Kapitel 6
Der Advanced Encryption Standard Rijndael

§1: Geschichte und Auswahlkriterien

DES wurde in Zusammenarbeit mit der National Security Agency der Vereinigten Staaten von IBM entwickelt und dann als amerikanischer Standard verkündet. Diese Vorgehensweise weckte von Anfang an den Verdacht, daß möglicherweise eine „Falltür“ eingebaut sei, insbesondere da zumindest ursprünglich nicht alle Design-Kriterien publiziert wurden.

Nachdem dreißig Jahre intensiver Kryptanalyse keine Falltür gefunden haben, müßte diese – so vorhanden – zumindest sehr gut versteckt sein; aber egal ob mit oder ohne Falltür: Wie wir im letzten Kapitel gesehen haben, erfüllt DES mit nur 56 Bit Schlüssellänge nicht mehr die heutigen Sicherheitsanforderungen. Ursprünglich war er ohnehin nur für eine Laufzeit von zehn Jahren vorgesehen und wurde nur immer wieder verlängert, weil die meisten Anwender von Kryptographie äußerst träge sind und sich nicht um Fortschritte der Kryptanalyse kümmern.

Obwohl es die Internationale Standardisierungsorganisation ISO ablehnt, ein Kryptoverfahren zu standardisieren (Ein Grund dafür ist die dann befürchtete Bündelung krimineller Energie auf dieses Verfahren), hat das das amerikanische Handelsministerium die Suche nach einem Nachfolgealgorithmus für DES am 2. Januar 1997 international ausgeschrieben; der vorläufige Name des Algorithmus war AES *(Advanced Encryption Standard)*. Federführend für die Auswahl war das *National Institute of Standards and Technology* (NIST) in Gaithersburg, Maryland.

Dieses ging von Anfang an davon aus, daß der zu wählende Algorithmus *stärker* sein müsse als Triple DES; er sollte zwanzig bis dreißig Jahre lang anwendbar sein und dementsprechende Sicherheit bieten. Nach einer internationalen Konferenz über die Auswahlkriterien am 15. April 1997 veröffentlichte es am 12. September 1997 die endgültige Ausschreibung.

Minimalanforderung an die einzureichenden Algorithmen waren danach, daß es sich um symmetrische Blockchiffren handeln muß, die mindestens eine Blocklänge von 128 Bit bei Schlüssellängen von 128 Bit, 192 Bit und 256 Bit vorsieht.

Als Kriterien für die Wahl zwischen den einzelnen Algorithmen wurden die folgenden Aspekte genannt:

1. **Sicherheit:** Wie sicher ist der Algorithmus im Vergleich zu den anderen Kandidaten? Inwieweit ist seine Ausgabe ununterscheidbar von der einer Zufallspermutation? Wie gut ist die mathematische Basis für die Sicherheit des Algorithmus begründet? (Im Gegensatz zu DES sollten dieses Mal alle Kriterien publiziert werden.)
2. **Kosten:** Welche Lizensgebühren werden fällig? Wie effizient geht der Algorithmus mit Rechenzeit und Speicherplatz um?
3. **Flexibilität:** Ein Algorithmus, der auf einer Vielzahl von Plattformen implementierbar ist (PCs, 8-Bit-Prozessoren, ATM-Netze, HDTV, ...) ist vorzuziehen, genauso einer, der auch als Stromchiffre, kryptographisch sichere Hash-Funktion oder ähnliches verwendet werden kann.
4. **Software:** Der Algorithmus *muß* auch softwaremäßig effizient implementierbar sein.
5. **Einfachheit:** Der Aufbau des Algorithmus soll möglichst einfach sein.

Nicht nur die Art der Suche unterschied sich also beträchtlich von der DES-Entwicklung hinter verschlossenen Türen, auch die Auswahlkriterien hatten sich geändert: DES war noch in erster Linie für Hardware entworfen; manche Aspekte wie etwa die für die kryptographische Sicherheit völlig irrelevante Anfangspermutation hatten wohl keinen anderen Sinn als die Verlangsamung von Software-Implementierungen.

Inzwischen hat freilich die Erfahrung mit der Kryptographie in der Geschäftswelt gezeigt, daß man dort den Kostenaufwand für Spezialhardware scheut und sich lieber mit den dubiosen Verfahren begnügt, die in der ohnehin vorhandenen Office-Software eingebaut ist (Preis für das Knacken durch spezialisierte Unternehmen je nach Programm zwischen 40 $ und 250 $), wohingegen sich die Hacker innerhalb und außerhalb der Geheimdienste durch einen hohen Aufwand nur wenig beeindrucken lassen.

Die Ausschreibung führte bis zum Abgabeschluß am 15. Juni 1998 zu fünfzehn Vorschlägen aus aller Welt. Diese wurden auf einer ersten AES-Konferenz vom 20.–22. August 1998 in Ventura, Kalifornien vorgestellt und der Fachwelt zur Analyse und Kommentierung empfohlen. Tatsächlich wurden vierzehn der fünfzehn Algorithmen schon vor der Konferenz veröffentlicht; lediglich der Algorithmus *Magenta* der Deutschen Telekom wurde erst am 20. August auf der Konferenz selbst bekanntgegeben. Er war auch der einzige Algorithmus, der bereits während der Konferenz geknackt wurde in einer auf den 20. August 1998 datierten Arbeit von E. BIHAM, A. BIRYUKOV, N. FERGUSON, L. KNUDSEN, B. SCHNEIER und A. SHAMIR, die noch auf der Konferenz verteilt wurde.

Die zweite AES-Konferenz am 22. und 23. April 1999 in Rom führte zu einer ersten Diskussion der Ergebnisse und Empfehlungen, welche der fünfzehn Algorithmen weiter betrachtet werden sollten. Die endgültige Entscheidung des NIST wurde am 9. August 1999 bekanntgegeben:

Bei fünf der eingereichten Algorithmen hatte sich herausgestellt, daß sie entweder mit einem Aufwand zu knacken sind, der deutlich unter der Durchsuchung des gesamten Schlüsselraums liegt (darunter natürlich auch Magenta) oder aber, daß es zu viele „schwache“ Schlüssel gibt. Da diese fünf Algorithmen auch mit die langsamsten Kandidaten waren, sprach nichts dafür, sie noch weiter zu betrachten.

Fünf weitere Kandidaten zeigten ebenfalls Schwächen, die zwar für sich allein betrachtet nicht so schwerwiegend waren, daß man die Verfahren eliminieren müßte; da diese fünf Kandidaten andererseits nirgends entscheidende Vorteile boten, wurden auch sie eliminiert, so daß noch fünf

Finalisten übrigblieben: Drei aus USA (MARS von IBM, RC6 von RSA und Twofish von Counterpane), Serpent von drei Kryptologen aus England, Israel und Norwegen, und Rijndael von zwei Kryptologen, die von der Katholischen Universität Leuven in Belgien kommen.

Bei der dritten AES-Konferenz am 13. und 14. April 2000 in New York bekam Rijndael 86 Stimmen, Serpent 59, Twofish 31, RC6 23 und MARS 13, so daß es nicht weiter verwunderte, daß das NIST am 2. Oktober 2000 Rijndael für den vorgeschlagenen Standard nominierte. Am 6. Dezember 2001 verkündete ihn dann das amerikanische Handelsministerium offiziell als Federal Information Processing Standard FIPS-197.

Rijndael hat seinen Namen von seinen beiden Autoren JOAN DAEMEN und VINCENT RIJMEN, die 1995 beziehungsweise 1997 an der Elektrotechnischen Fakultät der Katholischen Universität Leuven über Themen aus der Kryptographie promoviert hatten; RIJMEN arbeitet immer noch dort als Postdoc. Aufgrund der Wahl von Rijndael zum AES wurden sie als flämische Persönlichkeiten des Jahres 2000 gewählt.

