Seminar Kryptographie

Christian Wilkin

Seminararbeit

WS 2004/2005

Dezember 2004

Betreuung:

Prof. Dr. Alfred Scheerhorn

Fachbereich Design und Informatik
Fachhochschule Trier
University of Applied Sciences

FACHHOCHSCHULE TRIER
UNIVERSITY OF APPLIED SCIENCES
Fachbereich
DESIGN UND INFORMATIK

Autor: Christian Wilkin

Titel: Der Algorithmus des “Advanced Encryption Standard”

Studiengang: Informatik

Betreuung: Prof. Dr. Alfred Scheerhorn

http://www.ainformatik.fh-trier.de/~scheerhorn/

Dezember 04

Unterschrift des/der Autors/ren

0 Inhaltsverzeichnis

1 Zusammenfassung

Bisher war der Data Encryption Standard (DES) der am häufigsten genutzte symmetrische Algorithmus zur Verschlüsselung von Daten. Trotzdem wurde er oft stark kritisiert. Da er lange Zeit nicht vollsändig veröffentlicht war, wurden sogar Hintertüren der amerikanischen National Security Agency vermutet, die allerding nie gefunden wurden. Zweifel an der Sicherheit von DES waren jedoch begründet (schliesslich benutzt DES nur einen 56 Bit langen Schlüssel) und wurden duch die Electronic Frontier Foundation bestätigt, die 1998 eine Maschine zur Entschlüsselung von DES mittels Brute-Force-Verfahren[1] baute. Es musste also ein neues Verschlüsselungsstandard entwickelt werden, der nicht die Fehler seiner Vorgänger besitzt.

[1] Alle möglichen Schlüssel werden nacheinander durchprobiert. Die Reihenfolge wird gegebenenfalls nach der Wahrscheinlichkeit ausgewählt. Diese Methode ist auch bei modernen Verschlüsselungsverfahren sinnvoll, wenn von der Verwendung eines relativ schwachen Passwortes ausgegangen werden kann.

Die Wahl fiel hierbei auf den Rijndael Alorithmus (nach seinen Entwicklern Joan Daemen und Vincenr Rijmen) welchen ich im Folgenden näher beschreiben werde.

2 Rijndael – vom Vorschlag zum Standard

Im Jahre 1997 schreib das NIST[2] offiziell die Suche nach einer symmetrischen Blockchiffre für die Verschlüsselung von sensiblen Daten aus. Eine symmetrische Blockchiffre arbeitet im Prinzip wie folgt:

- Zunächst wird ein geheimer Schlüssel gewählt
- Dann werden die zu verschlüsselnden Daten, der Klartext, in viele gleich große Blöcke unterteilt
- Jeder Block wird nach einem festen Algorithmus transformiert und mit dem Schlüssel kombiniert, so dass aus dem neuen transformierten Block, ohne die Kenntnis des Schlüssels, nicht mehr auf den Klartext geschlossen werden kann.
- Die so entstandenen verschlüsselten Blöcke bilden den Geheimtext. Er ist ebenso lang wie der Klartext, da sich die Blockgröße durch das Verschlüsseln nicht ändert.
- Mit dem inversen Algorithmus und dem Schlüssel kann aus dem Geheimtext wieder blockweise der Klartext errechnet werden.

[2] U.S. National Institute of Standards and Technology (www.nist.gov)

2.1 Anforderungen an den neuen Standard

Die Ausschreibung von AES wurde sehr sorgfällig vorbereitet. Die wichtigsten Anforderungen lassen sich in drei Kategorien aufteilen:

1. Sicherheit
    - Robustheit der Verschlüsselung gegenüber der Kryptoanalyse
    - Stichhaltigkeit der mathematischen Basis
    - Zufälligkeit der ausgegebenen Geheimtexte
    - Relative Sicherheit bezüglich der anderen AES-Kandidaten
2. Kosten:
    - Keine Patentansprüche der Entwickler
    - Ausreichende Geschwindigkeit auf verschiedenen Rechnerarchitekturen
    - Geringe Speicheranforderungen für die Verwendung in mobilen Geräten
3. Eigenschaften für die Implementierung
    - Eignung für verschiedenste Hard- und Softwaresysteme
    - Einfachheit des Verfahrens
    - Flexible Schlüssellänge – 128, 192 und 256 Bit
    - Blockgröße soll mindestens 128 bit betragen

2.2 Mathematische Grundlagen

Wie bereits erwähnt, basiert Rijndael auf Bytes und Worten. Die 8 Bit eines Bytes bilden eine Menge von 256 Elementen. In [DaeRij99] wird jedes Byte $a = a_7 \ldots a_0$ als Polynom dargestellt:

$$a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

Auf das Polynom wird die Addition ($\oplus$) und eine Multiplikation ($\bullet$) erklärt:

- Die Addition besteht aus einer einfachen Exklusiv-oder-Verknüpfung (XOR)[3] der einzelnen Koeffizienten. Technisch kann XOR[2] direkt auf ganze Bytes angewendet werden.

