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

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

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

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

$$\begin{aligned} &63_{\text{hex}} + 05_{\text{hex}} Z \quad + 09_{\text{hex}} Z^2 \ + \text{F}9_{\text{hex}} Z^4 \ + 25_{\text{hex}} Z^8 \\ &\quad + \text{F}4_{\text{hex}} Z^{16} + 01_{\text{hex}} Z^{32} + \text{B}5_{\text{hex}} Z^{64} + 8\text{F}_{\text{hex}} Z^{128} \,. \end{aligned}$$

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

$$\begin{aligned} &63 + 05_{\text{hex}} Z^{254} + 09_{\text{hex}} Z^{253} + \text{F}9_{\text{hex}} Z^{251} + 25_{\text{hex}} Z^{247} \\ &\quad + \text{F}4_{\text{hex}} Z^{239} + 01_{\text{hex}} Z^{223} + \text{B}5_{\text{hex}} Z^{191} + 8\text{F}_{\text{hex}} Z^{127} \,. \end{aligned}$$

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

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

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

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

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

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

DOUGLAS R. STINSON: CRYPTOGRAPHY – THEORY AND PRACTICE, Chapman & Hall/CRC, [2]2002

oder

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

Kapitel 5
Das RSA-Verfahren

§ 1: New directions in cryptography

Bei allen bisher betrachteten Codes verläuft die Entschlüsselung entweder genauso oder zumindest sehr ähnlich wie die Verschlüsselung; insbesondere kann jeder, der eine Nachricht verschlüsseln kann, jede andere entsprechend verschlüsselte Nachricht auch entschlüsseln. Man bezeichnet diese Verfahren daher als *symmetrisch*.

Der Nachteil eines symmetrischen Verfahrens besteht darin, daß in einem Netzwerk jeder Teilnehmer mit jedem anderen einen Schlüssel vereinbaren muß. In militärischen Netzen war dies traditionellerweise so geregelt, daß das gesamte Netz denselben Schlüssel benutzte, der in einem Codebuch für jeden Tag im voraus festgelegt war; in kommerziellen Netzen wie beispielsweise einem Mobilfunknetz ist dies natürlich unmöglich.

1976 publizierten MARTIN HELLMAN, damals Assistenzprofessor in Stanford, und sein Forschungsassistent WHITFIELD DIFFIE eine Arbeit mit dem Titel *New directions in cryptography* (IEEE Trans. Inform. Theory **22**, 644–654), in der sie vorschlugen, den Vorgang der *Ver*schlüsselung und den der *Ent*schlüsselung völlig voneinander zu trennen: Es sei schließlich nicht notwendig, daß der Sender einer verschlüsselten Nachricht auch in der Lage sei, diese zu *ent*schlüsseln.

Der Vorteil eines solchen Verfahrens wäre, daß jeder potentielle Empfänger nur einen einzigen Schlüssel bräuchte und dennoch sicher sein könnte, daß nur er selbst seine Post entschlüsseln kann. Der Schlüssel

müßte nicht einmal geheimgehalten werden, da es ja nicht schadet, wenn jedermann Nachrichten *ver*schlüsseln kann. In einem Netzwerk mit n Teilnehmern bräuchte man also nur n Schlüssel, um es jedem Teilnehmer zu erlauben, mit jeden anderen so zu kommunizieren, und diese Schlüssel könnten sogar in einem öffentlichen Verzeichnis stehen. Bei einem symmetrischen Kryptosystem wäre der gleiche Zweck nur erreichbar mit $\frac{1}{2}n(n-1)$ Schlüsseln, die zudem noch durch ein sicheres Verfahren wie etwa ein persönliches Treffen oder durch vertrauenswürdige Boten ausgetauscht werden müßten.