Als Aussprachehilfe für Personen, die kein Niederländisch, Flämisch, Surinamer oder Afrikaans sprechen, geben die Autoren folgende englische Approximationen des Wortes „Rijndael“: „Reign Dahl“, „Rain Doll“ und „Rhine Dahl“.

§ 2: Algebraische Vorbereitungen

Alle Operationen von Rijndael sind algebraisch definiert. Es gibt einmal Operationen auf Byte-Ebene, die durch die Grundrechenarten im Körper mit 256 Elementen realisiert sind; dazu kommen Operationen auf 4-Byte-Wörtern, die im Polynomring über diesem Körper definiert sind. Als erstes müssen wir daher mit diesem Körper vertraut werden.

a) Euklidische Ringe

Für den EUKLIDischen Algorithmus im ersten Abschnitt waren im wesentlichen zwei Dinge wesentlich: Ersten mußten wir wissen, wann eine Zahl Teiler einer anderen ist, und zweitens brauchten wir eine Divison

mit Rest, für die der Rest in irgendeiner Weise kleiner als der Divisor ist, wobei jede Folge immer kleiner werdender Reste nach endlich vielen Schritten abbrechen muß. Diese beiden Eigenschaften werden im Begriff des EUKLIDischen Rings formalisiert:

Definition: *a)* Ein Ring ist eine Menge R zusammen mit zwei Verknüpfungen $+, \cdot\colon R \times R \to R$, für die gilt:

1.) $(R, +)$ ist eine abelsche Gruppe

2.) $a(bc) = (ab)c$ für alle $A, b, c \in R$

3.) $a(b + c) = ab + ac$ für alle $a, b, c \in R$.

b) R heißt *kommutativer Ring mit Eins,* falls zusätzlich gilt

4.) Es gibt ein Element $1 \in R$, so daß $1 \cdot a = a \cdot 1 = a$ für alle $a \in R$

5.) $a \cdot b = b \cdot a$ für alle $a, b \in R$.

c) Ein kommutativer Ring mit Eins heißt *Integritätsbereich,* wenn gilt:

6.) Für $a, b \in R \setminus \{0\}$ ist auch $ab \neq 0$.

d) Ein EUKLIDischer Ring ist ein Integritätsbereich R zusammen mit einer Abbildung $\nu\colon R \setminus \{0\} \to \mathbb{N}_0$, für die gilt:

7.) Zu je zwei Elementen $a, b \in R \setminus \{0\}$ gibt es $q, r \in R$ mit

$$a = bq + r \quad \text{und} \quad \nu(r) < \nu(b) \ \text{ falls } r \neq 0$$

und für jeden Teiler b von a gilt $\nu(b) \leq \nu(a)$.

Offensichtlich ist der Ring $\mathbb{Z}$ der ganzen Zahlen ein EUKLIDischer Ring; hier können wir einfach $\nu(a) = |a|$ setzen. Als Übung kann man auch leicht zeigen, daß der Ring

$$\Gamma = \mathbb{Z} \oplus \mathbb{Z}i = \left\{ a + bi \in \mathbb{C} \;\middle|\; a, b \in \mathbb{Z} \right\}$$

der GAUSSschen Zahlen EUKLIDisch ist mit

$$\nu(a + ib) = a^2 + b^2 \,.$$

Interessanter für uns ist, daß für jeden Körper k der Polynomring

$$k[X] = \left\{ \sum_{\ell=0}^{n} a_\ell X^\ell \;\middle|\; n \in \mathbb{N}_0 \ \text{ und } \ a_i \in k \right\}$$

ein EUKLIDischer Ring ist, wobei die Funktion ν hier einfach jedem Polynom seinen Grad zuordnet, also den Exponenten der höchsten vorkommenden X-Potenz. Hier zeigt die übliche Polynomdivison mit Rest, daß es in der Tat zu je zwei Polynomen f, g mit Koeffizienten aus k Polynome q, r gibt, so daß

$$f = gq + r \quad \text{und} \quad r = 0 \ \text{ oder } \ \deg r < \deg g \,.$$

Da EUKLIDische Ringe insbesondere Integritätsbereiche sind, können wir auch dort größte gemeinsame Teiler definieren:

Definition: *a)* Ein Element t eines Integritätsbereichs R heißt Teiler von $r \in R$, in Zeichen $t|r$, wenn es ein $q \in R$ gibt mit $r = qt$.
b) $d \in R$ heißt größter gemeinsamer Teiler von $r, s \in R$, wenn $d|r, s$ und wenn für jeden weiteren gemeinsamen Teiler t von r und s gilt: $t|d$.

Eine besondere Rolle spielen die Teiler der Eins: Für einen solchen Teiler t gibt es ein Element q mit $qt = 1$, d.h. t hat ein Inverses bezüglich der Multiplikation. Solche Elemente bezeichnen wir als *Einheiten;* offensichtlich bilden die Einheiten aus R eine multiplikative Gruppe, die sogenannte *Einheitengruppe* $R^\times$.

Man beachte, daß die so definierten größten gemeinsamen Teiler nicht eindeutig bestimmt sein müssen: In $\mathbb{Z}$ beispielsweise ist sowohl drei als auch minus drei ein größter gemeinsamer Teiler von sechs und minus neun. Allgemein gilt:

Lemma: In einem EUKLIDischen Ring R gibt es zu je zwei Elementen r, s, die nicht beide gleich null sind, einen größten gemeinsamen Teiler d. Dieser kann nach dem EUKLIDischen Algorithmus berechnet werden und läßt sich als Linearkombination

$$d = \alpha r + \beta s \quad \text{mit} \quad \alpha, \beta \in R$$

darstellen. Sind d und d' zwei größte gemeinsame Teiler von r und s, so gibt es eine Einheit e mit $d' = ed$.

Beweis: Die Existenz eines größten gemeinsamen Teilers folgt genau wie im Fall der natürlichen Zahlen: Ist $a = bq + r$, so ist ein Element

$t \in R$ genau dann gemeinsamer Teiler von a und b, wenn es gemeinsamer Teiler von b und r ist. Daher gibt es genau dann einen größten gemeinsamen Teiler von a und b, wenn es einen größten gemeinsamen Teiler von b und r gibt, und dieser ist dann gleichzeitig größter gemeinsamer Teiler von a und b.

Der EUKLIDische Algorithmus kann genauso definiert werden wie für ganze Zahlen, und genau wie dort muß er auch enden mit einem Divisionsrest null, denn die Folge der Zahlen $\nu(r_i) \in \mathbb{N}_0$ ist strikt fallend. Im Falle $r_n = 0$ ist $\mathrm{ggT}(r_{n-2}, r_{n-1}) = r_{n-1}$, da r_{n-1} dann r_{n-2} ohne Rest teilt; induktiv folgt, daß das auch ein ggT von a und b ist.

Sind d und d' zwei größte gemeinsame Teiler von a und b, so sind beide insbesondere gemeinsame Teiler von a und b; nach Definition eines größten gemeinsamen Teilers muß daher d Teiler von d' sein und umgekehrt. Es gibt somit ein $e \in R$ mit $d' = ed$ und ein $e' \in R$ mit $d = e'd'$. Also ist

$$d = e'd' = e'ed \Longrightarrow (1 - e'e)d = 0 \Longrightarrow 1 - e'e = 0 \Longrightarrow e'e = 1\,,$$

da R nach Voraussetzung ein Integritätsbereich ist. Die letzte Gleichung rechts zeigt, daß e eine Einheit ist.