- Die Multiplikation ist wie das gewöhnliche Polynomprodukt unter der Verwendung obiger Addition definiert. Anschliessend wird aber das so entstandene Polynom, das statt Grad 7 nun Grad 14 haben kann, modulo einem festen Polynom m(x) gerechnet. Dabei ist

  $$m(x) = x^8 + x^4 + x^3 + x + 1$$

  als Polynom über GF(2) irreduzibel, kann also nur duch 1 und sich selbst geteilt werden. Eine solche Multiplikation

  $$(a(x) \bullet (b(x)) \bmod m(x)$$

  kann mit der schrittweisen Multiplikation $x \bullet b(x)$ leicht errechnet werden.

Mit den so definierten Operationen bilden die 256 möglichen Werte eines Bytes einen endlichen Körper $GF(2^8)$. Von diesem Körper ausgehend wird nun eine Addition (+) und eine Multiplikation ($\otimes$) auf Worten definiert. Es sei nun $a_j \in GF(2^8)$. Auf Worten $a = a_3a_2a_1a_0$, als Polynom $a_3x^3 + a_2x^2 + a_1x + a_0$, wird definiert:

- Die Addition mit der oben definierten Addition auf den Koeffizienten:

  $$(a_3x^3 + a_2x^2 + a_1x + a_0) + (b_3x^3 + b_2x^2 + b_1x + b_0) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

---

[3] Eine exklusiv-Oder-Verknüpfung ist ein Begriff aus der Aussagenlogik. Die Gesamtaussage ist wahr, wenn entweder die erste Aussage oder die zweite Aussage wahr ist, aber nicht beide. Praktisch entspricht das der Addition zweier Bits modulo 2.

XOR-Verknüpung zweier Bits:
0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0

- Die Multiplikation wie das gewöhnliche Polynomprodukt über $GF(2^8)$ mod $M(x) = x^4+1$. M(x) ist nicht irreduzibel, da:

  $(x^2 + 1)(x^2 + 1) = x^4 + (1 \oplus 1)x^2 + 1 = x^4 + 1 = M(x)$

  Prinzipiell ist also nicht jedes Polynom über $GF(2^8)$ modulo M(x) invertierbar. Das in Rijndael einzige verwendete Polynom $c(x)$ [$= (03)x^3 + (01)x^2 + (01)x + (02)$] zur Multiplikation (in MixColumn) ist aber invertierbar gewählt. Damit können Multiplikationen mit c(x) durch Multiplikation mit $c(x)^{-1}$ [$= (0b)x^3 + (0d)x^2 + (09)x + (0e)$] wieder rückgängig gemacht werden.

Da die Worte oft als Spalte von 4 Bytes dargestellt werden, bietet es sich an, von der Polynomschreibweise in eine senkrechte Vektorschreibweise zu wechseln. Eine Multiplikation $b(x) = x \otimes a(x)$ kann dann als Matrix-Vektor Multiplikation wie folgt dargestellt werden:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 0001 \\ 1000 \\ 0100 \\ 0010 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} a_3 \\ a_0 \\ a_1 \\ a_2 \end{pmatrix}$$

Offensichtlich handelt es sich dabei um eine zyklische Vertauschung der Koeffizienten von a(x). Davon ausgehend können beliebige $a(x) \otimes b(x)$ mit einer Matrix-Vektor-Multiplikation berechnet werden.

3 Spezifikation

Rijndael[4] ist ein symmetrischer Blockchiffre. Das heißt, mit ein- und demselben Schlüssel können Daten blockweise verschlüsselt und entschlüsselt werden. Dabei können Blockgröße und Schlüssellänge unabhängig voneinander 128, 192 oder sogar 256 Bit betragen. Schon eine Schlüssellänge von 128 Bit macht das Durchprobieren aller Schlüssel $2^{128-56} = 2^{72}$ mal aufwendiger als bei DES. Nach Moore`s Law[5] verdoppelt sich die Rechenleistung alle 18 Monate. Danach ist in 100 Jahren das Durchprobieren aller 128 Bit-Schlüssel bei AES ungefähr so zeitaufwändig wie beim DES heute. Werden gegen Rijndael keine brauchbaren Angriffsmöglichkeiten gefunden, so dürfte er mit Schlüsseln ≥ 192 Bit für die nächsten 100 Jahre sicher genug sein.

3.1 Die Struktur des AES Algorithmus

Der AES Algorithmus wird in Runden ausgeführt. Runden bedeutet in diesem Zusammenhang, das Teile des Algorithmus mehrfach ausgeführt werden. Die Zahl der durchgeführten Rundendurchläufe hängt dabei von der Schlüssellänge (in Bit) und der Blockgrösse (in Bit) ab.

	Blockgröße 128	Blockgröße 192	Blockgröße 256
Schlüssellänge = 128	10	12	14
Schlüssellänge = 192	12	12	14
Schlüssellänge = 256	14	14	14

Tab. 1: Rundenanzahl

AES basiert auf Bytes (8 Bit) und Worten (32 Bit). Im Folgenden wird ein Block[6], als 16, 24 oder 32 Byte, beziehungsweise 4, 6 oder 8 Worte, als Rechteck dargestellt. Ein Kästchen $a_{i,j}$ ist dabei ein Byte, eine Spalte $a_{0,j} \dots a_{3,j}$ ein Wort:

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

[4] Die Bezeichnung Rijndael und AES sind äqvivalend und bezeichen beide denselben Alogithmus

[5] 1965 von Gordon Moore (Intel-Mitbegründer) formuliertes Gesetz, nach dem sich ca. alle achtzehn Monate die Anzahl der Transistoren pro Chipfläche verzweifacht; d.h. Verdoppelung der Geschwindigkeit von Computerprozessoren bei gleichbleibendem Preis in diesem Zeitraum.

