2
Kryptologische Grundlagen

In diesem Kapitel werden grundlegende kryptologische Mechanismen dargestellt. Diese wurden zunächst dafür entwickelt, die in Kap. 1 dargestellten Ziele zu verwirklichen. Für uns sind diese Mechanismen vor allem deswegen wichtig, weil sie als Grundbausteine komplexer Protokolle Verwendung finden.

2.1 Verschlüsselung

Der älteste Zweig der Kryptographie beschäftigt sich mit der Geheimhaltung von Nachrichten durch Verschlüsselung. Das Ziel ist dabei, die Nachricht so zu verändern, dass nur der berechtigte Empfänger, aber kein Angreifer diese lesen kann. Damit dies möglich ist, muss der Empfänger dem Angreifer eine Information voraushaben. Diese geheime Information wird **Schlüssel** genannt; mit Hilfe des Schlüssels kann der Empfänger den empfangenen Geheimtext **entschlüsseln**.

In der klassischen Kryptologie wird dies so realisiert, dass auch der Sender diesen geheimen Schlüssel besitzt, und sich Sender und Empfänger mit Hilfe ihres geheimen Schlüssels gegen die Außenwelt schützen. Man nennt solche Verfahren auch **symmetrische** Verschlüsselungsverfahren (siehe Abb. 2.1).

Genauer gesagt: Ein symmetrischer Verschlüsselungsalgorithmus besteht aus einer Funktion f mit zwei Eingabewerten, dem **Schlüssel** k und dem **Klartext** m, und einer Ausgabe, dem **Geheimtext** c, der sich aus k und m ergibt.

Eine Verschlüsselungsfunktion f muss **umkehrbar** sein, d. h. es muss eine Funktion f* geben, welche die Wirkung von f rückgängig macht. Das heißt, dass f* unter dem gleichen Schlüssel k aus dem Geheimtext c wieder den Klartext m rekonstruiert.

Der Sender verschlüsselt eine Nachricht m, indem er

$$c = f(k, m)$$

berechnet, wobei k der gemeinsame geheime Schlüssel von Sender und Empfänger ist. Alternativ dazu schreiben wir auch $c := f_k(m)$. Der Empfänger kann

DOI 10.1007/978-3-8348-2322-9_2

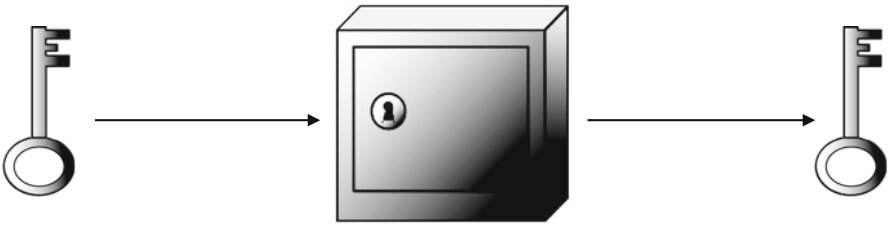

Abb. 2.1 Symmetrische Verschlüsselung. *Verschlüsselung schützt die Nachricht vor dem Gelesenwerden. Man kann sich vorstellen, dass der Sender die Nachricht in einen „Tresor" legt und diesen mit Hilfe seines „Schlüssels" abschließt. Dann schickt er Tresor samt Inhalt an den Empfänger. Nur dieser hat einen Zweitschlüssel, um den Tresor zu öffnen und die Nachricht zu lesen. In der Kryptographie werden die Nachrichten allerdings nicht durch physikalische Maßnahmen geschützt, sondern durch wesentlich elegantere, sicherere und kostengünstigere mathematische Methoden*

mit Hilfe desselben Schlüssels k den Geheimtext entschlüsseln, indem er

$$f^*(k, c) = m$$

berechnet. Dieser Vorgang ist schematisch in Abb. 2.2 dargestellt.