Schließlich müssen wir noch zeigen, daß sich *jeder* größte gemeinsame Teiler d von a und b als Linearkombination von a und b schreiben läßt. Der erweiterte EUKLIDische Algorithmus liefert eine solche Darstellung für *einen* größten gemeinsamen Teiler

$$d' = \alpha a + \beta b\,;$$

sind e, e' die oben betrachtete Einheit zu d und d', so folgt

$$d = e'd' = (\alpha e')a + (\beta e')b\,,$$

wie behauptet. ∎

Als Beispiel wollen wir für $k = \mathbb{Q}$ den größten gemeinsamen Teiler der beiden Polynome

$$P = X^8 + X^6 - 3X^4 - 3X^3 + 8X^2 + 2X - 5$$

und

$$Q = 3X^6 + 5X^4 - 4X^2 - 9X + 21$$

berechnen: Division von P durch Q führt auf den Quotienten $X^2/3 - 2/9$ und Divisionsrest

$$R_2 = -\frac{5}{9}X^4 + \frac{1}{9}X^2 - \frac{1}{3}\,.$$

Division von Q durch R_2 ergibt

$$R_3 = -\frac{117}{25}X^2 - 9X + \frac{441}{25}\,,$$

bei der Division von R_2 durch R_3 bleibt Rest

$$R_4 = \frac{233150}{6591}X - \frac{102500}{2197}\,,$$

und bei der letzten Divison verbleibt als Rest der ggT

$$R_5 = \frac{1288744821}{543589225}\,.$$

Da beide Ausgangspolynome ganzzahlige Koeffizienten haben, erscheint ein ggT mit einem so großen Nenner seltsam. Wir wissen aber, daß „der“ größte gemeinsame Teiler nur bis auf Einheiten bestimmt ist, und im Polynomring über einem Körper sind alle von null verschiedenen Konstanten Einheiten. Der größte gemeinsame Teiler ist daher nur eindeutig bis auf Multiplikation mit einer nichtverschwindenden Konstanten; diese Konstante kann nach Belieben gewählt werden und wird meist so gewählt, daß das Ergebnis in irgendeinem Sinne einfach wird.

Auf das obige Beispiel angewendet heißt das, daß mit

$$R_5 = \frac{1288744821}{543589225}$$

auch eins ein ggT von A und B ist und man daher im allgemeinen sagen würde, „der“ ggT von A und B sei eins. Es ist ein wohlbekanntes (und umgehbares) Problem der Computeralgebra, daß der EUKLIDische Algorithmus diese einfache Lösung in einer so komplizierten Form liefert; da wir aber vor allem Polynome über endlichen Körpern benötigen, braucht uns das nicht weiter zu kümmern.

b) Endliche Körper von Primzahlpotenzordnung

Beim Beweis, daß die ganzen Zahlen modulo einer Primzahl einen Körper bilden, gab es nur einen nichttrivialen Schritt: die Existenz des

multiplikativen Inversen, die wir aus der linearen Kombinierbarkeit des ggT folgerten und daraus, daß der ggT einer Zahl mit einer Primzahl gleich eins ist, falls die Zahl kein Vielfaches der Primzahl ist.

Genauso wollen wir jetzt Körper definieren, indem wir Polynome über einem festen Körper k modulo einem vorgegebenen Polynom P betrachten: Für ein beliebiges Polynom A über k ist A mod P gleich dem Rest bei der Division von A durch P.

Falls A kleineren Grad als P hat, ist natürlich einfach A mod $P = A$; zum konkreten Rechnen können wir daher ausgehen vom Vektorraum V aller Polynome vom Grad höchstens d, wobei $d+1$ der Grad von P ist. Die Addition ist die gewöhnliche Addition von Polynomen, das Nullpolynom ist Neutralelement, und $-A$ ist invers zu A.

Das Produkt AB zweier Polynome $A, B \in V$ kann größeren Grad als d haben; wir setzen daher

$$A \odot B = AB \bmod P\,;$$

dies ist ein Polynom vom Grad höchstens d, und es ist klar, daß die so definierte Multiplikation kommutativ und assoziativ ist und das Distributivgesetz erfüllt. Das konstante Polynom 1 ist Neutralelement auch bezüglich dieser Multiplikation. Algebraisch gesehen identifizieren wir V also mit dem Faktorring $k[X]/(P)$.

Ein inverses Polynom zu A ist ein Polynom B, für das $A \odot B = 1$ ist, d.h.

$$AB = 1 + CP \quad \text{oder} \quad AB + CP = 1$$

für ein geeignetes Polynom C. Zu vorgegebenen Polynomen A und P gibt es solche Polynome B und C genau dann, wenn der ggT von A und P gleich eins ist; alsdann lassen sich B und C nach dem erweiterten EUKLIDischen Algorithmus berechnen.

Wenn wir möchten, daß jedes Polynom A, dessen Grad kleiner als deg P ist, ein Inverses hat, müssen wir sicherstellen, daß A und P immer teilerfremd sind; dies ist offensichtlich genau dann der Fall, wenn P keinen nichttrivialen Teiler hat, keinen Teiler also, dessen Grad größer als null und kleiner als deg P ist. Ein solches Polynom heißt *irreduzibel.*

Falls es ein irreduzibles Polynom P vom Grad n mit Koeffizienten aus k gibt, läßt sich der Vektorraum k^n also zu einem Körper machen, indem wir ein n-tupel $(a_0, \ldots, a_{n-1})$ mit dem Polynom

$$a_{n-1}X^{n-1} + a_{n-2}X^{n-2} + \cdots + a_1X + a_0$$

identifizieren und die Multiplikation als Multiplikation von Polynomen modulo P erklären.

Bekanntestes Beispiel ist $k = \mathbb{R}$: Für $n = 2$ gibt es irreduzible Polynome vom Grad n, beispielsweise das Polynom $P = X^2 + 1$. Da

$$\begin{aligned}(a_1X + a_0)(b_1X + b_0) &= a_1b_1X^2 + (a_0b_1 + a_1b_0)X + a_0b_0 \\ &\equiv (a_0b_1 + a_1b_0)X + (a_0b_0 - a_1b_1) \bmod X^2 + 1\end{aligned}$$

ist, folgt

$$(a_0, a_1) \odot (b_0, b_1) = (a_0b_0 - a_1b_1, a_0b_1 + a_1b_0)\,,$$

wir erhalten also den Körper der komplexen Zahlen. Weitere Beispiele über $\mathbb{R}$ gibt es nicht, denn ein irreduzibles reelles Polynom muß entweder Grad eins oder Grad zwei haben, und da jedes irreduzible quadratische Polynom zwei konjugiert komplexe Nullstellen hat, entstehen dabei immer die komplexen Zahlen – lediglich die Basis über $\mathbb{R}$ ändert sich.

Über endlichen Körpern ist die Situation etwas komplizierter: Hier gibt es für *jedes* n mindestens ein irreduzibles Polynom vom Grad n, allerdings gilt auch hier, daß zwei irreduzible Polynome desselben Grads auf den gleichen Körper führen. Eine kurze Beweisskizze ist unten angedeutet; einen vollständigen Beweis findet man in jedem Lehrbuch der Algebra.

Es gibt keinen einfachen Ausdruck für ein irreduzibles Polynom vom Grad n über einem vorgegebenen endlichen Körper k; in der Computeralgebra behilft man sich meist damit, daß man so lange zufällige Polynome vom Grad n erzeugt, bis man ein irreduzibles gefunden hat – ein Algorithmus, der sich in der Praxis als deutlich effizienter erweist als manch ein deterministischer Algorithmus zur Lösung von Standardproblemen.