DIFFIE und HELLMAN machten nur sehr vage Andeutungen, wie so ein System mit öffentlichen Schritten aussehen könnte. Es ist zunächst einmal klar, daß ein solches System keinerlei Sicherheit gegen einen BAYESschen Gegner bieten kann, denn die Verschlüsselungsfunktion ist eine bijektive Abbildung zwischen endlichen Mengen, und jeder, der die Funktion kennt, kann zumindest im Prinzip auch ihre Umkehrfunktion berechnen.

Wer im Gegensatz zum BAYESschen Gegner nur über begrenzte Ressourcen verfügt, kann diese Berechnung allerdings möglicherweise nicht mit realistischem Aufwand durchführen, und nur darauf beruht die Sicherheit eines Kryptosystems mit öffentlichen Schlüsseln. DIFFIE und HELLMAN bezeichnen eine Funktion, deren Umkehrfunktion nicht mit vertretbarem Aufwand berechnet werden kann, als *Einwegfunktion* und schlagen als Verschlüsselungsfunktion eine solche Einwegfunktion vor.

Damit hat man aber noch kein praktikables Kryptosystem, denn bei einer echten Einwegfunktion ist es auch für den legitimen Empfänger nicht möglich, seinen Posteingang zu entschlüsseln. DIFFIE und HELLMAN schlagen deshalb eine Einwegfunktion mit *Falltür* vor, wobei der legitime Empfänger zusätzlich zu seinem öffentlichen Schlüssel noch über einen geheimen Schlüssel verfügt, mit dem er (und nur er) diese Falltür öffnen kann.

Natürlich hängt alles davon ab, ob es solche Einwegfunktionen mit Falltür wirklich gibt. DIFFIE und HELLMAN gaben keine an, und es gab unter den Experten einige Skepsis bezüglich der Möglichkeit, solche Funktionen zu finden.

Tatsächlich aber gab es damals bereits Systeme, die auf solchen Funktionen beruhten, auch wenn sie nicht in der offenen Literatur dokumentiert waren: Die britische *Communications-Electronics Security Group* (CESG) hatte bereits Ende der sechziger Jahre damit begonnen, nach entsprechenden Verfahren zu suchen, um die Probleme des Militärs mit dem Schlüsselmanagement zu lösen, aufbauend auf (impraktikablen) Ansätzen von AT&T zur Sprachverschlüsselung während des zweiten Weltkriegs. Die Briten sprachen nicht von Kryptographie mit öffentlichen Schlüsseln, sondern von *nichtgeheimer Verschlüsselung*, aber das Prinzip war das gleiche.

Erste Ideen dazu sind in einer auf Januar 1970 datierten Arbeit von JAMES H. ELLIS zu finden, ein praktikables System in einer auf den 20. November 1973 datierten Arbeit von CLIFF C. COCKS. Wie im Milieu üblich, gelangte nichts über diese Arbeiten an die Öffentlichkeit; erst 1997 veröffentlichten die *Goverment Communications Headquarters* (GCHQ), zu denen CESG gehört, einige Arbeiten aus der damaligen Zeit; eine Zeitlang waren sie auch auf dem Server `http://www.cesg.gov.uk/` zu finden, wo sie allerdings inzwischen anscheinend verschwunden sind.

Im akademischen Bereich gab es ein Jahr nach Erscheinen der Arbeit von DIFFIE und HELLMAN das erste Kryptosystem mit öffentlichen Schlüsseln: Drei Professoren am Massachussetts Institute of Technology fanden nach rund vierzig erfolglosen Ansätzen 1977 schließlich jenes System, das heute nach ihren Anfangsbuchstaben mit RSA bezeichnet wird: RON RIVEST, ADI SHAMIR und LEN ADLEMAN.

Das System wurde 1983 von der eigens dafür gegründeten Firma RSA Computer Security Inc. patentiert und mit großem kommerziellem Erfolg vermarktet. Das Patent lief zwar im September 2000 aus, die Firma ist aber weiterhin erfolgreich im Kryptobereich tätig; sie hatte beispielsweise auch einen Kandidaten für AES entwickelt, der es immerhin bis in die Endrunde schaffte.