Man muss grundsätzlich davon ausgehen, dass sowohl Ver- als auch Entschlüsselungsfunktion bekannt sind (**Kerckhoffssches Prinzip**). Es gibt zwar auch geheime Verfahren, doch jeder praktisch eingesetzte Algorithmus geht bei der Entwicklung, Spezifikation, Normung, Programmierung, Evaluierung und Implementierung durch so viele Hände, dass es großer Sicherheitsvorkehrungen bedarf, um ihn geheim zu halten.

Eine Verschlüsselungsfunktion f ist **sicher**, wenn sie die folgenden Angriffe übersteht:

- **Ciphertext-only attack**: Der Angreifer kennt eine begrenzte Anzahl von Geheimtexten und möchte daraus die zugehörigen Klartexte bzw. den verwendeten Schlüssel berechnen.

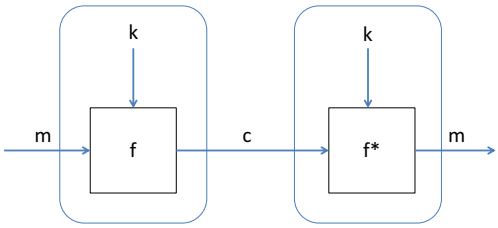

Abb. 2.2 Funktionsschema der symmetrischen Verschlüsselung. *Verschlüsselung c = f(k,m) und Entschlüsselung m = f*(k,c) müssen in einer sicheren Umgebung stattfinden*

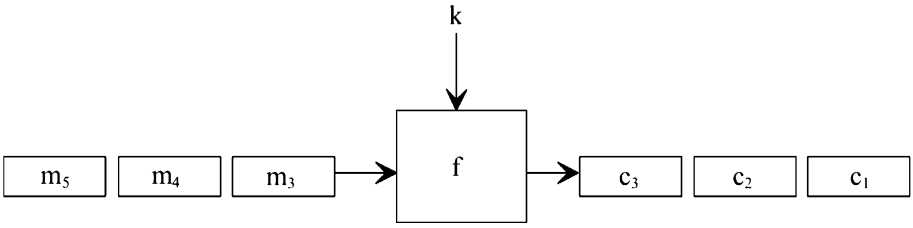

Abb. 2.3 Blockchiffre

- **Known-plaintext attack**: Der Angreifer kennt eine begrenzte Anzahl von Geheimtexten mit den zugehörigen Klartexten und möchte daraus den verwendeten Schlüssel bzw. den Klartext zu einem weiteren Chiffretext berechnen.
- **Chosen-plaintext attack**: Der Angreifer hat sich Zugang zu der mit dem Schlüssel k parametrisierten Verschlüsselungsfunktion f verschafft. Er kann den Schlüssel k zwar nicht auslesen, aber bestimmte von ihm ausgewählte Klartexte (z. B. spezielle Klartexte wie „100000000“) mit Hilfe von f_k verschlüsseln. Mit Hilfe dieser Information möchte er andere Geheimtexte entschlüsseln bzw. den Schlüssel k berechnen (dieser Angriff erscheint zunächst etwas akademisch, er stellt aber bei der im nächsten Kapitel eingeführten Public-Key Kryptographie eine echte Bedrohung dar).
- **Chosen-ciphertext attack**: Der Angreifer hat sich Zugang zu der mit dem Schlüssel k parametrisierten Entschlüsselungsfunktion f^* verschafft. Er kann den Schlüssel k zwar nicht auslesen, aber bestimmte von ihm ausgewählte Geheimtexte mit Hilfe von f_k^* entschlüsseln. Mit Hilfe dieser Information versucht er, den Schlüssel k zu berechnen.

Die Algorithmen zur symmetrischen Verschlüsselung von Daten unterteilt man in **Blockchiffren** und **Stromchiffren**.