[6] Block bezeichnet hier auch die momentane Belegung der Bits mit den Werten 0 oder 1. Im Englischen wird dies state genannt.

Abb. 1: Byte vs. Block

Zunächst wird auf den Imput-Block (State) ein Rundenschlüssel (Roundkey) angewendet, der aus dem Schlüssel errechnet wird. Diese Transformation wird in AddRoundKey beschrieben. Nun beginnen die eigentlichen Runden. Jede Runde, außer der letzten, besteht aus vier Komponenten. Es folgt der Algorithmus der Verschlüsselung in PseudoCode:

```
Round (word State,  word Roundkey)
{
        ByteSub(State);
        ShiftRow(State);
        MixColumn(State);
        AddRoundKey(State, Roundkey);
}
FinalRound (word State,  word Roundkey)
{
        ByteSub(State);
        ShiftRow(State);
        AddRoundKey(State, Roundkey);
}
Rijndael (byte State, byte CipherKey)
{
        KeyExpansion (CipherKey, ExpandedKey);
        AddRoundKey(State, ExpandedKey);
        for (i = 1; I < Nr; i++)
                Round (State, ExpandedKey + Nb*i);
        FinalRound(State, ExpandedKey + Nb*Nr);
}
```

3.2 Die Transformationen ByteSub

ByteSub arbeitet byteweise auf einem gegebenen Block:

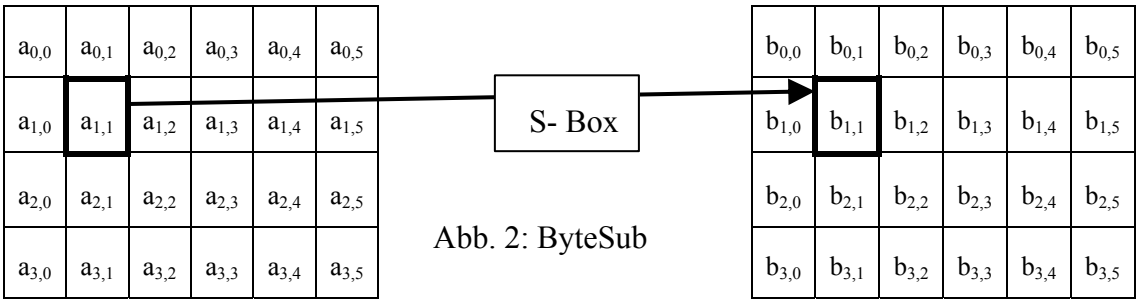

Abb. 2: ByteSub

Die ByteSub Transformation ist eine nicht lineare Byte Substitution, die auf jedes Byte eines Blocks unabhängig von anderen Bytes angewendet wird. Dabei wird eine Substitutions-Tabelle (S-Box) verwendet.

Für die S-Box gibt es eine explizite, nachvollziehbare Berechnungsvorschrift. Die S-Box wird durch Hintereinanderausführung der folgenden beiden (invertierbaren) Transformationen erzeugt:

Die 1. Transformation:

Ist a $=a_7a_6a_5a_4a_3a_2a_1a_0 \neq 0$ ein Byte, so wird zur dazugehörigen Polynomdarstellung

$$f(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \in GF(2^8) <=> GF(256)$$

das (bezüglich der Multiplikation) inverse Polynom bestimmt:

$$f^{-1}(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

Das Ergebnis der 1. Transformation sind dann die Koeffizienten von $f^{-1}(x)$, d.h.:

$$b = b_7b_6b_5b_4b_3b_2b_1b_0.$$

Das Nullbyte a = 0 wird von der 1. Transformation auf das Nullbyte b = 0 abgebildet.

Die 2. Transformation:

Auf $b = b_7b_6b_5b_4b_3b_2b_1b_0$ wird folgende affine Transformation angewandt:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Das Ergebnis lautet somit $c = c_7c_6c_5c_4c_3c_2c_1c_0$

Bemerkungen:

- Die 1. Transformation schützt AES gegen die differentielle und lineare Kryptoanalyse. Die Verknüpfung mit der 2. Transformation dient als Schutz gegen Interpolations-Attacken.
- Es gibt kein Byte a, für das S(a) = a oder S(a) = a^{-1} gilt (S bezeichnet die S-Box Abbildung).
- Die S-Box und damit die gesamte ByteSub Transformation sind invertierbar.