RSA ist übrigens identisch mit dem von COCKS vorgeschlagenen System, so daß einige Historiker auch Zweifel an den Behauptungen der GCHQ haben. Die Beschreibung durch RIVEST, SHAMIR und ADLEMAN

erschien 1978 unter dem Titel *A method for obtaining digital signatures and public-key cryptosystems* in Comm. ACM **21**, 120–126.

§ 2: Algebraische Vorbereitungen

Das RSA-Verfahren hängt ab von Multiplikationen modulo einer großen Zahl N, die Produkt zweier Primzahlen p und q ist; wir wollen uns daher zunächst etwas mit dem Rechnen modulo einer solchen Zahl N beschäftigen.

Wir wissen bereits, daß die ganzen Zahlen modulo einer Primzahl p einen Körper bilden, den Körper $\mathbb{F}_p$ mit p Elementen. Die für das RSA-Verfahren wichtigste Eigenschaft des Rechnens modulo p ist der folgende Satz:

Kleiner Satz von Fermat: Für jedes $a \in \mathbb{Z}$ ist

$$a^p \equiv a \bmod p\,;$$

ist a nicht durch p teilbar, ist auch

$$a^{p-1} \equiv 1 \bmod p\,.$$

Beweis: Für zwei Zahlen $x, y \in \mathbb{Z}$ ist

$$(x+y)^p = \binom{p}{0}x^p + \binom{p}{1}x^{p-1}y + \cdots + \binom{p}{p-1}xy^{p-1} + \binom{p}{p}y^p$$

mit

$$\binom{p}{j} = \frac{p!}{j!(p-j)!}\,.$$

Für $j = 0$ und $j = p$ ist $\binom{p}{0} = \binom{p}{p} = 1$, ansonsten sind j und $p-j$ beide kleiner als p. Im Nenner steht daher dann kein Faktor p, der das p aus dem Zähler wegkürzen könnte, so daß alle $\binom{p}{j}$ mit $1 \le j \le p-1$ durch p teilbar sind, und

$$(x+y)^p \equiv x^p + y^p \bmod p\,.$$

Induktiv folgt, daß eine entsprechende Gleichung auch für Summen mit mehr als zwei Summanden gilt, insbesondere gilt also für beliebig viele Summanden eins, daß

$$(1+\cdots+1)^p \equiv 1^p + \cdots + 1^p = 1 + \cdots + 1 \bmod p\,.$$

Damit ist $a^p \equiv a \bmod p$ für jede natürliches Zahl a. Für $a = 0$ gilt die Beziehung auch, und da für jedes positive a

$$0 = \big(a + (-a)\big)^p = a^p + (-a)^p = a + (-a)^p \bmod p$$

ist, gilt sie auch für negative a.

Falls a nicht durch p teilbar ist, gibt es, da $\mathbb{F}_p$ ein Körper ist, ein b mit $ba \equiv 1 \bmod p$; mit diesem b ist

$$a^{p-1} \equiv (ba)a^{p-1} = ba^p \equiv ba \equiv 1 \bmod p\,.$$

■

Korollar: Falls die natürliche Zahl e keinen Teiler mit $p-1$ gemeinsam hat, gibt es eine natürliche Zahl d, so daß

$$\big(a^e\big)^d \equiv a \bmod p$$

für alle $a \in \mathbb{Z}$.

Beweis: Nach dem erweiterten EUKLIDischen Algorithmus läßt sich der größte gemeinsame Teiler eins von e und $p-1$ als ganzzahlige Linearkombination dieser beiden Zahlen schreiben, es gibt also ganze Zahlen d und r, so daß

$$d \cdot e + r \cdot (p-1) = 1$$

ist. Hier könnte d eventuell noch negativ sein, aber indem wir nötigenfalls noch ein Vielfaches der Gleichung

$$(p-1) \cdot e - e \cdot (p-1) = 0$$

dazuaddieren, können wir annehmen, daß d positiv ist (und r dann natürlich negativ). Für nicht durch p teilbares a ist dann

$$\big(a^e\big)^d = a^{de} = a^{1-r(p-1)} = a \cdot \big(a^{p-1}\big)^{-r} \equiv a \cdot 1^{-r} = a \bmod p\,,$$

wie behauptet.