Bei einer **Blockchiffre** (Abb. 2.3) wird die Nachricht in Blöcke m_1, m_2, m_3, … fester Länge eingeteilt (eine typische Zahl ist 64 Bit), und jeder Block m_i wird unter Verwendung des Schlüssels einzeln verschlüsselt:

$$c_i = f(k, m_i) \quad \text{für} \quad i = 1, 2, 3, \ldots$$

Beispiele für Blockchiffren sind der **DES** (Data Encryption Standard) und der IDEA (International Data Encryption Algorithm). Beides sind Algorithmen mit einer Blocklänge von 64 Bit; beim DES werden Schlüssel mit 56 Bit benutzt, während ein IDEA-Schlüssel aus 128 Bit besteht.

Der DES wurde 1977 veröffentlicht und hat sich als sehr zuverlässig erwiesen; er ist der im kommerziellen Bereich am häufigsten eingesetzte Algorith-

mus. Allerdings ist seine Schlüssellänge von nur 56 Bit der heute verfügbaren Rechenleistung nicht mehr gewachsen: Am 19. Januar 1999 wurde ein DES-Schlüssel mit einer known-plaintext attack durch vollständige Suche in nur 22 Stunden und 15 Minuten berechnet [EFF99].

Der Nachfolger des DES ist der **AES** (Advanced Encryption Standard), der nach langen und intensiven Studien am 2.10.2000 veröffentlicht wurde. Es handelt sich um eine von J. Daemen und V. Rijmen entwickelte Blockchiffre, die Blocklängen von 128 Bit und Schlüssellängen von 128, 192 oder 256 Bit zulässt. Eine AES-Operation besteht aus 10 bis 14 Runden, abhängig von der Schlüssellänge. Intern werden die Daten als Folge von Bytes behandelt, die in einer vierzeiligen Matrix angeordnet sind. Neben der Addition entsprechender Rundenschlüssel besteht die Verschlüsselungsoperation aus einer Byte-Substitution, einer zyklischen Verschiebung der Zeilen der Matrix und einer Transformation der Spalten der Matrix; diese Operationen werden in jeder Runde wiederholt. Bei der ersten und dritten Operation wird in wesentlicher Weise die Tatsache benützt, dass man ein Byte auch als Element des Körpers $GF(2^8)$ auffassen kann (siehe [Mey76]). Für weitere Details siehe [NIST00].

Mittlerweile wurde die Struktur von AES besser untersucht. So stellte sich z. B. heraus, dass man das 128-bit-AES-Problem (das Problem, den geheimen 128-Bit-Schlüssel aus einem Chiffretext-Klartext-Paar zu bestimmen), als System von 8000 quadratischen Gleichungen mit 1600 binären Unbekannten darstellen kann [CP02].

Blockchiffren können in verschiedenen Modi betrieben werden, die z. B. in [Bih93] beschrieben werden. Weitere Informationen über diese und andere Blockchiffren findet man z. B. im Buch von Fumy und Ries [FR94] oder von Paar und Pelzl [PP10].

Bei einer **Stromchiffre** wird eine Nachricht zeichenweise verschlüsselt. Hierzu wird ein Schlüsselstrom erzeugt, der die gleiche Länge hat wie der Klartext, so dass jeweils ein Klartextzeichen mit einem Schlüsselzeichen zu einem Chiffretextzeichen verknüpft werden kann. Während also bei einer Blockchiffre (jedenfalls im Grundmodus) gleiche Klartextblöcke in gleiche Geheimtextblöcke überführt werden, so wird (und sollte) eine Stromchiffre gleiche Klartextzeichen nicht in gleiche Geheimtextzeichen verschlüsseln.

Der Prototyp aller Stromchiffren ist das **One-Time-Pad** (Abb. 2.4). Dazu muss die Nachricht als Folge von Bits vorliegen. Der Schlüssel ist ebenfalls eine Folge von Bits, die (im Gegensatz zu den Bits der Nachricht) zufällig und unabhängig voneinander gewählt wurden. Die Verschlüsselung ist denkbar einfach: Entsprechende Bits des Klartextes und des Schlüssels werden modu-