3.2.1 Die S-Box des AES

Die Hintereinanderausführung der beiden Transformationen der ByteSub Transformation ergibt folgende 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

(Quelle: Scan aus [Waet03] S. 233)

Abb.3: S-Box, Substitutionswerte für das Byte (xy)

Anwendung:

Das Byte xy (in Hexadezimaldarstellung) wird auf das Byte in der Zeile x und Spalte y abgebildet,

z.B. S(47) = A0 = bzw. S(0100 0111) [4 7] = 1010 0000 [A 0]

3.2 Die Transformationen ShiftRow

Bei ShiftRow werden die Zeilen eines Blockes zyklisch geschoben:

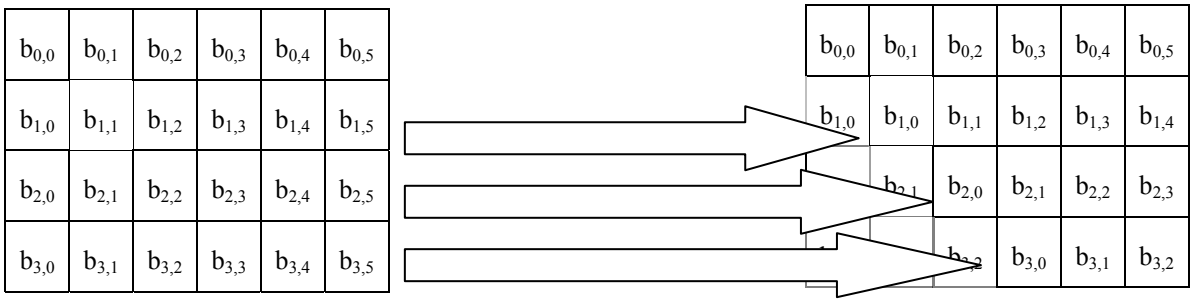

Abb.4: Darstellung der ShiftRow Transformation

Wie weit die Zeilen dabei geschoben werden, sind Parameter, die nur von der Blockgrösse abhängen. Bei 128 Bit Blocklänge wird die

- 2. Zeile um eine Position

- 3. Zeile um 2 Positionen

- 4. Zeile um 3 Positionen nach links geschoben.

Für 192 Bit arbeitet die ShiftRow Transformation genauso, bei 256 Bit wird dagegen in der 3. und 4. Zeile um je 1 Position mehr verschoben, d.h. in der 3. Zeile um 3 und in der 4. Zeile um 4 Positionen.

3.3 Die Transformationen MixColumn

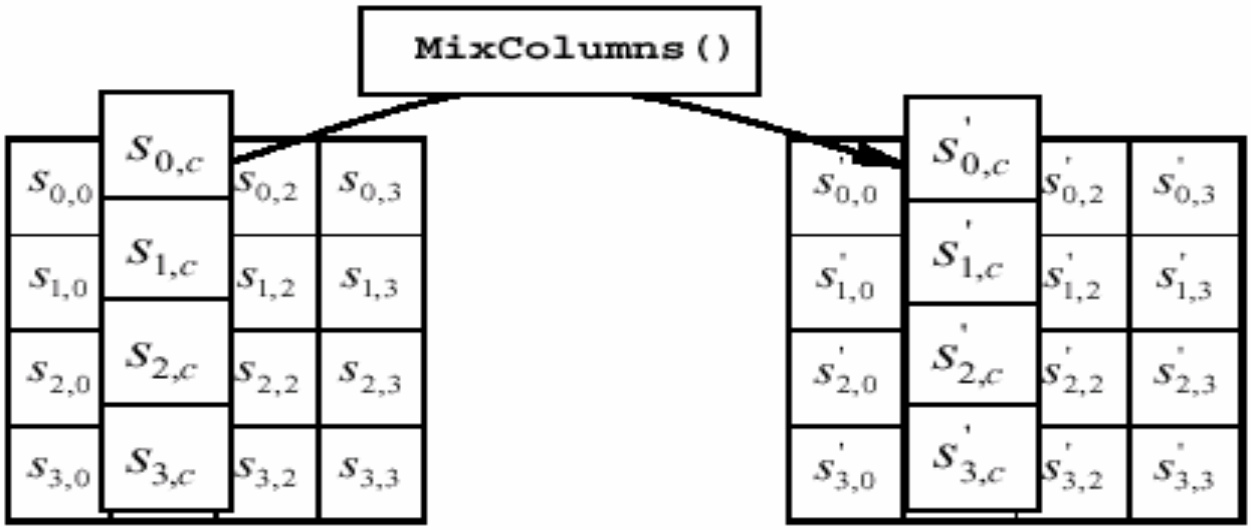

Abb.5: Darstellung der MixColumn Transformation

Schließlich werden die Spalten vermischt. Es wird zunächst jede Zelle einer Spalte mit einer Konstanten multipliziert und anschließend die Ergebnisse XOR verknüpft.

$$\begin{pmatrix} S`_{0,c} \\ S`_{1,c} \\ S`_{2,c} \\ S`_{3,c} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \bullet \begin{pmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{pmatrix}$$

Die Werte 01, 02 und 03 sind dabei ebenso wie $S_{0,c}$... S_3,c und $S`_{0,c}$... $S`_{3,c}$ als Bytes zu interpretieren, auf die die Addition $\oplus$ und die Multiplikation $\bullet$ angewandt wird, d.h.