Für durch p teilbares a sind beide Seiten der zu beweisenden Kongruenz durch p teilbar, so daß sie auch in diesem Fall richtig ist.

■

Für RSA interessiert, wie bereits erwähnt, vor allem das Rechnen modulo des Produkts zweier Primzahlen; hier gilt entsprechend

Satz: Ist $N = pq$ Produkt zweier verschiedener Primzahlen p und q, so gilt
a) Für jede ganze Zahl a ist $a^{1+(p-1)(q-1)} \equiv a \bmod N$.
b) Falls die natürliche Zahl e keinen Teiler mit $(p-1)(q-1)$ gemeinsam hat, gibt es eine natürliche Zahl d, so daß

$$\big(a^e\big)^d \equiv a \bmod p$$

für alle $a \in \mathbb{Z}$.
c) Die Berechnung von $(p-1)(q-1)$ aus N ist äquivalent zur Faktorisierung von N.

Beweis: a) Nach dem kleinen Satz von FERMAT ist für jedes nicht durch p teilbare a und jedes r

$$a^{1+r(p-1)} \equiv a \bmod p\,.$$

Für Vielfache von p sind beide Seiten durch p teilbar, so daß die Gleichung tatsächlich für alle ganzen Zahlen a gilt.

Entsprechendes gilt natürlich auch für die Primzahl q, und damit ist für jede ganze Zahl s

$$a^{1+s(p-1)(q-1)} \equiv a \bmod p \quad \text{und} \quad a^{1+(p-1)(q-1)} \equiv a \bmod q\,.$$

Diese beiden Kongruenzen besagen, daß die Differenz zwischen linker und rechter Seite sowohl durch p als auch durch q teilbar ist, also auch durch deren Produkt N, d.h.

$$a^{1+s(p-1)(q-1)} \equiv a \bmod N\,.$$

b) Geht wegen *a)* genau wie beim Primzahlfall im obigen Korollar.

c) Falls die Faktorisierung $N = pq$ bekannt ist, läßt sich natürlich leicht $(p-1)(q-1)$ berechnen. Ist umgekehrt

$$(p-1)(q-1) = pq - p - q + 1 = N + 1 - (p+q)$$

bekannt, so kennt man außer dem Produkt N auch die Summe M der beiden Primfaktoren, und diese sind die leicht berechenbaren Lösungen der quadratischen Gleichung $x^2 - Mx + N = 0$.

■

§3: Das RSA-Verfahren zur Verschlüsselung und für elektronische Unterschriften

Zur praktischen Durchführung des RSA-Verfahrens wählt man sich zwei verschiedene Primzahlen p, q, die unbedingt geheim gehalten werden müssen, und eine natürliche Zahl e, die keinen gemeinsamen Teiler mit $(p-1)(q-1)$ hat. Dann veröffentlicht man die beiden Zahlen $N = pq$ und e als öffentlichen Schlüssel.

Sodann berechnet man zu $(p-1)(q-1)$ gemäß obigem Satz nach dem EUKLIDischen Algorithmus eine Zahl d, so daß

$$\left(a^e\right)^d \equiv a \bmod p$$

für alle a; diese Zahl ist der geheime Schlüssel.

a) Verschlüsselung

Jeder, der den öffentlichen Schlüssel (N, e) kennt, kann Nachrichten verschlüsseln: Er bricht die Nachricht auf in Blöcke, die durch ganze Zahlen zwischen null und $N-1$ dargestellt werden können, berechnet für jeden so dargestellten Block den Chiffretext $b \equiv a^e \bmod N$, der als Zahl zwischen null und $N-1$ interpretiert und an den Inhaber des geheimen Schlüssels geschickt wird. (Wir schreiben in solchen Fällen in Zukunft kurz $b = a^e \bmod N$, wobei „mod“ in diesem Zusammenhang als die Berechnung des Divisionsrests bei Division durch N zu interpretieren ist.)