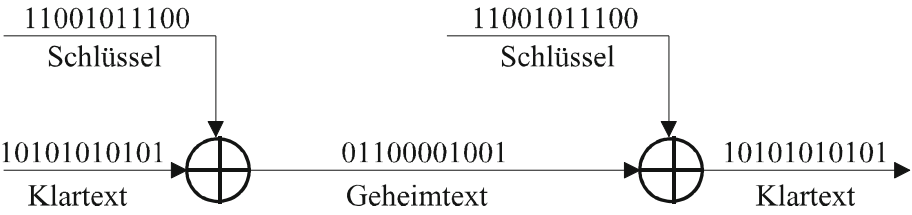

Abb. 2.4 One-time-pad

lo 2 addiert, das heißt nach den Regeln

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1 \oplus 0 = 1 \quad \text{und} \quad 1 \oplus 1 = 0$$

verknüpft.

In diesem Fall verläuft die Entschlüsselung genauso wie die Verschlüsselung, es ist also $f^* = f$: Zur Geheimtextfolge wird die Schlüsselfolge modulo 2 addiert, und man erhält wieder den Klartext.

Wenn der Schlüssel nur einmal verwendet wird, ist dies ein absolut sicheres („perfektes“) Verfahren, dessen Sicherheit sogar theoretisch bewiesen werden kann. Allerdings hat es den Nachteil, dass der Schlüssel genauso lang sein muss wie der Klartext [Sha49]. Wird derselbe Schlüssel mehrmals verwendet, so kann das System mit einem Known-Plaintext-Angriff gebrochen werden.

Bemerkung: Unter einem One-Time-Pad kann man sich einen Abreißblock vorstellen. Auf jedem Blatt steht ein Schlüsselbit; wenn dieses verwendet wurde, wird das Blatt abgerissen und weggeworfen.

In der Praxis benutzt man in der Regel keine perfekten Stromchiffren, sondern setzt zur Erzeugung der Schlüsselfolge einen Pseudozufallsgenerator ein (siehe Abschn. 8.6). Der Vorteil ist dabei, dass nur eine kurze geheime Information zur Initialisierung des Pseudozufallsgenerators benötigt wird; wie bei den Blockchiffren muss also nur eine kurze geheime Information vom Sender zum Empfänger übertragen werden.

Man kann die Herausforderung, die in der Konstruktion praktischer symmetrischer Verschlüsselungsalgorithmen liegt, wie folgt ausdrücken: Man konstruiere ein Verfahren, mit dem man die vertrauliche Übertragung beliebig vieler, beliebig langer Nachrichten auf die einmalige geheime Übermittlung eines kurzen Datensatzes (nämlich des Schlüssels) zurückführen kann.

2.2 Asymmetrische Verschlüsselung

Wir haben uns im vorigen Abschnitt klargemacht, dass zumindest der Empfänger einer verschlüsselten Nachricht einen geheimen Schlüssel braucht. Jahrtausendelang ging man davon aus, dass auch der Sender einen geheimen Schlüssel, und zwar den gleichen wie der Empfänger, benötigt. Die Geburtsstunde der **asymmetrischen Kryptographie** (**Public-Key-Kryptographie**, Abb. 2.5) schlug, als die Frage, ob dies so sein müsse, ernsthaft gestellt wurde. Die sich daran anschließenden Überlegungen führten W. Diffie und M. Hellman 1976 zum Konzept des asymmetrischen Verschlüsselungsverfahrens [DH76].

Jedem Teilnehmer T des Systems wird ein **privater Schlüssel** $d = d_T$ und ein so genannter **öffentlicher Schlüssel** $e = e_T$ zugeordnet. Wie die Namen der beiden Schlüssel schon andeuten, ist dabei nur d_T geheim zu halten; er ist der eigentliche Schlüssel im Sinne von Abschn. 2.1 und wird daher auch oft als **geheimer Schlüssel** bezeichnet. Im Gegensatz dazu sollte e_T möglichst vielen Personen zugänglich sein.