Es gibt allerdings auch eine Alternative, die zumindest für kleine Körper durchaus anwendbar ist: Ist $k = \mathbb{F}_q$ ein endlicher Körper mit q Elementen, so ist $k \setminus \{0\}$ eine multiplikative Gruppe der Ordnung $q-1$. Da die Ordnung eines jeden Elements einer Gruppe Teiler der Gruppenordnung ist, folgt

$$x^{q-1} = 1 \quad \text{für alle } x \in \mathbb{F}_q \setminus \{0\}\,.$$

Dies gilt insbesondere für das Element X mod P, das $\mathbb{F}_q$ über $\mathbb{F}_p$ erzeugt, also ist P ein Teiler von $X^{q-1}-1$. Die möglichen Polynome P zur Erzeugung von $\mathbb{F}_{p^n}$ über $\mathbb{F}_p$ sind also genau die irreduziblen Teiler vom Grad n des Polynoms $X^{p^n}-1$. Die Computeralgebra stellt gerade für Polynome über endlichen Körpern effiziente Faktorisierungsalgorithmen zur Verfügung, so daß man diese Teiler noch für recht große Werte von p^n relativ schnell berechnen kann.

Als weitere Konsequenz aus obiger Formel kommt man auch zu einer neuen Interpretation des Körpers mit $q = p^n$ Elementen: Da alle q-1 Elemente von $\mathbb{F}_q \setminus \{0\}$ Nullstellen des Polynoms $x^{q-1}-1$ sind und dieses Polynom den Grad $q-1$ hat, handelt es sich hier um *alle* Nullstellen von $x^{q-1}-1$; in der Sprache der Algebra ist $\mathbb{F}_q$ daher der Zerfällungskörper des Polynoms $x^{q-1}-1$ über $\mathbb{F}_p$. Daraus folgt wegen der Eindeutigkeit des Zerfällungskörpers, daß alle Körper mit q Elementen isomorph sind.

c) Der Körper mit 256 Elementen

Für AES ist der Körper $\mathbb{F}_{256}$ von zentraler Bedeutung; da $256 = 2^8$ ist, handelt es sich hier um den Vektorraum $\mathbb{F}_2^8$. Die Addition ist sehr einfach: Da die Addition in $\mathbb{F}_2$ mit dem exklusiven Oder übereinstimmt, ist die Addition in $\mathbb{F}_{256}$ das bitweise exklusive Oder, eine Operation, die nicht nur in den gängigen CPUs, sondern auch in vielen Prozessoren für spezielle Anwendungen in der Signalverarbeitung als Grundoperation implementiert und somit sehr schnell ist.

Um eine Multiplikation zu definieren, brauchen wir ein irreduzibles Polynom vom Grad acht über $\mathbb{F}_2$. Da $\mathbb{F}_2$ ein sehr kleiner Körper und acht eine ziemlich kleine Zahl ist, hat zumindest ein Computer keinerlei Schwierigkeiten, alle diese Polynome zu bestimmen: Wie wir im vorigen Abschnitt gesehen haben, sind das genau die irreduziblen Faktoren vom

Grad acht in der Faktorisierung des Polynoms $X^{255}-1$ über $\mathbb{F}_2$. Faktorisierung von Polynomen über Körpern von Primzahlordnung ist einer der Grundalgorithmen der Computeralgebra und geht auch noch bei sehr viel komplizierteren Polynomen sehr schnell; hier zeigt das Ergebnis, daß es dreißig Faktoren vom Grad acht gibt. Diese führen zwar alle auf denselben Körper, aber das praktische Rechnen in diesem Körper hängt natürlich stark von der Wahl des Polynoms ab. Insbesondere wird die Geschwindigkeit umso höher, je weniger Terme das Polynom hat.

Dreizehn der dreißig Polynome bestehen aus sieben nichtverschwindenden Termen, die restlichen siebzehn aus fünf; Wir wählen natürlich eines der letzteren. Alle diese Polynome haben, wie jedes Polynom vom Grad acht über $\mathbb{F}_2$, den führenden Term X^8; danach folgen vier weitere Terme. Bei der Reduktion modulo einem solchen Polynom $P = X^8 + \mathit{Rest}$ benutzt man, daß dann

$$X^8 \equiv \mathit{Rest}, \qquad X^9 \equiv X \cdot \mathit{Rest}, \quad \ldots$$

ist; dies wird umso häufiger mehrfach angewandt werden müssen, je höheren Grad die Terme in *Rest* haben. Am effizientesten kann man also rechnen, wenn das Polynom *Rest* den kleinstmöglichen Grad hat, und wenn zudem auch noch die hinteren Terme von *Rest* möglichst kleinen Grad haben. Inspektion der siebzehn Polynome mit fünf Termen zeigt, daß das Polynom

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

in dieser Hinsicht optimal ist.

Die genaue Festlegung für das Rechnen in $\mathbb{F}_{256} = \mathbb{F}_2^8$ für die Zwecke von AES ist folgende: Wir schreiben ein Byte als $(a_7, a_6, \ldots, a_0)$ und identifizieren es mit dem Polynom

$$a_7X^7 + a_6X^6 + \cdots + a_1X + a_0\,;$$

das Byte 0000 0010 entspricht also X.

Der Einfachheit halber schreiben wir Bytes meist als zweiziffrige Hexadezimalzahlen: Im betrachteten Beispiel wäre das 02_{hex}, und das Byte $\text{A5}_{\text{hex}} = 1010\,0101$ entspricht dem Polynom $X^7 + X^5 + X^2 + 1$.

Man beachte, daß trotz dieser Schreibweise die Addition und Multiplikation in $\mathbb{F}_{256}$ natürlich nichts mit der Addition und Multiplikation von Hexadezimalzahlen zu tun haben. Zwar ist $01_{\text{hex}} + 02_{\text{hex}} = 03_{\text{hex}}$, aber $01_{\text{hex}} + 01_{\text{hex}} = 00_{\text{hex}}$ und $05_{\text{hex}} + 04_{\text{hex}} = 01_{\text{hex}}$.

Zur Berechnung von $A5_{\text{hex}} \odot 01_{\text{hex}}$ müssen wir das Polynom

$$(X^7 + X^5 + X^2 + 1) \cdot X = X^8 + X^6 + X^3 + X$$

berechnen und modulo $m(X)$ reduzieren. Da

$$X^8 \bmod m(X) = X^4 + X^3 + X^2 + 1$$

ist (in $\mathbb{F}_2$ ist $-1 = 1$), ist dies

$$(X^4 + X^3 + X^2 + 1) + (X^6 + X^3 + X) = X^6 + X^4 + X^2 + X + 1 \,.$$

Somit ist $A5_{\text{hex}} \odot 01_{\text{hex}} = 56_{\text{hex}}$.

Trotz der Wahl des optimalen Polynoms ist die Multiplikation also immer noch erheblich aufwendiger als die Addition.

§3: Spezifikation von Rijndael

a) Terminologie und Bezeichnungen

Rijndael arbeitet mit verschiedenen Blocklängen sowie auch verschiedenen Schlüssellängen: Beide können (unabhängig voneinander) alle durch 32 teilbaren Werte zwischen 128 und 256 annehmen. Wir schreiben die Blocklänge als $32N_b$ und die Schlüssellänge als $32N_k$; die Zahlen N_b und N_k liegen also jeweils zwischen vier und acht.

Der offizielle FIPS-Standard AES ist nicht wirklich *gleich* Rijndael; für AES ist nur die eine Blocklänge 128 normiert und nur die drei Schlüssellängen $128, 196$ und 256; hier ist also stets $N_b = 4$ und N_k kann die drei Werte $4, 6, 8$ annehmen.

Rijndael verschlüsselt somit einen Block von $32N_b$ Bit oder $4N_b$ Bytes, und genau wie DES geschieht dies sukzessive in mehreren Runden. In der Terminologie von Rijndael wird der Block in seinem jeweiligen Bearbeitungsstand als „Zustand“ bezeichnet; die einzelnen Runden finden also jeweils einen bestimmten Zustand vor und verändern diesen.