Der Empfänger berechnet $b^d \bmod N$; da

$$b^d \equiv a^{ed} \equiv a \bmod N$$

ist, entschlüsselt dies die Nachricht.

b) Identitätsnachweis

Im Gegensatz zu symmetrischen Kryptoverfahren endet die Nützlichkeit des RSA-Verfahrens nicht mit der bloßen Möglichkeit einer Verschlüsselung; das Verfahren kann beispielsweise auch benutzt werden, um in Zugangskontrollsystemen, vor Geldautomaten oder bei einer Bestellung im Internet die Identität einer Person Zu beweisen: Nur der

Inhaber des geheimen Schlüssels d kann zu einem gegebenen a eine Zahl b berechnen, für die $b^e \equiv a \bmod N$ ist.

Falls also der jeweilige Gegenüber eine Zufallszahl a erzeugt und als Antwort das zugehörige b verlangt, kann er anhand eines öffentlichen Schlüsselverzeichnisses die Richtigkeit von b überprüfen und sich so von der Identität seines Partners überzeugen. Im Gegensatz zu Kreditkarteninformation oder Paßwörtern ist dieses Verfahren auch immun gegen Abhören: Falls jedesmal ein neues zufälliges a erzeugt wird, nützt ein einmal abgehörtes b nichts.

Trotzdem ist das Verfahren in dieser Form nicht als Ersatz zur Übertragung von Kreditkarteninformation oder ähnlichem geeignet, da der Gegenüber anhand des öffentlichen Schlüssels jederzeit zu einer willkürlich gewählten Zahl b die Zahl $a = b^e \bmod N$ erzeugen kann um dann zu behaupten, er habe b als Antwort darauf empfangen. Man müßte also beispielsweise noch zusätzlich verlangen, daß die Zahl a eine spezielle Form hat, etwa daß die vordere Hälte der Ziffernfolge identisch mit der hinteren Hälfte ist.

c) Elektronische Unterschriften

Praktische Bedeutung hat vor allem eine weitere Variante: die elektronische Unterschrift. Her geht es darum, daß der Empfänger erstens davon überzeugt wird, daß eine Nachricht tatsächlich vom behaupteten Absender stammt, und daß er dies zweitens auch einem Dritten gegenüber beweisen kann. (In Deutschland sind solche elektronischen Unterschriften, sofern gewisse formale Voraussetzungen erfüllt sind, rechtsverbindlich.)

Um einen Nachrichtenblock a mit $0 \leq a < N$ zu unterschreiben, berechnet der Inhaber des öffentlichen Schlüsseln (N, e) mit seinem geheimen Schlüssel d die Zahl

$$b = a^d \bmod N$$

und sendet das Paar (a, b) an den Empfänger. Dieser überprüft, ob

$$b^e \equiv a \bmod N\,;$$

falls ja, akzeptiert er dies als unterschriebene Nachricht a. Da er ohne Kenntnis des geheimen Schlüssels d nicht in der Lage ist, den Block (a, b) zu erzeugen, kann er auch gegenüber einem Dritten beweisen, daß der Absender die Nachricht a unterschrieben hat.

Für kurze Nachrichten ist dieses Verfahren in der vorgestellten Form praktikabel; in vielen Fällen kann man sogar auf die Übermittlung von a verzichten, da $b^e \bmod N$ für ein falsch berechnetes b mit an Sicherheit grenzender Wahrscheinlichkeit keine sinnvolle Nachricht ergibt.