Der Algorithmus f ordnet unter einem öffentlichen Schlüssel e jedem Klartext m einen Geheimtext

$$c = f_e(m)$$

zu. Umgekehrt ordnet f unter jedem privaten Schlüssel d jedem Geheimtext c einen Klartext

$$m' = f_d(c)$$

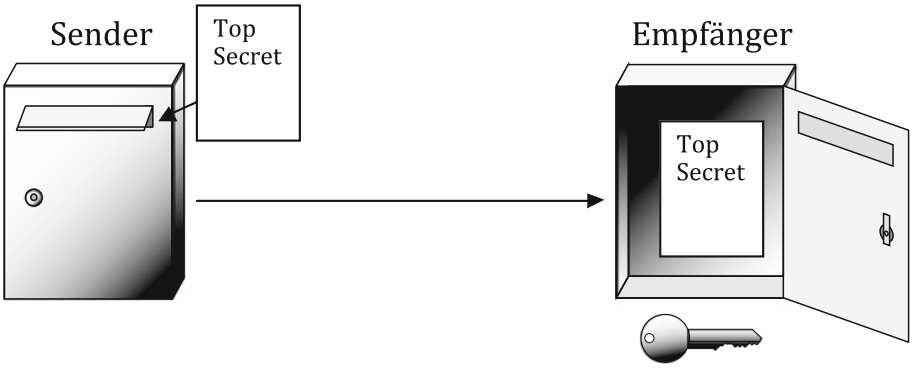

Abb. 2.5 Asymmetrische Verschlüsselung. *Man kann sich asymmetrische Verschlüsselung als Briefkasten vorstellen, in den der Sender die Nachricht wirft. Das Einwerfen der Nachricht in den Briefkasten entspricht der Verschlüsselung mit dem öffentlichen Schlüssel des Empfängers: Jeder Teilnehmer kann das machen. Nur der Empfänger ist aber in der Lage, mit seinem geheimen Schlüssel den Briefkasten zu öffnen und die Nachricht zu lesen*

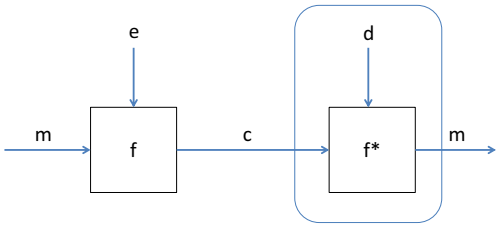

Abb. 2.6 Funktionsschema der asymmetrischen Verschlüsselung. *Nur die Entschlüsselung muss in einer sicheren Umgebung vonstattengehen, um den privaten Schlüssel d zu schützen*

zu. Durch Parametrisierung der Funktion f mit e erhält man also die Verschlüsselungsfunktion f_e, bei Parametrisierung mit d die Entschlüsselungsfunktion f_d (Abb. 2.6). Dabei müssen die folgenden Eigenschaften erfüllt sein.

Korrekte Entschlüsselung Bei der Entschlüsselung muss der korrekte Klartext reproduziert werden; das bedeutet

$$m' = f_d(c) = f_d(f_e(m)) = m$$

für alle Klartexte m.

Public-Key-Eigenschaft Es ist praktisch unmöglich, aus der Kenntnis des öffentlichen Schlüssels e auf den privaten Schlüssel d zu schließen.

Um einem Teilnehmer T eine verschlüsselte Nachricht zu senden, muss man zunächst den öffentlichen Schlüssel $e = e_T$ der Person T ermitteln, welche die Nachricht empfangen soll (Suche in einem elektronischen „Telefonbuch“).

Dann wendet man die Funktion f_e auf die zu verschlüsselnde Nachricht m an und schickt den so erhaltenen Geheimtext

$$c = f_e(m)$$

an T. Nur dieser kann mit Hilfe seines privaten Schlüssels $d = d_T$ die Funktion f_d bilden und damit m entschlüsseln:

$$m = f_d(c) = f_d(f_e(m)).$$

Bemerkungen

1. Die Bezeichnungen e und d für den öffentlichen und den geheimen Schlüssel sind in Anlehnung an die englischen Worte „encryption“ (Verschlüsselung) und „decryption“ (Entschlüsselung) gewählt worden.
2. Oft schreibt man auch E(m) und D(c) statt $f_e(m)$ und $f_d(c)$.