Die Anzahl der Runden wird als N_r bezeichnet; sie hängt von N_b und N_k ab gemäß der Gleichung

$$N_r = \max(N_b, N_k) + 6 \,.$$

b) Die Grundoperationen

Auch wenn Rijndael nicht mehr nach dem Prinzip eines FEISTEL-Netzwerks arbeitet, sind in den einzelnen Runden immer noch die klassischen SHANNONschen Prinzipien von Konfusion und Diffusion realisiert.

Für die Konfusion sind bei DES die verschiedenen S-Boxen zuständig; bei Rijndael gibt es nur eine einzige Funktion zu diesem Zweck; diese ist definiert als Hintereinanderausführung der Bildung des (multiplikativen) Inversen in $\mathbb{F}_{256}$ mit einer über $\mathbb{F}_2$ affinen Abbildung von $\mathbb{F}_{256}$ nach $\mathbb{F}_{256}$.

Die Inversenbildung ist natürlich nur für $\mathbb{F}_{256} \setminus \{0\}$ erklärt; um trotzdem eine bijektive Abbildung von $\mathbb{F}_{256}$ nach $\mathbb{F}_{256}$ zu bekommen, nehmen wir die einzig mögliche Fortsetzung, bilden die Null also auf sich selbst ab. Auch wenn es algebraisch kompletter Unsinn ist, wollen wir zur Vermeidung von Fallunterscheidungen für die Definition von Rijndael die Kurzschreibweise $0^{-1} = 0$ verwenden mit der Interpretation, daß $x \mapsto x^{-1}$ die Fortsetzung der Inversenbildung zu einer bijektiven Abbildung von $\mathbb{F}_{256}$ nach $\mathbb{F}_{256}$ sein soll.

Diese Abbildung ist weder über $\mathbb{F}_{256}$ noch über $\mathbb{F}_2$ linear, und sie ist das einzige nichtlineare Element von Rijndael. Rechnerisch ist die Bestimmung von x^{-1} aufwendig, allerdings gibt es nur 256 mögliche Werte für x, die man vorausberechnen und in 256 Byte abspeichern kann; dieser Speicherbedarf dürfte selbst für Smartcards problemlos sein.

Diffusion wird durch eine Reihe von $\mathbb{F}_{256}$-linearen Abbildungen erzielt, die Operationen sind also im Gegensatz zu den Bit-Permutationen von DES allesamt auf Byte-Niveau definiert. Die Multiplikation von $\mathbb{F}_{256}$ sorgt dafür, daß trotzdem auch eine beträchtliche Diffusion innerhalb der Bytes stattfindet. Durch Tabellen gegebene Permutationen gibt es in AES überhaupt nicht; alle Diffusion wird durch Shift-Operationen sowie durch eine algebraische Operation realisiert.

Letztere arbeitet mit Wörtern von vier Byte, die wiederum mit Polynomen identifiziert werden, jetzt aber nicht, wie bei der Definition der Multiplikation in $\mathbb{F}_{256}$, mit Polynomen über $\mathbb{F}_2$, sondern solchen über $\mathbb{F}_{256}$. Wir schreiben die neuen Polynome daher zur besseren Unterscheidung in einer neuen Variablen Y und identifizieren $\mathbb{F}_{256}^4$ somit mit dem Vektorraum aller Polynome vom Grad höchstens drei in Y mit Koeffizienten aus $\mathbb{F}_{256}$, indem wir das Quadrupel (b_0, b_1, b_2, b_3) auffassen als Polynom

$$b_3Y^3 + b_2Y^2 + b_1Y + b_0 \quad \text{mit} \quad b_i \in \mathbb{F}_{256} .$$

Diese Polynome multiplizieren wir modulo dem Polynom $M = Y^4 + 1$.

Man beachte, daß dieses Polynom über $\mathbb{F}_2$ (und erst recht über $\mathbb{F}_{256}$) nicht irreduzibel ist: Da alle Binomialkoeffizienten außer dem ersten und dem letzten gerade sind, ist

$$(Y + 1)^4 = Y^4 + 1 \bmod 2 .$$

Mit der hier definierten Multiplikation wird $\mathbb{F}_{256}^4$ also nicht zu einem Körper. Rijndael verwendet diese Multiplikation allerdings nur für eine einzige lineare Abbildung von $\mathbb{F}_{256}^4$ nach $\mathbb{F}_{256}^4$, und diese ist gegeben durch Multiplikation mit dem Polynom

$$C = 03_{\text{hex}}Y^3 + Y^2 + Y + 02_{\text{hex}} .$$

An der Stelle $Y = 1$ ist

$$C = 03_{\text{hex}} + 1 + 1 + 02_{\text{hex}} = 03_{\text{hex}} + 02_{\text{hex}} = 1 ,$$

das Polynom ist also nicht durch $Y + 1$ teilbar (zur Erinnerung: über $\mathbb{F}_2$ wie auch $\mathbb{F}_{256}$ ist $Y + 1 = Y - 1$), und damit auch teilerfremd zu $Y^4 + 1 = (Y + 1)^4$. Damit hat zumindest dieses Polynom ein multiplikatives Inverses, das man mit dem erweiterten EUKLIDischen Algorithmus über $\mathbb{F}_{256}$ berechnen kann – auch wenn das von Hand ziemlich aufwendig ist. Im Maple-*worksheet* zu diesem Paragraphen findet man die detaillierte Rechnung mit Unterprogrammen für die Rechenoperationen von $\mathbb{F}_{256}$; als Ergebnis findet man dort das inverse Polynom

$$C' = 0B_{\text{hex}}Y^3 + 0D_{\text{hex}}Y^2 + 09_{\text{hex}}Y + 0E_{\text{hex}} .$$

Wer will, kann nachrechnen, daß $C \cdot C'$ modulo M gleich eins ist, allerdings erfordert bereits das recht viele Multiplikationen in $\mathbb{F}_{256}$.

c) Der Aufbau der Runden

Nach diesen Vorbereitungen können wir uns mit dem Aufbau der einzelnen Runden beschäftigen. Abgesehen von einer leichten Modifikation bei der letzten Runde besteht jede Runde aus denselben vier Schritten

1. Bytesubstitution
2. Zeilenshift
3. Spaltenmix
4. Addition des Rundenschlüssels

1.) Die Bytesubstitution: Dies ist der Konfusionsschritt; er operiert auf Bytes und wird auf jedes einzelne Byte des Zustands in derselben Weise angewendet; mit geeigneter Hardware kann dies also auch parallel erfolgen. Da es nur 256 mögliche Byte gibt, wird man diese Operation im allgemeinen über eine Tabelle implementieren; der Speicherbedarf von 256 Byte sollte auch auf einer Smartcard im allgemeinen problemlos sein,

Der erste Schritt ist, wie bereits erwähnt, die Inversenbildung im Körper $\mathbb{F}_{256}$; danach folgt eine Diffusion auf Bitniveau, indem $\mathbb{F}_{256}$ als affiner Raum $\mathbb{F}_2^8$ interpretiert wird und die affine Abbildung

$$\vec{x} \mapsto M\vec{x} + \vec{b}$$

mit

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{und} \quad \vec{b} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

angewandt wird. Insgesamt führt die Bytesubstitution ein Byte x also über in $Mx^{-1} + \vec{b}$.