$S`_{0,c} = (02 \bullet S_{0,c}) \oplus (03 \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$

$S`_{1,c} = S_{0,c} \oplus (02 \bullet S_{1,c}) \oplus (03 \bullet S_{2,c}) \oplus S_{3,c}$

...

3.4 Verschlüsselung mit AddRoundKey

In der Vorrunde und nach jeder weiteren Verschlüsselungsrunde wird AddRoundKey ausgeführt. Hierbei wird eine bitweise XOR-Verknüpfung zwischen dem Block und dem aktuellen Rundenschlüssel vorgenommen. Dies ist die einzige Funktion in AES, die den Algorithmus vom Benutzerschlüssel abhängig macht.

Die Abbildung AddRoundKey ist eine einfache Exklusiv-Oder-Verknüpfung des Blocks mit einem ebenso langen Rundenschlüssel.

Der AES-Schlüssel mit Nk (Nk stellt die Anzahl der 32-Bit-Wörter im Schlüssel dar) Worten (1 Wort = 4 Bytes) wird zu einem Schlüssel mit Nb (Anzahl der Spalten in einem Zustand) * (Nr [Anzahl der Runden] + 1) Worten w[i] erweitert (siehe 3.5 Berechnung des Rundenschlüssels)

Diesen erweiterten Schlüssel werden mit w[0] beginnend der Reihe nach die erforderlichen Nr +1 Rundenschlüssel für die AddRoundKey() Transformation entnommen. Im folgenden Schema betrachte ich aus Gründen der Übersichtlichkeit nur den spezifizierten Fall Nb = 4.

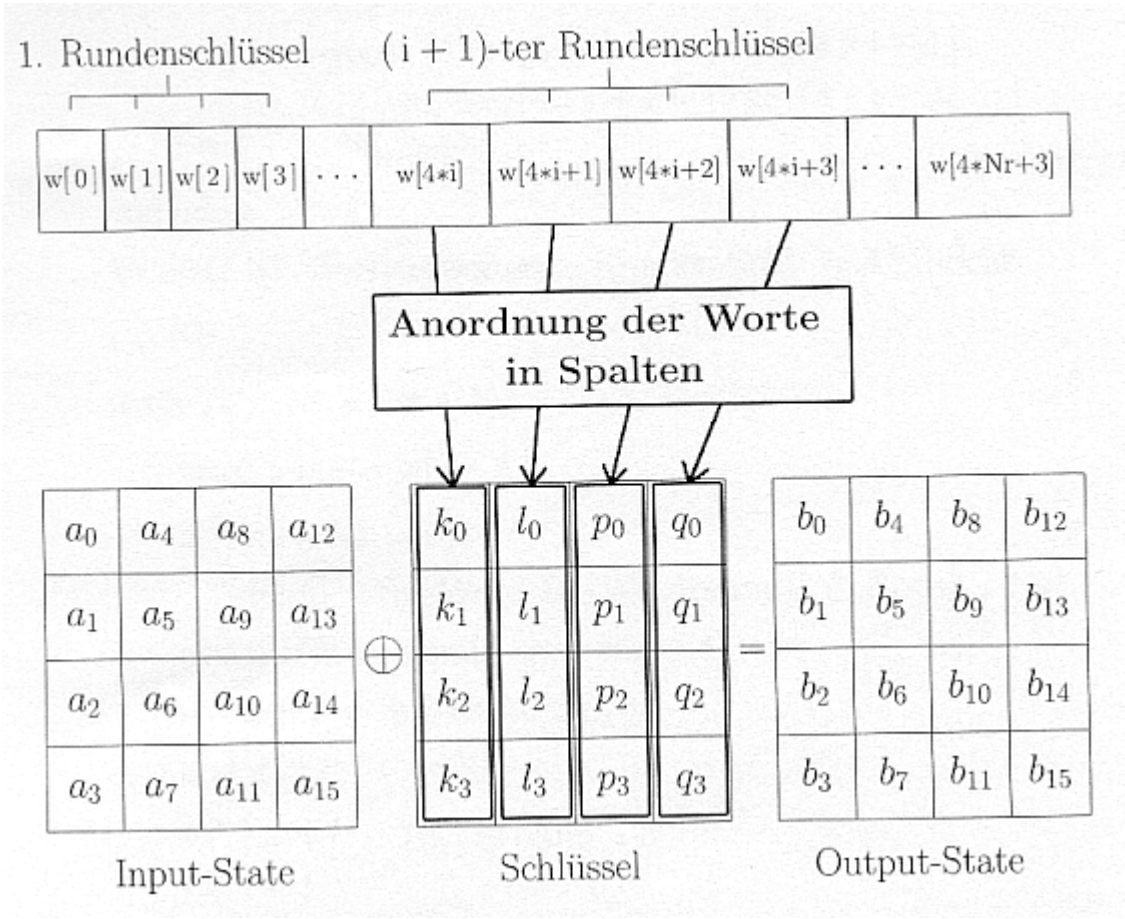

Abb. 6: Ablauf von „AddRoundKey“

Die XOR-Verknüpfung der AddRoundKey() Transformation erfolgt byteweise z.B:

$a_7 \oplus l_3 = b_7$ oder $a_{13} \oplus q_1 = b_{13}$.