Der RSA-Algorithmus ist der Prototyp für Public-Key-Kryptographie schlechthin; er wird in Abschn. 2.8 behandelt.

2.3 Einwegfunktionen

Eine Einwegfunktion ist eine Funktion, die einfach auszuführen, aber schwer – praktisch unmöglich – zu invertieren ist. Etwas genauer formulieren wir: Eine **Einwegfunktion** (Abb. 2.7) ist eine Abbildung f einer Menge X in eine Menge Y, so dass f(x) für jedes Element von X leicht zu berechnen ist, während es für (fast) jedes y aus Y extrem schwer ist, ein Urbild x (d. h. ein x mit f(x) = y) zu finden.

Wenn eine Einwegfunktion f bijektiv, also eine Permutation ist, so spricht man auch von einer **Einwegpermutation**.

Eine Einwegfunktion heißt **kollisionsfrei**, falls es praktisch unmöglich ist, zwei verschiedene Werte x und x′ in der Urbildmenge X zu finden mit f(x) = f(x′).

Es ist klar, dass jede Einwegpermutation kollisionsfrei ist, denn hier gibt es keinen Funktionswert mit zwei verschiedenen Urbildern, da jeder solche Wert nur genau ein Urbild hat. Weitere Eigenschaften von Einwegfunktionen werden z. B. in [MOV97] diskutiert.

Ein alltägliches Beispiel für eine Einwegfunktion ist ein Telefonbuch; die auszuführende Funktion ist die, einem Namen die entsprechende Telefonnummer zuzuordnen. Da die Namen alphabetisch geordnet sind, ist diese Zuordnung einfach auszuführen. Aber ihre Invertierung, also die Zuordnung eines Namens zu einer *gegebenen* Nummer, ist offensichtlich viel schwieriger, wenn man nur ein Telefonbuch zur Verfügung hat.

Abb. 2.7 Eine Einwegfunktion im Straßenverkehr

Einwegfunktionen spielen sowohl in der theoretischen als auch in der praktischen Kryptographie eine entscheidende Rolle. Zum Beispiel kann man fast alle kryptographischen Begriffe auf den Begriff der Einwegfunktion zurückführen.

Leider weiß man bis heute nicht, ob es Einwegfunktionen überhaupt gibt. Man kann zeigen, dass Einwegfunktionen genau dann existieren, wenn $\mathbf{P} \neq \mathbf{NP}$ gilt (vgl. [BDG88], S. 63). Diese berühmte Vermutung aus der Komplexitätstheorie (siehe Abschn. 9.7) widersteht aber bisher erfolgreich allen Bemühungen, sie zu beweisen. Nach heutigem Wissensstand sind die diskreten Exponentialfunktionen ebenso wie das Quadrieren modulo n (mit $n = pq$) Einwegfunktionen (vgl. die Abschn. 8.3 und 8.4).

Eine wichtige Klasse von Einwegfunktionen sind die, die man aus symmetrischen Verschlüsselungsverfahren konstruieren kann. Dazu wählt man ein solches Verfahren $f(\cdot,\cdot)$, eine feste Nachricht m_0 und erhält die Einwegfunktion

$$F(\cdot) := f(\cdot, m_0),$$

indem man das Argument von F anstelle des Schlüssels in $f(\cdot, m_0)$ einsetzt. Bei einem symmetrischen Verschlüsselungsverfahren muss es auch bei Kenntnis von Klar- und Geheimtext unmöglich sein, auf den verwendeten Schlüssel zu schließen („Known-Plaintext-Attacke"); diese Eigenschaft von f garantiert, dass die Funktion F eine Einwegfunktion ist.