Betrachtet man $\mathbb{F}_{256}$ nicht als affinen Raum, sondern als Raum der Polynome vom Grad höchstens sieben, kann man die affine Abbildung

auch interpretieren als

$$P \mapsto (X^7 + X^6 + X^5 + X^4 + 1)P + (X^7 + X^6 + X^2 + X) \bmod X^8 + 1\,;$$

da $X^7 + X^6 + X^5 + X^4 + 1$ an der Stelle eins den Wert eins annimmt, ist dieses Polynom teilerfremd zu $X^8 + 1 = (X + 1)^8$, die Abbildung und damit die Matrix M sind also invertierbar – was sie natürlich auch sein müssen, damit die Chiffre entschlüsselbar ist.

Als dritte Alternative kann man die Abbildung auch durch ein Polynom über $\mathbb{F}_{256}$ beschreiben, denn da der Körper endlich ist findet sich zu *jeder* Abbildung $\mathbb{F}_{256} \to \mathbb{F}_{256}$ ein Interpolationspolynom, das sie beschreibt. Im vorliegenden Fall ist dies das Polynom

$$\begin{aligned} &63_{\text{hex}} + 05_{\text{hex}} Z \quad + 09_{\text{hex}} Z^2 \; + \text{F9}_{\text{hex}} Z^4 \; + 25_{\text{hex}} Z^8 \\ &\quad + \text{F4}_{\text{hex}} Z^{16} + 01_{\text{hex}} Z^{32} + \text{B5}_{\text{hex}} Z^{64} + 8\text{F}_{\text{hex}} Z^{128}\,. \end{aligned}$$

Diese Abbildung wird angewandt auf das multiplikative Inverse eines Elements von $\mathbb{F}_{256}$. Im Körper $\mathbb{F}_{256}$ ist für jedes Element $z \neq 0$

$$z^{255} = 1 \quad \text{und} \quad (z^{-1})^n = z^{-n} = z^{255-n}\,,$$

wobei letztere Gleichung wegen unserer Konvention $0^{-1} = 0$ für *alle* z gilt. Damit können wir die Bytesubstitution insgesamt auch beschreiben durch das Polynom

$$\begin{aligned} &63 + 05_{\text{hex}} Z^{254} + 09_{\text{hex}} Z^{253} + \text{F9}_{\text{hex}} Z^{251} + 25_{\text{hex}} Z^{247} \\ &\quad + \text{F4}_{\text{hex}} Z^{239} + 01_{\text{hex}} Z^{223} + \text{B5}_{\text{hex}} Z^{191} + 8\text{F}_{\text{hex}} Z^{127}\,. \end{aligned}$$

Die Umkehrabbildung der Bytesubstitution hat als affine Abbildung die Form

$$\vec{y} \mapsto M^{-1}\vec{y} + M^{-1}\vec{b}\,;$$

dabei ist

$$M^{-1} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad \text{und} \quad M^{-1}\vec{b} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}\,.$$

Die Inversenabbildung ist natürlich (auch mit der Konvention, daß die Null auf sich selbst abgebildet wird) zu sich selbst invers; die Bytesubstitution wird also rückgängig gemacht, indem man ein Byte $\vec{y}$ zunächst auf $A^{-1}\vec{y} + A^{-1}\vec{b}$ abbildet und dann die erweiterte Inversenabbildung von $\mathbb{F}_{256}$ anwendet.

2.) Die Zeilenshifts: Der Diffusionsschritt ist zweigeteilt: Der zu verschlüsselnde Block ist ein Vektor aus $4N_b$ Bytes; er wird umgeschrieben in eine Byte-Matrix mit vier Zeilen und N_b-Spalten, die spaltenweise aufgefüllt wird. Wird die Matrix als $A = (a_{ij})$ bezeichnet, wobei der Zeilenindex i von 0 bis 3 und der Spaltenindex j von 0 bis $N_b - 1$ geht, ist der ursprüngliche Vektor also (in Zeilenschreibweise)

$$(a_{00}, a_{10}, a_{20}, a_{30}, a_{01}, a_{11}, \ldots, a_{N_b-2,3}, a_{N_b-1,0}, \ldots, a_{N_b-1,3})\,.$$

Indiziert man die Komponenten des Vektors durch einen Index n linear von 0 bis $4N_b - 1$, ist also

$$i = n \bmod 4, \quad j = \left[\frac{n}{4}\right] \quad \text{und} \quad n = i + 4j\,.$$

Der erste Diffusionsschritt ist eine zyklische Verschiebung der Zeilen von A: Die 0-te Zeile wird überhaupt nicht verschoben und die erste um eine Stelle; die Richtung der Verschiebung ist, wie auch stets im folgenden, nach links. Bei den beiden unteren Reihen hängt die Weite der Verschiebung von N_b ab: Für $N_b \neq 8$ wird die zweite Zeile um zwei Stellen verschoben, für $N_b = 8$ und drei. Die dritte Zeile wird für $N_b \leq 6$ um drei Stellen, für $N_b \leq 7$ um drei Stellen verschoben. Mit entsprechender Hardware können die Verschiebungen der einzelnen Zeilen natürlich parallel durchgeführt werden.

3.) Der Spaltenmix: Auf Spaltenebene ist die Diffusion aufwendiger: Ein Spaltenvektor ist ein Element von $\mathbb{F}_{256}^4$. Diesen Vektorraum hatten wir oben mit den kubischen Polynomen über $\mathbb{F}_{256}$ identifiziert und dort eine Multiplikation eingeführt über die Polynommultiplikation modulo Y^4+1. Genau dies verwendet der Spaltenmix, indem jeder Spaltenvektor mit dem Polynom

$$C = 03_{\text{hex}} Y^3 + Y^2 + Y + 02_{\text{hex}}$$

multipliziert wird. Wegen der einfachen Struktur des Polynoms $Y^4 + 1$ ist dies eine sehr einfache lineare Abbildung mit Matrix

$$\begin{pmatrix} 02_{\text{hex}} & 03_{\text{hex}} & 01_{\text{hex}} & 01_{\text{hex}} \\ 01_{\text{hex}} & 02_{\text{hex}} & 03_{\text{hex}} & 01_{\text{hex}} \\ 01_{\text{hex}} & 01_{\text{hex}} & 02_{\text{hex}} & 03_{\text{hex}} \\ 03_{\text{hex}} & 01_{\text{hex}} & 01_{\text{hex}} & 02_{\text{hex}} \end{pmatrix}$$

bezüglich der Basis $\{1, Y, Y^2, Y^3\}$. Diese kann für die N_b Spalten parallel durchgeführt werden.

Im Vergleich zum Zeilenshift ist diese Operation relativ aufwendig, da mehrere Multiplikationen in $\mathbb{F}_{256}$ durchgeführt werden müssen. In der letzten Runde, in der dieser Diffusionsschritt keiner anschließenden Konfusion mehr unterworfen wird und somit keinen großen Sicherheitsgewinn mehr bietet, wird daher auf diesen Schritt verzichtet.

Wie wir bereits gesehen haben, ist C modulo $Y^4 + 1$ invertierbar mit inversem Polynom

$$C' = 0\text{B}_{\text{hex}} Y^3 + 0\text{D}_{\text{hex}} Y^2 + 09_{\text{hex}} Y + 0\text{E}_{\text{hex}} ;$$

Multiplikation mit C' ist als lineare Abbildung gegeben durch die Matrix

$$\begin{pmatrix} 0\text{E}_{\text{hex}} & 0\text{B}_{\text{hex}} & 0\text{D}_{\text{hex}} & 09_{\text{hex}} \\ 09_{\text{hex}} & 0\text{E}_{\text{hex}} & 0\text{B}_{\text{hex}} & 0\text{D}_{\text{hex}} \\ 0\text{D}_{\text{hex}} & 09_{\text{hex}} & 0\text{E}_{\text{hex}} & 0\text{B}_{\text{hex}} \\ 0\text{B}_{\text{hex}} & 0\text{D}_{\text{hex}} & 09_{\text{hex}} & 0\text{E}_{\text{hex}} \end{pmatrix} ;$$

als Übung für das Rechnen in $\mathbb{F}_{256}$ sollte man zumindest für einige Einträge nachrechnen, daß das Produkt dieser beiden Matrizen über $\mathbb{F}_{256}$ gleich der Einheitsmatrix ist.