Der genaue Ablauf findet sich in wenigen Zeichen Pseudocode in [DaeRij99].

3.5 Die Berechnung des Rundenschlüssels (Schlüssel-Erweiterung)

Die ersten Nk Worte w[0],. . . ,w[Nk-1] des erweiterten Schlüssels sind der originale AES-Schlüssel. Die weiteren Worte ergeben sich nach dem folgenden Schema, wobei Nk = 4, d.h. ein 128 Bit Schlüssel angenommen wird:

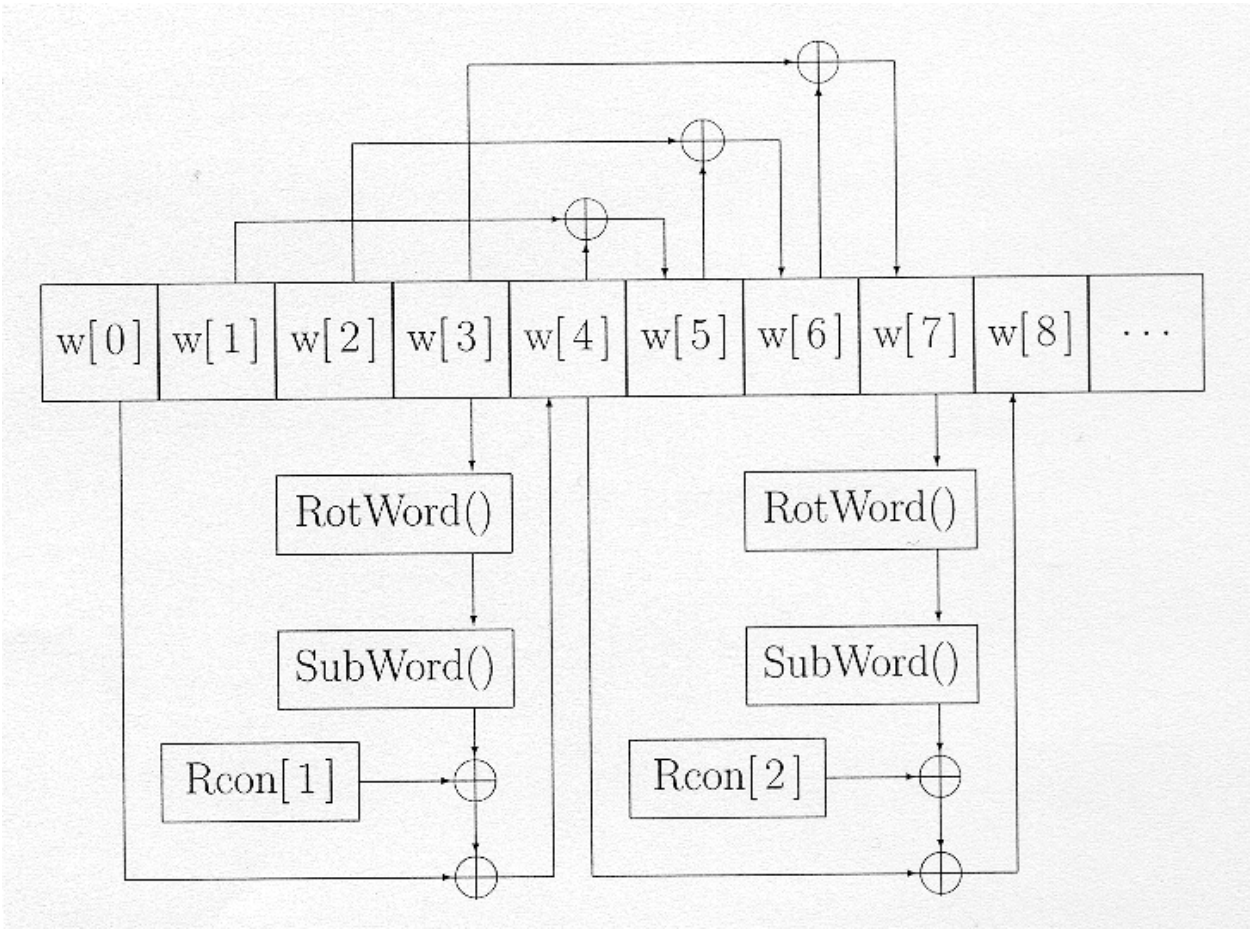

Abb.7: Schlüssel-Erweiterung

RotWord() verschiebt die 4 Bytes eines Wortes zyklisch um 1 Position nach links, d.h. RotWord($[a_0,a_1,a_2,a_3]$) = $[a_1,a_2,a_3,a_0]$.

SubWord() wendet auf jedes der 4 Bytes eines Wortes die S-Box Abbildung an.

Die i-te Rundenkonstante ist das Wort Rcon[i] = $[x^{i-1},0,0,0]$, wobei 0 das Nullbyte und x^{i-1} die (i-1)-te Potenz von $x=(02)_{hex}$ bzgl. Der Multiplikation • in GF (2^8) bezeichnet.

3.6 Die Entschlüsselung – der inverse Algorithmus

Die oben genannten vier Schritte in einer Runde von Rijndael lassen sich jeweils leicht invertieren:

• AddRoundKey ist zu sich selbst invers, da es ja nur XOR verwendet. Allerdings muss nun der Rundenschlüssel blockweise rückwärts verwendet werden.