Falls die übermittelte Nachricht geheimgehalten werden soll, müssen a und b natürlich noch vor der Übertragung mit dem öffentlichen Schlüssel des Empfängers oder nach irgendeinem anderen Kryptoverfahren verschlüsselt werden.

Bei langen Nachrichten ist die Verdoppelung der Nachrichtenlänge nicht mehr akzeptabel, und selbst, wenn man auf die Übertragung von a verzichten kann, ist das Unterschreiben jedes einzelnen Blocks sehr aufwendig. Deshalb unterschreibt man meist nicht die Nachricht selbst, sondern einen daraus extrahierten Hashwert. Dieser Wert muß natürlich erstens von der gesamten Nachricht abhängen, und zweitens muß es für den Empfänger (praktisch) unmöglich sein, zwei Nachrichten zu erzeugen, die zum gleichen Hashwert führen. Mit der Theorie (und Praxis) dieser sogenannten kollisionsfreien Hashfunktionen werden wir uns später beschäftigen.

d) Blinde Unterschriften und elektronisches Bargeld

Einer der erfolgversprechendsten Ansätze zum Aushebeln eines Kryptosystems besteht darin, sich auf die Dummheit seiner Mitmenschen zu verlassen.

So sollte es durch gutes Zureden nicht schwer sein, jemanden zu Demonstrationszwecken zum Unterschreiben einer sinnlosen Nachricht zu bewegen: Eine Folge von Nullen und Einsen ohne sinnvolle Interpretation hat schließlich keine rechtliche Wirkung.

Nun muß eine sinnlose Nachricht aber nicht unbedingt eine Zufallszahl sein: Sie kann sorgfältig präpariert sein. Sei dazu etwa m eine Nachricht,

die ein Zahlungsversprechen enthält, (N, e) der öffentliche Schlüssel des Opfers und r eine Zufallszahl zwischen 2 und $N - 2$. Dann wird

$$x = m \cdot r^e \bmod N$$

wie eine Zufallsfolge aussehen, für die man eine Unterschrift

$$u = x^d \bmod N = (mr^e)^d \bmod N = m^d r \bmod N$$

bekommt. Multiplikation mit r^{-1} macht daraus eine Unterschrift unter die Zahlungsverpflichtung m.

Das angegebene Verfahren kann nicht nur von Trickbetrügern benutzt werden; blinde Unterschriften sind auch die Grundlage von *digitalem Bargeld*.

Zahlungen im Internet erfolgen meist über Kreditkarten; die Kreditkartengesellschaften haben also einen recht guten Überblick über die Ausgaben ihrer Kunden und machen teilweise auch recht gute Geschäfte mit Kundenprofilen.

Digitales Bargeld will die Anonymität von Geldscheinen mit elektronischer Übertragbarkeit kombinieren und so ein anonymes Zahlungssystem z.B. für das Internet bieten.

Es wir ausgegeben von einer Bank, die für jede angebotene Stückelung einen öffentlichen Schlüssel (N, e) bekanntgibt. Eine Banknote ist eine mit dem zugehörigen geheimen Schlüssel unterschriebene Seriennummer.

Die Seriennummer kann natürlich nicht einfach *jede* Zahl sein; sonst wäre jede Zahl kleiner N eine Banknote. Andererseit dürfen die Seriennummern aber auch nicht von der Bank vergeben werden, denn sonst wüßte diese, welcher Kunde Scheine mit welchem Seriennummern hat. Als Ausweg wählt man Seriennummern einer sehr speziellen Form: Ist $N > 10^{150}$, kann man etwa als Seriennummer eine 150-stellige Zahl wählen, deren Ziffern spiegelsymmetrisch zur Mitte sind, d.h. ab der 76. Ziffer werden die vorherigen Ziffern rückwärts wiederholt. Die Wahrscheinlichkeit, daß eine zufällige Zahl x nach Anwendung des öffentlichen Exponenten auf so eine Zahl führt, ist 10^{-75} und damit vernachlässigbar.