4.) Schlüsselexpansion und Rundenschlüssel: Die bisherigen drei Schritte sorgten für Konfusion und Diffusion; sie sind aber für jeden, der das grundsätzliche Verfahren kennt, leicht rückgängig zu machen. Das ist nicht weiter verwunderlich, denn bislang haben wir ja den *Schlüssel* noch nicht ins Spiel gebracht, an dem bei einem normierten Standard wie AES die ganze Sicherheit des Verfahrens hängt.

Der vierte Schritt jeder Runde ist genau wie bei DES eine einfache Addition des Rundenschlüssels; die Addition ist dabei die gewöhnliche

Vektoraddition in $\mathbb{F}_2^{32N_n}$, also das logische XOR. Sicherheitsrelevant ist, genau wie bei DES, die Berechnung der Rundenschlüssel aus dem vorgegebenen Schlüssel; der Schlüssel mit seinen $4N_k$ Bytes muß so aufgebläht werden auf $N_r \times 4N_b$ Schlüsselbytes, daß ein Kryptanalytiker aus einem oder auch einigen irgendwie ermittelten Rundenschlüsseln nicht auf den Gesamtschlüssel schließen kann.

Rijndael verwendet dazu im wesentlichen dieselben Techniken wie bei den Verschlüsselungsschritten: Das erweiterte Schlüsselfeld wird aufgefaßt als eine Folge von Wörtern W_i mit einer Länge von vier Byte oder 32 Bit. Die ersten Wörter W_0 bis W_{N_k-1} sind der eigentliche Schlüssel des Verfahrens, der Rest wird rekursiv daraus berechnet. Da die ersten N_b Worte nicht als Rundenschlüssel verwendet werden, sondern vor der ersten Runde zum Klartext addiert werden, hat das gesamte Schlüsselfeld eine Länge von $N_b(N_r + 1)$ Worten.

Für $i \geq N_k$ wird W_i sukzessive aus seinem unmittelbaren Vorgänger W_{i-1} und dem N_k Wörter zurückliegenden Vorgänger W_{i-N_k} berechnet:

- Falls i nicht durch N_k teilbar ist und im Falle $N_k > 6$ auch nicht durch vier, ist $W_i = W_{i-N_k} \oplus W_{i-1}$, wobei $\oplus$ die Vektorraumaddition in $\mathbb{F}_2^{32} = \mathbb{F}_{256}^8$ bezeichnet, also das logische XOR für 32-Bit-Wörter.
- Falls i durch N_k teilbar ist, wird W_{i-1} zunächst zyklisch um eins nach links verschoben. Dann wird auf jedes Byte des so entstandenen Worts die Bytesubstitution aus Absatz *1)* angewendet, und das Ergebnis wird mit $\oplus$ zu W_{i-N_k} addiert. Dazu wird noch eine Rundenkonstante addiert, deren erstes Byte dasjenige Element von $\mathbb{F}_{256}$ ist, das der Potenz X^{i/N_k} modulo dem die Multiplikation definierenden Polynom entspricht und dessen weitere drei Bytes alle Null sind.
- Falls $N_k > 6$ und $i \equiv 4 \bmod N_k$, wird die Bytesubstitution auf W_{i-1} angewendet und das Ergebnis zu W_{i-N_k} addiert.

d) Gesamtablauf von Rijndael

Nachdem nun alle Einzelheiten spezifiziert sind, kann der Gesamtablauf

von Rijndael leicht angegeben werden:

1. Die ersten N_b Worte des Schlüsselfelds werden zum Klartext addiert.
2. $N_r - 1$ Runden werden gemäß obiger Beschreibung durchgeführt.
3. Die N_r-te Runde wird entsprechend ausgeführt, aber ohne den Spaltenmix.

e) Geschwindigkeitsoptimierung

So wie Rijndael spezifiziert ist, ist vor allem die Bytesubstitution sehr langsam; aber wie bereits erwähnt, läßt sie sich leicht über eine Tabelle implementieren.

Ein weiterer rechnerisch aufwendiger Bestandteil von Rijndael sind die Multiplikationen im Körper $\mathbb{F}_{256}$. Eine vorausberechnete Multiplikationstabelle würde $256 \times 256 = 64 \times 1024$ Byte oder 64 Kilobyte in Anspruch nehmen, was erstens etwas viel und zweitens auch noch optimal ist: Die Multiplikation von $\mathbb{F}_{256}$ wird nur beim Spaltenmix benötigt, und da das hierzu verwendete Polynom C nur $01_{\text{hex}}, 02_{\text{hex}}$ und 03_{hex} als Koeffizienten hat, reicht es für die Verschlüsselung, wenn man die Produkte mit 02_{hex} und 03_{hex} bilden kann. Der jeweils andere Faktor des Produkts ist stets das Ergebnis einer Bytesubstitution, man spart also Rechenzeit, wenn man für jedes Byte das Ergebnis der Bytesubstitution sowie dessen Produkt mit 02_{hex} und mit 03_{hex} abspeichert; mit 768 Byte für Tabellen kann man also zur Laufzeit auf alle Multiplikationen und Divisionen in $\mathbb{F}_{256}$ verzichten.

Mit vier Kilobyte Tabellenplatz läßt sich sogar die gesamte Rundentransformation auf $4N_b$ XOR-Operationen auf 32-Bit-Wörtern zurückführen: Man speichere für jedes Byte $a \in \mathbb{F}_{256}$ die vier 32-Bit-Wörter

$$T_0(a) = \begin{pmatrix} 02_{\text{hex}} \odot S(a) \\ S(a) \\ S(a) \\ 03_{\text{hex}} \odot S(a) \end{pmatrix}, \quad T_1(a) = \begin{pmatrix} 03_{\text{hex}} \odot S(a) \\ 02_{\text{hex}} \odot S(a) \\ S(a) \\ S(a) \end{pmatrix},$$

$$T_2(a) = \begin{pmatrix} S(a) \\ 03_{\text{hex}} \odot S(a) \\ 02_{\text{hex}} \odot S(a) \\ S(a) \end{pmatrix} \quad \text{und} \quad T_3(a) = \begin{pmatrix} S(a) \\ S(a) \\ 03_{\text{hex}} \odot S(a) \\ 02_{\text{hex}} \odot S(a) \end{pmatrix},$$

wobei $S\colon \mathbb{F}_{256} \to F_{256}$ die Bytesubstitution ist.

Ist dann (bei Matrizenschreibweise) $\vec{b}_j$ der j-te Spaltenvektor des Zustands zu Beginn einer Runde, $\vec{e}_j$ der entsprechende Vektor am Ende der Runde und $\vec{k}_j$ der j-te Spaltenvektor des Rundenschlüssels, so ist

$$\vec{e}_j = T_0(b_{0j}) \oplus T_1(b_{1,j\ominus 1}) \oplus T_2(b_{2,j\ominus c_2}) \oplus T_3(b_{3,j\ominus c_3}) \oplus \vec{k}_j \; ;$$

dabei stehen c_2 und c_3 die Beträge, um die die zweite und dritte Zeile verschoben werden, d.h.

$$c_2 = \begin{cases} 2 & \text{für } N_b < 8 \\ 3 & \text{für } N_b = 8 \end{cases} \quad \text{und} \quad c_3 = \begin{cases} 2 & \text{für } N_b < 7 \\ 3 & \text{für } N_b \geq 7 \end{cases},$$

und $\ominus$ steht für die Subtraktion modulo N_b, wie man sie für die Zeilenshifts braucht. Man beachte, daß auch diese Rechnung bei entsprechender Hardware für alle N_b Spalten parallel ausgeführt werden kann.