• InvMixColumn funktioniert wie MixColumn, es muss lediglich das Polynom[7]:

$$c^{-1}(x) = 0B_{hex}x^3 + 0D_{hex}x^2 + 09_{hex}x + 0E_{hex}$$

verwendet werden.

• InvShiftRow verhält sich wie ShiftRow. Es wird lediglich in die andere Richtung geschoben.

• InvByteSub lässt sich direkt aus ByteSub konstruieren. Wiederum kann hierfür eine Tabelle mit 256 Paaren $(b, a) \in GF(2^8) \times GF(2^8)$ verwendet werden.

$c^{-1}(x)$ in InvMixColumn hat offensichtlich deutlich größere Koeffizienten als $c(x)$. Daher ist das Entschlüsseln von Nachrichten aufwändiger als das Verschlüsseln. Obige vier Schritte können zum Teil vertauscht werden. Zum Beispiel können InvByteSub und InvShiftRow vertauscht werden, da InvByteSub ein einzelnes Byte unabhängig von den anderen ändert und InvShiftRow jedes Byte für sich unverändert lässt. Wendet man InvMixColumn auch auf den Rundenschlüssel an, so kann man sogar InvMixColumn und AddRoundKey vertauschen. So erhält man für die Entschlüsselung die selbe Struktur wie bei der Verschlüsselung:

• Zunnächst einmal AddRoundKey(State, Roundkey).

• Dann Runden der Form:

```
InvRound(State, Roundkey)

{

        InvByteSub(State);

        InvShiftRow(State);

        InvMixColumn(State);

        AddRoundKey(State, InvMixColumn(Roundkey));

}
```

• Eine letzte Runde ohne InvMixColumn und mit dem gewöhnlichen AddRoundKey(State, Roundkey).

[7] Hier sind die Polynome von $GF(2^8)$ kexadezimal notiert

4.0 Kryptographische Stärke von AES

Bei nur 6 Runden könnte unter Verwendung von $6*2^{32}$ ausgewählen Klartextblöcken mittels 2^{44} komplexer Operation (d. h. ca. 17 Billionen) der Schlüssel berechnet werden. Praktisch heißt das: Etwa 400 GB vom Angreifer vorgegebener Klartext muss verschlüsselt und analysiert werden; wenn eine komplexe Operation eine Mikrosekunde dauert, braucht man dazu rund 200 Tage.

7 Runden; Hierzu sind fast 2^{128} ausgewähle Klartexte (entsprechend ca. $5*10^{39}$ Byte) und ein Rechenaufwand von 2^{120} notwendig; bei 1 Nanosekunde pro Operation ergäbe das $4*10^{19}$ Jahre (40 Trillionen).

Man beachte, wie sprunghaft die Sicherheit alleine durch das Hinzufügen einer 7. Runde wächst! Rijndael führt jedoch, wie obiger Tabelle entnommen werden kann, mindestens 10 Runden durch!

Wie bei allen praktischen kryptografischen Verfahren, kann die Sicherheit jedoch nicht bewiesen werden. Man kann nur testen, ob die bisher bekannten Angriffsmethoden der Kryptoanalyse versagen.

Beim Entwurf von Rijndael wurde auf alle bekannten Attacken, wie lineare und differenzielle Kryptoanalyse eingegangen, so das ein Angriff mit allen herkömmlichen Verfahren nicht effizienter sein sollte als ein Brute-Force-Angriff. Mit der Schlüssellänge von 128 Bit ergibt sich ein Schlüsselraum von $3{,}4 \times 10^{38}$, mit 192 Bit $6{,}2 \times 10^{57}$ und mit 256 Bit $1{,}1 \times 10^{77}$. Wenn man nun annimmt, das es eine Maschine gibt, die den ganzen DES Schlüsselraum in einer Sekunde durchsucht, dann würde es bei Rijndael ungefährt 149.000 Billionen Jahre dauern. Man bedenke, das es das Universum seit weniger als 20 Billionen Jahren gibt bzw. geben soll!

Man bedenke allerdings: Ähnlich unvorstellbar war das vor 20 Jahren auch noch für DES!

AES erwies sich als resistent gegen alle möglichen Angriffe. Implementierungen von Rijndael können im Vergleich zu den anderen Kandidaten mit dem geringsten Aufwand gegen Angriffe geschützt werden, die auf Messungen von Änderungen der Stromaufnahme beruhen (s.g. Power Analysis-Attacken).

Einfluss der Schlüssel:

Durch die Verknüpfung mit dem Rundenschlüssel von der ersten Runde und als letzter Schritt innerhalb einer Runde wirkt sich dieser auf jedes Bit der Rundenergebnisse aus. Es gibt im Verlauf der Verschlüsselung eines Blocks keinen Schritt, dessen Ergebnis nicht in jedem Bit vom Schlüssel abhängig wäre.

Nichtlineare Schicht:

Die S-Box Substitution ist eine stark nichtlineare Operation. Die Konstruktion der S-Box sorgt für nahezu idealen Schutz vor differentieller und linearer Krypoanalyse.

