen (irreduziblen) Polynoms vom Grad 8. Beim AES-Algorithmus lautet dieses Polynom $m(x) = x^8 + x^4 + x^3 + x + 1$.
Damit realisiert AES den Körper GF($2^8$) als Erweiterungskörper $GF(2)[x]/(m(x))$ vom Grad 8 mit dem über GF(2) irreduziblen Polynom $m(x) = x^8 + x^4 + x^3 + x + 1$.

AES verwendet diese Struktur in der S-Box-Transformation.

**Polynomdarstellung in $R_4$**
Ein 4-Byte-Wort lässt sich als Polynom vom Grad 4, $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$, auffassen, wobei die Koeffizienten selber Elemente des endlichen Körpers GF($2^8$) sind. Dies entspricht mathematisch dem Ring $R_4 = GF(2^8)[x]/(x^4 + 1)$, als Reduktionspolynom wird $m(x) = x^4 + 1$ verwendet.
AES verwendet diese Struktur in der Rundenfunktion `MixColumns`.