Achtung: Die oben beschriebene Konstruktion ist nicht symmetrisch bezüglich der beiden Argumente von f. Wählt man anstelle eines konstanten Klartextes einen konstanten Schlüssel, so ist die sich daraus ergebende Funktion $G(\cdot) := f(k_0, \cdot)$ *keine* Einwegfunktion, denn man kann für einen Wert y leicht ein Urbild $x := f^*(k_0, y)$ berechnen.

2.4 Kryptographische Hashfunktionen

In der Kryptographie dienen Hashfunktionen dazu, einen unmanipulierbaren „Fingerabdruck" von Nachrichten herzustellen. Eine **Einweg-Hashfunktion** ist eine kollisionsfreie Einwegfunktion, die Nachrichten beliebiger Länge auf einen Hashwert einer festen Länge (typischer Wert: 160 Bit) komprimiert. Man nennt Einweg-Hashfunktionen manchmal auch **kryptographische Hashfunktionen**.

Dies entspricht genau der Vorstellung eines Fingerabdrucks: Aus einem Fingerabdruck kann ich nicht auf den zugehörigen Menschen schließen, und keine zwei Personen haben denselben Fingerabdruck.

Prüfsummen („checksums“) sind nicht brauchbar zur Bildung von kryptographischen Hashfunktionen, da es leicht möglich ist, verschiedene Nachrichten mit derselben Prüfsumme zu konstruieren: Stellen wir uns eine Überweisung vor, bei der ein Betrag von € 2580,– auf das Konto 82677365 übertragen werden soll. Die Prüfsumme sei die Quersumme aller in dieser Überweisung auftretenden Ziffern. Braucht nun ein Angreifer mit der Kontonummer 71234599 Geld, so muss er nur zusätzlich zur Kontonummer noch den Betrag in € 2980,– abändern, und die Prüfsumme merkt den Betrug nicht:

$$2+5+8+0+8+2+6+7+7+3+6+5 = 59$$
$$= 2+9+8+0+7+1+2+3+4+5+9+9$$

Es scheint in der Praxis außerordentlich schwierig zu sein, gute kryptographische Hashfunktionen zu finden. Praktisch eingesetzte Hashfunktionen sind MD5, RIPEMD und SHA-1 und SHA-2; vom Einsatz von MD5 ist abzuraten, da man hier Kollisionen konstruieren kann [WY05]. Einen Überblick zu diesem Thema bieten [Heise05, BR05].

Ein Beispiel aus dem täglichen Leben ist der Übergang von einem Rezept zum Essen bzw. Getränk. So scheint es beispielsweise weder möglich zu sein, aus der Analyse von Coca-Cola auf das Rezept zu schließen (Einwegeigenschaft), noch kann man ein anderes Rezept (eine Kollision) für dieses Getränk finden.

2.5 Trapdoor-Einwegfunktionen

Einwegfunktionen werden zwar in der Kryptographie angewendet (insbesondere als Hashfunktionen), ihr Einsatzbereich ist aber nur auf die Berechnungen beschränkt, die von *allen* Beteiligten durchgeführt werden dürfen. Für die moderne Kryptographie benötigt man daher noch ein weiteres Konzept, nämlich das der Trapdoor-Einwegfunktion.

Eine **Trapdoor-Einwegfunktion** ist eine Einwegfunktion, also eine außerordentlich schwer zu invertierende Funktion, zu der es aber eine Geheiminformation („Geheimtür“, englisch „trapdoor“) gibt, mit Hilfe derer man die Funktion leicht invertieren kann.

Zum Beispiel ist das Quadrieren

$$x \mapsto x^2 \bmod n$$

für $n = pq$ eine Trapdoor-Einwegfunktion, denn ohne Kenntnis der Faktorisierung von n ist es praktisch unmöglich, diese Funktion zu invertieren; mit

http://www.springer.com/978-3-8348-1927-7

Moderne Verfahren der Kryptographie
Von RSA zu Zero-Knowledge
Beutelspacher, A.; Schwenk, J.; Wolfenstetter, K.-D.
2015, XV, 186 S. 86 Abb., Softcover
ISBN: 978-3-8348-1927-7