Lineare Schicht:

Die Shift Rows- und Mix Columns-Transformation sorgen für eine optimale Durchmischung der Bits eines Blocks.

4.1 Kryptoanalyse des AES

Im Mai 2001 gelang es Fergusion, Schroeppel und Whiting, den Algorithmus als geschlossene Formel darzustellen [FeScWh 2001] – allerdings mit 2^{50} Termen, also etwa einer Billiarde Summanden. Niemand weiß, ob daraus jemals ein sinnvoller Angriff auf AES konstruiert werden kann, doch bisher ließ sich kein anderes sicheres Verfahren in solch einer „einfachen" Form darstellen.

Ein Qualitätssprung war jedoch die Arbeit von Courtois und Pieprzyk. Die Mathematiker beschrieben ganze Klassen von Chiffrierungen mittels sehr grosser Systeme quatratischer Gleichungen, zum Beispiel 128-Bit-AES als System von 8000 Gleichen mit 1600 Variablen. Derartige Systeme lassen sich im Allgemeinen nicht mit vertretbarem Rechenaufwand lösen, doch in diesem Fall sind starke Vereinfachungen mittels der so genannten XSL-Methode möglich. Sie nutzt aus, dass die Gleichungssysteme mehr Gleichungen als Unbekannte enthalten (sie sind überbestimmt), die meisten Koeffizenten Null sind (schwach besetzt) und das sie eine besonders reguläre Struktur haben. Dank der Weiterentwicklung der XSL-Methode erscheint mittlerweile ein Angriff auf AES mit „lediglich" 2^{100} Rechenoperationen denkbar.

Diese Ergebnisse stellen einen Qualitätssprung in der Kryptoanalyse dar. Die bisherigen Methoden ließen Schlüssel ermitteln, allerdings nur mittels enormer Menge von Geheim- und meist auch Klartext. Der XSL-Angriff hingegen könnte die Rekonstruktionen des Schlüssels aus kleinen Mengen Geheimtext ermöglichen. Es überrascht auch, dass die Sicherheit eines Blockalgorithmus gegen diesen Angriff offenbar nicht mehr exponentiell mit der Rundenzahl steigt.

Aber trotzdem gilt AES nicht als unsicher. Viele Abschätzungen des Arbeitsaufwands basieren auf Vermutungen und auch 2^{100} Rechenschritte sind eine sehr große Zahl.

Die genannten Angriffe sind bislang also rein akademischer Natur und sind nicht in der Praxis umsetzbar!

5 Bewertung und Ausblick

Rijndael wurde vom NIST in einem sehr transparenten Prozess zum AES gewählt. Weltweit konnten sich viele Kryptographen vorab von der Sicherheit von Rijndael überzeugen. Zwar wurden in den vergangenen vier Jahren theoretische Schwächen bei Rijndael ausgemacht, doch bis heute ist kein brauchbarer Angriff gegen Rijndael bekannt.
Hauptkritikpunkt an Rijndael ist, dass sich der Algorithmus oder Teile davon sehr leicht in geschlossenen mathematischen Formeln darstellen lassen. Dies scheint eine notwendige Konsequenz der Einfachheit und Eleganz von Rijndael zu sein. Die Tatsache, dass ein Algorithmus in einem knappen Formelsystem beschrieben werden kann, sagt aber gar nichts darüber aus, wie schnell entsprechende Gleichungen gelöst werden können.
Rijndael/AES hat in den vergangenen Jahren sehr viel an Akzeptanz gewonnen und wird in vielen Soft- und Hardwareprodukten, zum Beispiel PGP, SSH, IBM zSeries 990, Microsoft .NET und IBM WebSphere J2EE, eingesetzt.
Meiner Meinung nach wurde mit Rijndael ein Verschlüsselungsalgorithmus gefunden, der den Namen AES zurecht verdient!

6 Literaturhinweise

[DaeRij99] Doemen, Joan; Rijmen, Vincent
AES Proposal: Rijndael, Proton World Int., Katholische Universität Leuven, 09/1999 (http://www.esat.kuleuven.ac.be/~rijmen/rijndael/)

[waet03] Wätjen, Dietmar
Kryptographie - Grundlagen, Algorithmen, Protokolle, Spectrum-Verlag, Berlin, 2003, ISBN 3-8274-1431-8, S. 225-240

[wob01] Wobst, Reinhard
Abenteuer Kryptographie – Methoden, Risiken und Nutzen der Datenverschlüsselung, Addison-Wesley-Verlag, München, 2001, ISBN 3-8273-1815-7, S. 230-237 und S. 117-181

[FeScWh 2001] Niels Ferguson; Richard Schroeppel; Doug Whiting
A simple algebraic representation of Rijndael. Counterpane Internet Security, Sandia National Laboratory, Hi/fn, Inc. 2001.

Web-Quellen:

Rijndael Specification:

http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael-ammended.pdf

Reinhard Wobst. AES unter Beschuss: http://www.heise.de/ct/02/21/038/

AES Beschreibung von Wikipedia.org: http://de.wikipedia.org/wiki/AES

Beschreibung von Kristof Hamann: http://www.korelstar.de/aes.php