Da sich auch die sämtlichen Rundenschlüssel mit einem Speicheraufwand von weniger als einem halben Kilobyte vorausberechnen lassen, läßt sich die Verschlüsselung mittels Rijndael also auch auf einem Standard-PC sehr schnell durchführen.

Bei anderen Implementierungen, bei denen es Probleme mit dem Speicherplatz gibt, kann man auf Kosten einer höheren Rechenzeit mit sehr viel weniger Speicher auskommen. Eine relativ billige Art und Weise, wie man bei 32-Bit-Prozessoren drei Kilobyte einsparen kann, folgt beispielsweise aus der Beobachtung, daß die verschiedenen $T_i(a)$ durch zyklische Verschiebung auseinander entstehen. Somit reicht es, die $T_0(a)$ abzuspeichern, und indem man analog zum HORNER-Schema rechnet, liegt der zusätzliche Rechenaufwand nur bei drei zyklischen Vertauschungen pro Spalte und pro Runde: Ist $\mathcal{Z}$ die zyklische Linksverschiebung um ein Byte, so ist

$$\vec{e}_j = \vec{k}_j \oplus T_0(b_{0j}) \oplus \mathcal{Z}\Big(T_0(b_{1,j-1}) \oplus \mathcal{Z}\Big(T_0(b_{2,j-c_2}) \oplus \mathcal{Z}\big(T_0(b_{3,j-c_3})\big)\Big)\Big) \; .$$

Für die Entschlüsselung braucht man natürlich entsprechende Tabellen; hier werden die Produkte mit $0B_{\text{hex}}, 0D_{\text{hex}}, 09_{\text{hex}}$ und $0E_{\text{hex}}$ benötigt.

Das Arbeiten mit Tabellen kann übrigens unter Sicherheitsaspekten vorteilhaft sein gegenüber direktem Rechnen: Ein Gegner, der den Stromverbrauch der Rechnung kontinuierlich messen kann (z.B. weil eine Smartcard ihren Strom aus seinem Lesegerät bezieht), kann daraus eventuell Rückschlüsse auf Klartext und/oder Schlüssel ziehen, da etwa eine Ziffer eins in einer Multiplikation mehr Aufwand erfordert als eine Ziffer null. Bei Multiplikation via Tabellen ist der Stromverbrauch jedoch unabhängig von den Faktoren, da stets nur ein Tabellenwert gelesen und weiterverwendet wird.

§4: Angriffe auf Rijndael

Wie jede andere Blockchiffre bietet auch Rijndael keinerlei Sicherheit gegen einen BAYESschen Gegner – zumindest dann nicht, wenn man (wie praktisch immer) redundante Information verschlüsselt. Da allerdings bereits die einfachste Variante von Rijndael mit einer Schlüssellänge von 128 arbeitet, ist das Durchprobieren aller Schlüssel für real existierende Gegner mit heutiger Technologie unrealistisch: Gegenüber DES mit seiner Schlüssellänge 56 steigt der Aufwand immerhin um den Faktor

$$2^{128-56} = 2^{72} = 4\,722\,366\,482\,869\,645\,213\,696\,,$$

also um mehr als zwanzig Größenordnungen. Ein Gegner muß daher, um erfolgreich zu sein, Methoden finden, die mit deutlich geringerem Aufwand auskommen. Da er in der Wahl seiner Methoden frei ist, können wir nie wirklich wissen, wie er arbeitet; alle Sicherheitsaussagen beruhen nur darauf, daß keine realistische Attacke *bekannt* ist, obwohl im Verlauf des Begutachtungsprozesses international führende Experten mehrere Jahre lang danach gesucht haben. Natürlich geht die Suche auch jetzt noch weiter, und in der Tat sind inzwischen auch neue Ansätze aufgetaucht, die bei der Wahl von Rijndael noch nicht bekannt waren.

Natürlich ist das Design von Rijndael so gewählt, daß der Algorithmus resistent ist gegen differentielle und lineare Kryptanalyse; beides war schließlich zum Zeitpunkt seines Entwurfs wohlbekannt und gut verstanden. Auch ist die grundsätzliche Struktur von Rijndael nicht neu: Es ist zwar kein FEISTEL-Netzwerk mehr, aber er kommt aus einer Familie

von Kryptoverfahren um den Algorithmus Square, der bereits seit einiger Zeit kryptanalysiert worden war. Insbesondere hatten DAEMEN und RIJMEN die sogenannte Square attack entwickelt, die mit speziell gewählten Klartexten den letzten Rundenschlüssel angreift: Verschlüsselt werden 256 Blöcke, die in allen Bytes mit einer Ausnahme übereinstimmen; das variable Byte nimmt alle 256 möglichen Werte an.

Verfolgt man diese Blöcke geschickt durch den Algorithmus, läßt sich mit hinreichend vielen Gruppen solcher Blöcke auch ein auf sechs Runden reduzierter Rijndael kryptanalysieren; da dies den beiden Designern bewußt war, ist die Rundenzahl entsprechend größer gewählt.

Ein Kritikpunkt an Rijndael war seine relativ einfache algebraische Struktur, die zwar die Implementierung erleichtert und beschleunigt, aber eventuell auch Sicherheitsprobleme aufwerfen könnte.

Im Mai 2001 gelang es NIELS FERGUSON, RICHARD SCHROEPPEL und DOUG WHITING, die allesamt in der Wirtschaft (bei verschiedenen Unternehmen) über Sicherheitsfragen arbeiten, Rijndael in einer geschlossenen Formel darzustellen; für die 128 Bit Version hat diese Formel $2^{50} \approx 10^{15}$ Terme, für 256 Bit sind es $2^{70} \approx 10^{21}$. Obwohl die Formel natürlich hoch strukturiert ist, ist allerdings völlig unklar, ob dieser Ansatz je zu einer Bedrohung für Rijndael werden kann; schließlich ist eine Formel nur selten die beste Möglichkeit für den Umgang mit einer Funktion. Die Originalarbeit ist zu finden unter `www.xs4all.nl/ vorpal/pubs/rdalgeq.html` .

Potentiell gefährlicher ist ein Ansatz von NICOLAS COURTOIS und JOSEF PIEPRZYK, deren XSL-Attacke (`eprint.iacr.org/2002/044/`) das Knacken von Rijndael übersetzt in die Lösung eines überbestimmten nichtlinearen Gleichungssystems. Ob dieser Ansatz langfristig zu einem Angriff führt, der schneller ist als das Durchprobieren aller Schlüssel, ist im Augenblick noch nicht abzuschätzen.

§5: Literatur

Die beste und ausführlichste Quelle ist das Buch der beiden Autoren von Rijndael:

JOAN DAEMEN, VINCENT RIJMEN: The Design of Rijndael: AES – the Advanced Encryption Standard, *Springer,* 2002

Eine neuere Darstellung, die auch auf seither diskutierte Angriffsmöglichkeiten eingeht, ist

CARLOS CID, SEAN MURPHY, MATTHEW ROBSHAW: Algebraic Aspects of the Advanced Encryption Standard, *Springer,* 2006

Kürzere Darstellungen findet man in neueren Lehrbüchern der Kryptologie, z.B.

DOUGLAS R. STINSON: CRYPTOGRAPHY – THEORY AND PRACTICE, Chapman & Hall/CRC, ^{3}2006

oder

RICHARD A. MOLLIN: AN INTRODUCTION TO CRYPTOGRAPHY, Chapman & Hall/CRC, 2001